Nature-inspired survivability: Prey-inspired survivability countermeasures for cloud computing security challenges

Siyakha Njabuliso Mthunzi

STAFFORDSHIRE UNIVERSITY

A Thesis submitted in fulfilment of the requirement of Staffordshire University for the award of the degree of Doctor of Philosophy

2019

To my parents and my family

In memory of my father, the late J.G. Mthunzi

and

to my mother M.G Mthunzi

, who gave me invaluable educational opportunities and boundless permanence throughout my life.

Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma, except where due reference is made in the text of the thesis. To the best of my knowledge, this thesis contains no material previously published or written by another person except where due reference is made in the text of the thesis.

Siyakha Njabuliso Mthunzi

Acknowledgements

Deepest gratitude to my principal supervisor Professor Elhadj Benkhelifa for the insightful guidance and supporting my thesis actively throughout the PhD journey. The continuous cooperation, relentless drive and exchange of ideas made this Thesis possible. Thanks to my secondary supervisor Dr Tomasz Bosakowski for the support and guidance.

A special thanks to my colleagues in the Cloud Computing and Applications Research Laboratory at Staffordshire University, for the interesting discussions and constructive feedback. Many thanks especially to Lavorel Anaïs Léa Céline for the collaboration.

No words can express my gratitude and thanks to my entire family, for the love and support, and the genuine caring you show whatever the weather. Your extraordinary self-sacrifice saw me through overwhelming obstacles. You are my heroes! I am grateful to my friends whose support and encouragement always motivates me to do better.

Finally, a special shout-out to me for never giving up!

Siyakha Njabuliso Mthunzi

Staffordshire University

Table of Contents

List of Tables

List of Figures

Abbreviations

Several key terms are used throughout this document and are defined here:

- API – Application programming interface
- CC – Cloud consumer
- CSP – Cloud service provider
- DDoS – Distributed denial of service
- DoS – Denial of service
- DTMC - Discrete-time Markov chain
- EscS – Escalation system
- eDDoS - Economic distributed denial of service
- IaaS -Infrastructure as a Service
- IoT - the internet of things
- LV – Lotka and Volterra
- MDP - Markov decision processes
- Pi-CCSF – Prey-inspired cloud computing survivability framework
- PaaS – Platform as a Service
- QoS – Quality of service
- RT - Recovery time
- RTO - Recovery time objective
- RO – Recovery objective
- RPO - Recovery point objective
- SaaS – Software as a Service
- SIEM - Security information and event management
- SLA – Service level agreement
- SSM -Survivability strategy manager
- SM – Survivability monitor
- TIPS - the Theory of Inventive Problem Solving
- TRIZ - Teorija Rezbenija Izobretatelskib Zadach
- UUUR – Uncertain malicious actions, latent risks, Unobserved or Unobservable risk
- VM – Virtual machine
- VMM -Virtual machine monitor.

List of publications

Pub	Title	Type
RP 1:	Mthunzi, Siyakha N, BENKHELIFA, Elhadj, Bosakowski, Tomasz and Hariri, Salim (2018) *A Bio-inspired Approach to Cyber Security.* In: Computer Security: Principles, Algorithm, Applications, and Perspectives. Unspecified. ISBN 9780815371335	**Book Chapter**
RP 2:	Mthunzi, S.N., Benkhelifa, E., Bosakowski, T., Guegan, C.G. and Barhamgi, M., 2020. Cloud computing security taxonomy: From an atomistic to a holistic view. Future Generation Computer Systems, 107, pp.620-644.	**FGCS Journal**
RP 3:	Mthunzi, S.N. and Benkhelifa, E., 2017, September. Trends towards Bio-Inspired Security Countermeasures for Cloud Environments. In *2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS* W)* (pp. 341-347). IEEE.	IEEE **Conference**
RP 4:	Mthunzi, S.N. and Benkhelifa, E., 2017, October. Survivability analogy for cloud computing. In *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)* (pp. 1056-1062). IEEE.	**IEEE Conference**
RP 5:	Mthunzi, S.N., Benkhelifa, E., Alsmirat, M.A. and Jararweh, Y., 2018, April. Analysis of VM communication for VM-based cloud security systems. In *2018 Fifth International Conference on Software Defined Systems (SDS)* (pp. 182-188). IEEE.	**IEEE Conference**
RP 6:	Mthunzi, S. and Benkhelifa, E., 2018, April. Mimicking Prey's escalation predation-avoidance techniques for cloud computing survivability using fuzzy cognitive map. In *2018 Fifth International Conference on Software Defined Systems (SDS)* (pp. 189-196). IEEE.	**IEEE Conference**

Abstract

As cloud computing environments become complex, adversaries have become highly sophisticated and unpredictable. Moreover, they can easily increase attack power and persist longer before detection. Uncertain malicious actions, latent risks, Unobserved or Unobservable risks (UUURs) characterise this new threat domain. This thesis proposes prey-inspired survivability to address unpredictable security challenges borne out of UUURs. While survivability is a well-addressed phenomenon in non-extinct prey animals, applying prey survivability to cloud computing directly is challenging due to contradicting end goals. How to manage evolving survivability goals and requirements under contradicting environmental conditions adds to the challenges. To address these challenges, this thesis proposes a holistic taxonomy which integrate multiple and disparate perspectives of cloud security challenges. In addition, it proposes the TRIZ (Teorija Rezbenija Izobretatelskib Zadach) to derive prey-inspired solutions through resolving contradiction. First, it develops a 3-step process to facilitate interdomain transfer of concepts from nature to cloud. Moreover, TRIZ's generic approach suggests specific solutions for cloud computing survivability. Then, the thesis presents the conceptual prey-inspired cloud computing survivability framework (Pi-CCSF), built upon TRIZ derived solutions. The framework run-time is pushed to the user-space to support evolving survivability design goals. Furthermore, a target-based decision-making technique (TBDM) is proposed to manage survivability decisions. To evaluate the prey-inspired survivability concept, Pi-CCSF simulator is developed and implemented. Evaluation results shows that escalating survivability actions improve the vitality of vulnerable and compromised virtual machines (VMs) by 5% and dramatically improve their overall survivability. Hypothesis testing conclusively supports the hypothesis that the escalation mechanisms can be applied to enhance the survivability of cloud computing systems. Numeric analysis of TBDM shows that by considering survivability preferences and attitudes (these directly impacts survivability actions), the TBDM method brings unpredictable survivability information closer to decision processes. This enables efficient execution of variable escalating survivability actions, which enables the Pi-CCSF's decision system (DS) to focus upon decisions that achieve survivability outcomes under unpredictability imposed by UUUR.

This page has been intentionally left blank

Chapter 1 Introduction

This chapter introduces the thesis and outlines the motivations to overarching themes within the research area. It presents the research hypothesis, research objectives and highlights the research contributions. Finally, it details the research methodology before concluding with a detailed outline of the thesis chronology on a per-chapter basis.

1.1 Introduction

Cloud computing has become topical due to its global adoption and use in both critical and non-critical environments. The ability to provide compute resources (infrastructures, platforms and software) as services adds to its attractive attributes. Resources are then accessible on-demand on a pay-per-use basis, adding cloud computing to the list of utilities such as electricity, gas, water, and others. Since its inception as the de facto computing platform, businesses report increased flexibility, tremendous reduction in costs, and a wide range of opportunities (Sarkar *et al.*, 2019). From the cloud consumer's (CC) perspective, on-demand access to cloud services and resources facilitates dynamic scalability according to a cloud user's computing resource requirements, arguably ensuring improved overall efficiency (Mansouri *et al.*, 2018). Moreover, with the responsibility for routine maintenance of the infrastructure left to the Cloud Service Providers (CSP), management costs are usually low and system management is less cumbersome (Lombardi & Pietro, 2011). From a CSP perspective, the ability deploy a global network of data centres enables greater flexibility and manageability of security (Shahriar *et al.*, 2017).

A global network of data centres ushers in one of cloud computing's attractive attributes; accessibility over the internet, from anywhere using any device. Moreover, it enables CSPs to co-locate cloud entities on shared infrastructure and have a broader picture of security attacks (Dorey & Leite, 2011). Cloud entities in this regard embody individuals or organisations whose interactions to perform tasks facilitate the functions of the cloud (Zott *et al.*, 2012)(Liu & Jiang, 2011). Individuals or organisations, also referred to as tenants, may exist as disjoint legal entities, but with a common share to a view to services, applications, data, and configurations. Hence, multi-tenancy, which is a core attribute of cloud computing, describes the method of sharing an instance of an application among co-located entities, by providing every entity with an isolated share of that instance (Coppolino *et al.*, 2017). However, low-cost and scalable computing resources accessible to anyone from anywhere, on a pay-per use service inadvertently benefit the hacker community.

Despite the attractive features of cloud computing, 90% of organisations using the cloud have some security concerns (PwC, 2015). Interestingly, a common view suggests that most of these security challenges existed pre-cloud resurgence, with known solutions. Arguably, what has changed is the environment and the players in which these challenges exist. According to (Ziring, 2015), security has become harder to manage due to a widening attack surface. As shown by (Coppolino *et al.*, 2017), a malicious entity can attack VMs through commonly shared resources with little detection. Besides, the growth of "cyber things" and the proliferation of a plethora of emerging technologies complicates security management as the environment becomes complex (Yannakogeorgos, Lowther & Hayden, 2013)(Taleb *et al.*, 2017).

Unsurprisingly, malware strains such as key loggers and phishing or RAM scrapping became notorious (Levy, 2018), while Heartbleed, Shellshock, and Poodle became notorious vulnerabilities in 2014 (Coty, 2014). In 2011, the number of malware attacks increased from 3 billion to 5.5 billion with about 403 unique malware types (up from 286 in 2010) and high zero-day vulnerabilities exploited per day (Razzaq *et al.*, 2014) (Shahzad & Woodhead, 2014) (Ardagna *et al.*, 2015a). In 2016 alone, there were 1 200 data breaches, over 1.1 billion identity exposures and more than 450 000 ransomware attacks (Wiles, 2018). With increasing exposure, various new and sophisticated attacks have become commonplace, giving rise to the view that cloud computing has become a global threat domain (Zafar *et al.*, 2017a).

As interconnected devices are projected to increase to 50 billion by 2020 (Eoin *et al.*, 2018)(Moreno *et al.*, 2019) and over 75 billion by 2025 (Alam, 2019), generating over 79.4ZB of Data in 2025, security breaches are anticipated to rise exponentially, rapidly evolve and increasingly become sophisticated with a wider global reach. The result is security risk that are characteristically unpredictable and difficult to detect. Uncertain, latent, unobserved, or unobservable risks (UUURs) are difficult to predict, particularly in complex and dynamic environments. Hence, managing security considering this unpredictability will increasingly become a bigger challenge for cloud computing (Albanese, Jajodia & Venkatesan, 2018).

Unpredictability describes the character of something irregular; something that cannot be predicted, and is often cited as a significant component of UUURs (Ma, Krings & Sheldon, 2009)(Ma & Krings, 2011)(Ma *et al.*, 2014). Several authors suggest that current security information and event management (SIEM) solutions cannot adequately address unknown or unpredictable behaviour patterns (Marshak & Duer, 2016)(David & Kris, 2016). Rieke et al. (2012) concurs, suggesting that it is almost impossible to predict UUURs in cloud computing environments (Rieke *et al.*, 2012). Thus, the main problem for the cloud computing model is how unpredictability can be managed to ensure that security and quality levels of service are met yet preserving its attractive attributes. Logically, this begs the question of how to efficiently manage unpredictable changes in cloud service or system states, given dynamic and unpredictable requirements.

1.1.1 Existing security approaches

With large-scale virtualisation central to the cloud computing, physical machines (PMs), networks and other cloud-based systems fall outside the traditional enterprise security perimeter (Ali, Khan & Vasilakos, 2015). PMs are therefore susceptible to UUURs owing to advanced and sophisticated threats which easily bypass detection. On the other hand, the virtual machine (VM) model which facilitates cloud computing's resource utilisation through augmenting hardware, software, storage and networks also introduces vulnerabilities (Virvilis & Gritzalis, 2013) and (Hummaida, Paton & Sakellariou, 2016). As argued by Ali et al. (2015), VM spoofing easily introduces data leakage vulnerabilities.

Moreover, current security management and control processes are inclined towards business policy and regulation. In this sense, security decisions arise out of deterministic assessment of the impact of a security incident to business objectives, and not the unpredictable probability of the incident itself (Ni *et al.*, 2002)(Cser, 2016). Security risk, for instance, is quantified according to current or forecasted network operating conditions.

The challenge of applying this traditional approach in cloud environments is that high-level failures are easily passed down to low-level components of the cloud computing ecosystem (Harknett & Stever, 2011). Consequently, mitigating security threats remains a major concern for cloud computing environments (Shahriar *et al.*, 2017)(Zhang *et al.*, 2013). Moreover, literature shows that unpredictability limits efficient threat detection (Wang *et al.*, 2014). Hence, when applied directly to address UUURs, traditional security approaches result in high false positives and false negatives (Ma *et al.*, 2014)(Albanese, Jajodia & Venkatesan, 2018). A high occurrence of false-positives or false-negatives limits the ability to implement appropriate countermeasures (Knemeyer, Zinn & Eroglu, 2009). Nguyen et al. (2016) further suggests that addressing UUURs using traditional methods is too simplistic and negates the sensitivities of complex environments (Nguyen, Kim & Park, 2016). More so, it stands to reason that the domino effect of misapplied countermeasures complicates service costing, yet cost is an important metric for secure, reliable and resilient service provision.

According to Prasad et al. (2011), resolving cloud computing security challenges requires security processes that can maintain systems stability and ensure continuous provision of services (Padhy, Patra & Satapathy, 2011). Considering cloud's pay-per-use service model, the ability to guarantee continuous provision of services within acceptable levels of service, is a critical mission. Thus, the goal for security should be to maximise the availability, resilience, reliability, .etc. Nonetheless, the critical question is how and where to apply principles such as availability, resilience, reliability, and others in cloud environments? Moreover, how can cloud systems and underlying structures be designed to achieve this critical mission?

This research proposes survivability as a critical property to achieve security in cloud computing environments. This emphasises that cloud environments are robust under the unpredictability imposed by UUURs. Hence, the role of security is to define controls that reduce susceptibility to threats and enhance repair and recovery against UUURs, and other threats affecting high-risk environments (Romine, 2019).

1.1.2 Survivability as a suitable mission for security

Survivability is the property which describes the ability of a system to continue the mission of providing services despite underlying failures to security, resiliency, reliability and dependability. It is a proactive approach and a continuous mission; a capability to which a system can timely provide services after intrusion or compromise occurs (Wang *et al.*, 2012b). As (Mehresh & Upadhyaya, 2012) notes, the 'mission' element of survivability suggests ensuring the continuity of a set of essential services, considering that precautionary countermeasures will fail. Hence, Survivability-over-Security (SOS) (Yurcik & Doss, 2002) was proposed as a descriptive focus to explicitly demonstrate how security relates to survivability.

Traditional survivability is considered around the technical and business perspectives of security, while redundancy is specified as a requirement for survivability. This notion of survivability and redundancy is suited for static environments (Prusty, Sethi & Nayak, 2016)(Singh & Srivastava, 2018)" where virtual network embedding (VNE) techniques achieve survivability based upon the predictable availability of redundant resources at the physical layer (Lipson & Fisher, 2004)(Chowdhury, Rahman & Boutaba, 2012). By simply mapping a system to a static structure or network, fault trees and reliability block diagrams can therefore be easily manage survivability (O'Connor & Kleyner, 2011)(Khan *et al.*, 2015). However, VNE techniques imply that survivability is constrained to the hardware layer, which is a challenge where resources are abstracted or limited, and resource requirements are unpredictable. Clearly, while the survivability concept is well-defined and understood in this traditional context, some drawbacks exist with its application in cloud environments.

Foremost, survivability in the cloud context is an emerging phenomenon whose character arises from interactions among multiple elements. While survivability behaviours are predictable and easy to manage where interactions are deterministic, this is not the case in cloud environments. Heterogeneity in cloud environments means that interactions are probabilistic and survivability behaviours are unpredictable, dynamic and complex. If survivability requirements frequently change, meeting survivability objectives becomes a near impossibility. As a result, current designs of survivability solutions for cloud environments are poorly understood, leaving existing survivability architectures undeployable directly in cloud computing.

As the state of a system changes due to UUURs, a survivable should restore that system's capacities, and as well, adapt to dynamic changes in order to maintain it. Equally, survivability should also aim to improve a future system's capacity to maximise survivability response measures (both passive and active tools and mechanisms). The main challenge, nonetheless, pertains to the unpredictable and unobservable evolution of the system. (Rieke *et al.*, 2012) hypothesises predictive monitoring methods to provide insights into proactive mitigations against future negative actions.

1.1.3 The need for bio-inspiration

Whereas maintaining security remains critical, survivability in view of unpredictable compromise and catastrophic failure, is a problem of significant practical interest for cloud computing. In the military context, this survivability contemplates damage tolerance and damage avoidance, i.e. vulnerability, recoverability and susceptibility, as central to decision-making (Rodríguez, Merseguer & Bernardi, 2014) (Vassalos, 2019). In nature, survivability is observable in several natural systems, as well as it is described in several biological theories and principles (Quach *et al.*, 2013). Along with resilience, robustness and adaptability, survivability is a well-addressed phenomenon in natural systems (Dressler & Akan, 2010). Species survivability, in particular, is addressed in nature's prey animals, where anti-predator and predation avoidance mechanisms and behaviours ensure survival against predation (Quach *et al.*, 2013). Nature has contributed successful solutions to problems in a range of domains, including finance (Brabazon & O'Neill, 2006) and robotics (Oates *et al.*, 2009) to name a few. The strength of natural systems reside in the ability of autonomous entities to make local decisions, continuously coordinate and share information to maintaining a global form of order (Sayed, 2014).

Given the successful application of biological systems in other areas and the critical positioning of cloud computing society in general and critical infrastructures, it seems logical to investigate how natural systems survive. Specifically, with focus upon the survival mechanisms and behaviours prey utilise against predators, and its possible application in cloud computing. However, the use of biological survival and prey's survival against predation in particular, for survivability in cloud computing is an under-researched area (Stoykov & Yazidi, 2016).

Several factors should be considered before biological systems are applied within/as security systems. According to (Somayaji, Locasto & Feyereisl, 2007), the first and most important is that biological and computer systems do not share an end-goal. For instance, whereas biological systems aim to attain global survival under dynamic environmental conditions, the goal of many computer systems is to simply accomplish individual computational tasks. Thus, assuming a biological model for survival is used in computing, the effectiveness of the model must be understood when a computing system's environments change at the global context. More so, survivability in biological systems is said to be instinctive, wherein individual sacrifice can be performed to achieve the global goal to survival (Somayaji, Locasto & Feyereisl, 2007). On the contrary, an analogous sacrifice to a computing task, i.e. downtime, is undesirable in computing systems. Logically, implementing such biological model directly will be operationally challenging since the systems under consideration have contrasting goals. Conceivably, developing bio-inspired solutions for cloud computing should consider the diversity of entities and the dynamic changes (e.g. changes to system requirement, system goals, security, survivability, etc.) such diversity brings.

1.2 Research hypotheses

Achieving effective security in cloud computing using traditional countermeasures and approaches is challenging and nearly impossible due to the unpredictability imposed by UUURs. Hence, in an ideal cloud environment, the following properties should be satisfied:

i. UUURs are proactively managed and controlled.

ii. UUURs can be autonomously managed and controlled.

iii. Ensure the survivability of cloud systems regardless of unpredictable threats.

iv. Enable survivability mechanisms to be controlled according to one's requirements.

v. Enable decision-making on survivability management and control mechanisms to be intelligent to cope with uncertain changes and unpredictable survivability requirements.

These properties are difficult to achieve due to theoretical and physical contradictions. Hence, the thesis attempts to employ bio-computational design as realised through bio-inspired approaches, to propose a method that improves cloud computing security by ensuring survivability. Research efforts should aim to understand survivability in the context of the holistic view to cloud computing and services design, and its mission as a viable complement to security.

The hypothesis under test is specifically that:

H1:

> Escalating survival behaviours and mechanisms employed by natural preys against known and unknown predators can be applied as unconventional solutions to enhance survivability in cloud computing environments. The above approximate to properties **i-iii** and satisfies properties **iv-v** above.

H2:

> Escalating survival mechanisms by natural preys, against known and unknown predators as integrated into survivability management and control system, where survivability is proactively managed and controlled within the system and user-space, to localise failure and enable user-level input at run-time, respectively.

1.3 Research aim and objectives

This thesis aims to address cloud computing's security challenges using inspiration from natural prey's survival solutions. Notwithstanding many other challenges, addressing the scientific challenges of attaining bio-inspired cloud computing survivability will be central to this thesis's contributions. Specifically, how to apply prey-inspired survivability to address problems in cloud environments and offer innovative solutions. Moreover, how to create computationally systematic, repeatable and scalable methods for the bio-inspired cloud computing survivability domain. Given the motivations and the research hypotheses introduced in section 1.1 and section 1.2, it is necessary to address the following research objectives.

1. To perform comprehensive literature review of the cloud paradigm, focusing on existing cloud security challenges, survivability, the application of bio-inspired approaches in computing and underpinning theoretical perspectives for addressing cloud computing security challenges.
2. To comprehensively investigate the classification of cloud computing security challenges and critically analyse gap areas.
3. To propose a systematic method for inter-domain transfer of concepts; from nature to cloud computing, then applying a TRIZ-based approach to derive innovative solutions. In addition, to evaluate the efficacy of derived prey-inspired solutions for cloud computing and perform a pilot simulation in NetLogo.
4. To propose and develop a prey-inspired survivability framework for cloud computing environments based upon mechanisms in prey and solutions derived using TRIZ.
5. To develop and implement a prey-inspired survivability framework simulator to evaluate the efficacy of the proposed survivability framework.
6. To perform an experimental analysis of the prey-inspired survivability solution and assess the implications of novelty against the hypothesis under test.

1.4 Contribution to knowledge

This thesis's main contribution is the prey-inspired cloud computing survivability framework (Pi-CCSF) presented in Chapter 5 and evaluated in Chapter 7. The other contributing areas of this work are in Chapter 3, Chapter 4 and Chapter 6 . Some of these works are published in book chapters, in peer-reviewed journals and conferences. The following broadly represent the main contributions of this thesis.

1. A comprehensive critical review literature of cloud computing focused on security challenges and gap analysis, survivability, theoretical perspectives towards solving cloud computing security challenges and bio-inspired approaches. Contributing publications include RP 1, RP 2, RP3 and RP5.

2. A holistic taxonomy for cloud computing security challenges. The contributing is Chapter 3 which is published in RP 2.

3. A generalisable TRIZ-based method for prey-inspired design. Contributing chapter is published in RP 3, RP 4 and RP 6.

4. A Prey-inspired cloud computing survivability framework (Pi-CCSF) and the custom-built Pi-CCSF simulator to evaluate the practical implication of escalating actions on survivability of cloud computing environments under attack. The contributing chapters are Chapter 5 and Chapter 7 .

5. A prey-inspired target-based decision-making technique for cloud computing environments to manage survivability decision-making on escalating actions in view of fuzzy survivability and unpredictability. The contributing chapter is Chapter 6 .

1.5 Research methodology

This thesis investigates survivability as a suitable property to address cloud computing security problem. Hence, prey-inspired cloud computing survivability framework (Pi-CCSF) is coined out to describe how survivability will be enabled and controlled at the control-plane using the existing cloud infrastructures. Key to Pi-CCSF is the high-level survivability management system, low-level prey-inspired survivability mechanisms and additional resources. To achieve the foregoing, this section presents the thesis's research process, and lays bare this research's underlying philosophical perspectives and the methods applied. The research onion postulated by (Saunders, 2009), which highlights some common research types, approaches, strategies and choices places the current research into context.

1.5.1 Research philosophy

Research paradigms inform a researcher's methodology and influence their data analysis strategy (Saunders, 2009) (Anderson & Ellenbogen, 2012). Hogan and Maglienti (2001)'s opinions are that epistemological perspectives prescribe both domain-specific and domain-general rules for what a researcher believes. Also, what a researcher accepts, rejects or modifies before the planning, evaluating and monitoring processes of researching (Hogan & Maglienti, 2001). Given the previous, it is conceivable that epistemological underpinnings also hugely influence a researcher's judgement on the validity and limits of literature. To this end, (Hogan & Maglienti, 2001) thus notes observations and conclusions to be independent of theories that follow a positivist philosophical perspective. Among many perspectives, post-positivism's strengths lie in its use of hypothesis developed through statistical and quantitative measures and experimental designs. In addition, this approach allows for mixed methods and the generalisation of secondary data (Stage & Manning, 2003). A post-positivist approach suits the current research as it allows both qualitative and quantitative researches. Furthermore, an empirical analysis will focus upon the quantitative measurement of complex attributes of survivability in cloud computing environments.

This thesis takes a pragmatic approach to address the hypotheses under consideration guided by the epistemic support for hypothesis testing. In this sense, the pragmatic criteria equate simulations outcomes to real cloud computing outcomes, regardless of different environments or attributes. As (Stage & Manning, 2003) suggests, pragmatism is apt for novel researches as it emphasises objective and logical inference, and measurable facts that are verifiable. Moreover, it allows for a subjective relationship between the researcher and the subject, whereupon conclusions depend upon context.

1.5.2 Research approach

Several research methods within the conceptual and experimental domains apply to different levels of research. To address this thesis's objectives, three levels of research apply.

Foremost, an interpretive level identifies through the researcher's logical interpretations of literature, theory in both cloud computing and predator-prey systems. This level also identifies concepts and relationships within the study's boundaries to understand, describe and explain the research developments. In this research, and as postulated by (Iivari, 2007) hypotheses, concepts, classifications and taxonomies are epistemological knowledge classes that precipitate in systems concepts and causation scenarios which are later validated. Hence for instance, a proposed holistic approach aims to show cloud security challenges as an integrated tool, encompassing multiple security failure points and perspectives. The holistic interpretation exposes entity relationships among cloud components and security areas, which facilitates adequate security enforcement through the correct implementation of countermeasures. Thus, this approach is aimed to achieve the objective to consolidate existing knowledge (contexts) and make interpretations. By reconceptualising existing textual knowledge into graphical formats, better visualisation will help designers to distinguish varying contexts in a non-ambiguous manner. Disambiguation of contexts is itself an important aspect of design, particularly where knowledge from multiple domains is combined (Medathati *et al.*, 2015).

A comparative research level focuses upon the "how" questions to understand large scale historical data, both subjective and objective, within the areas under consideration. As postulated by (Ragin, 2014), this level enables exploration to understand subcomponents of a topic area, exploration of similarities and differences across comparable areas, and iteration to expose contradictions. Accordingly, it stands to reason that the comparative level benefits this research by enabling the review of cloud and natural systems, and systematically mapping similarities.

Finally, the Inventive level focuses upon determining and classifying technical and technological processes by deriving information from the applied knowledge on nature (Savransky, 2000). This level employs TRIZ as a human-oriented knowledge-based systematic methodology for inventive problem solving (Savransky, 2000). For instance, the current study uses descriptive and perspective statements obtained through inductive processes to combine knowledge (effects and phenomena) from predator-prey and inform artificial technical and technological solutions for cloud computing environments. The inventive level is thus important for the analysis of design principles and modelling cloud environments. The preceding is a problem-driven approach (Burke, 2007)(Helms, Vattam & Goel, 2009) to model cloud survivability based upon models of predation-avoidance and anti-predation behaviours and mechanisms of natural preys. A descriptive is employed for its strength in analysing quantitative data (Chapman, Lawless & Boor, 2001).

1.5.3 Research design

When considering the research design, the current study aims to evaluate hypotheses and theory which are generalisable (Amaratunga *et al.*, 2002). Hence, it investigates survival in prey animals to establish the efficacy of applying prey's survival behaviours and mechanisms in cloud computing environments. As such, it would be possible to perform repeatable and objective comparative evaluations to determine the reliability and validity of the investigations against a verifiable hypothesis. Methods for analysis including taxonomies and simulation enable effective integration of research evidence around a central proposition (Amaratunga *et al.*, 2002).

Evidence in literature suggests that two design approaches are prominent in computer science research; theoretical and experimental (Wainer *et al.*, 2009)(Coiro, 2014). Conceptual and formal modelling and mathematical formalisms typify commonly applied theoretical approaches, while simulation, prototype development and results evaluation and analysis are commonly applied in experimental approaches. Conceptualisation, modelling, simulation and experimentation therefore tie in well with the positivist elements in the post-positivist paradigm underpinning the current research. Descriptions and the application of the above are briefly introduced below.

Conceptualisation: Due to the level of abstraction in the current research idea, conceptual design is deemed suitable due to the "incompleteness of initial knowledge of requirements and constraints" (Hsu & Liu, 2000) for the prey-inspired design for cloud computing survivability. For instance, it is important to conceptualise the holistic taxonomy (publication PR2), due to non-existent holistic data on computing security challenges (See Section 3.2, holistic approach; design and requirements). There is also merit for conceptual design in the application of TRIZ to derive conceptual survivability solutions for cloud computing, since TRIZ requires other pieces of conceptual implementation (See 4.2, the method and application of a TRIZ-based approach). The theory of inventive problem solving (TIPS), an English translation of the Russian acronym TRIZ (Teorija Rezbenija Izobretatelskib Zadach), is a popular design technique among engineers. TRIZ is centred on the premises of ideality; that a benefit of a system outweighs the products of cost and harm, contradiction; pertaining to the elimination of solution with harmful effects and system approach. Since TRIZ it is systematic, repeatable and based upon successful patents (Fu et al., 2014), the TRIZ method is applied to build a toolkit for prey-inspired survivability design. Thus, conceptualisation culminates into the proposed prey-inspired cloud computing survivability framework (Pi-CCSF) as a feasible integration of earlier conceptualisations (See 5.1; survivability design and the Pi-CCSF).

Modelling: While conceptual analysis is a theoretical approach to decompose a complete problem into smaller components that can be well understood, the relationships between the components define the complete problem. According to (Vessey, Ramesh & Glass, 2005), mathematical modelling can be used to show these relationships. Hence in this research, modelling is applied to describe the behaviour of VMs' response to prey-inspired mechanisms and the practical significance to the Pi-CCSF. Moreover, the modelling method is used to develop the concept of a target-based decision-making technique for cloud computing survivability proposed in Chapter 6.

Simulation and experimentation: Whereas the modelling of complete real cloud and natural prey environments is challenging due to limited domain-knowledge, time and resources, simulations allow for the recreation and analysis of the behaviour of those systems by using a model. However, the accuracy of results is an issue since the simulated model does not offer real system variables (Hsu & Liu, 2000). In the current research, Lotka and Volterra's (LV) model is used to study VM's survivability behaviours. By simulating predator-prey survival analogies using NetLogo (Wilensky, 1999), it is possible to evaluate the efficacy of the hypotheses under test and define the direction and further developments of the current research idea. NetLogo enables simulation of ecological agents in near nature conditions (Lytinen & Railsback, 2012). Hypotheses testing on simulated data (See 7.4; results and analysis) gives an indication of how the research idea may perform in a real cloud environment.

The table below shows the link between the research contributions and the design.

Table 1. Research method per thesis contribution

	Conceptual	Modelling	Simulation	Experimentation
Contribution 1	✓			
Contribution 2	✓			
Contribution 3	✓		✓	
Contribution 4	✓	✓	✓	
Contribution 5	✓	✓		✓

1.6 Thesis outline

The thesis organisation is as follows:

Chapter 1 – Introduction to the research

This chapter introduces the research area and details the motivation, the hypothesis, the research methodology, the aim and objectives of the research, and the contributions to knowledge.

Chapter 2 – Literature review

This chapter interrogates the existing literatures on cloud computing areas motivated in the introduction. Primarily, the literature review aims to establish the current state-of-the-art in domains of cloud computing security, survivability, bio-inspired systems and relevant theoretical perspectives contributing to these domains. Through a critical analysis, the findings identify open challenges for cloud computing security, within which the current research contributes.

Chapter 3 – A Holistic taxonomy of cloud computing security challenges

This chapter proposes a holistic taxonomy of cloud computing security challenges. In addition, it highlights relationships among cloud entities thereby exposing security areas for further analysis. This research's holistic notion (postulates the source and origin) facilitates the bio-inspired design of conceptual countermeasures.

Chapter 4 – A TRIZ-based method for prey-inspired cloud computing survivability design

This chapter presents a systematic method for developing solutions. A 3-step process facilitates a generalisable process for transferring concepts across domains (biology to cloud computing). A TRIZ-based process is applied to derive specific solutions for cloud survivability. Each derived analogy serves to address a specific contradiction and generates creative solutions for cloud computing, with varying levels of abstractions.

Chapter 5 – Prey-inspired cloud computing survivability framework (Pi-CCSF)

This chapter proposes the prey-inspired cloud computing survivability framework (Pi-CCSF). Pi-CCSF builds upon TRIZ-derived solutions for prey-inspired survivability. Pi-CCSF's design is placed in the engineering context to encompass security design as a component of the survivability service-oriented mission assurance (SOMS). The framework run-time is scoped in the IaaS model, to enable user-input and support other concepts such as security, reliability, fault-tolerance, etc.

Chapter 6 – Prey-inspired target-based decision-making technique (Pi-TBDM) based fuzzy cloud computing survivability requirements

Details the decision-making system (DS); a critical component of Pi-CCSF and formulates a target-based decision-making technique (TBDM) for cloud computing survivability. Considering the unpredictability imposed by UUURs, this chapter addresses the main issue of how to bring imprecise or fuzzy survivability information closer to survivability decision processes. Additionally, how to factor into the overall decision process, other contexts including the user's decision attitudes and preferences for escalating survivability actions, as may be outlined in an SLA.

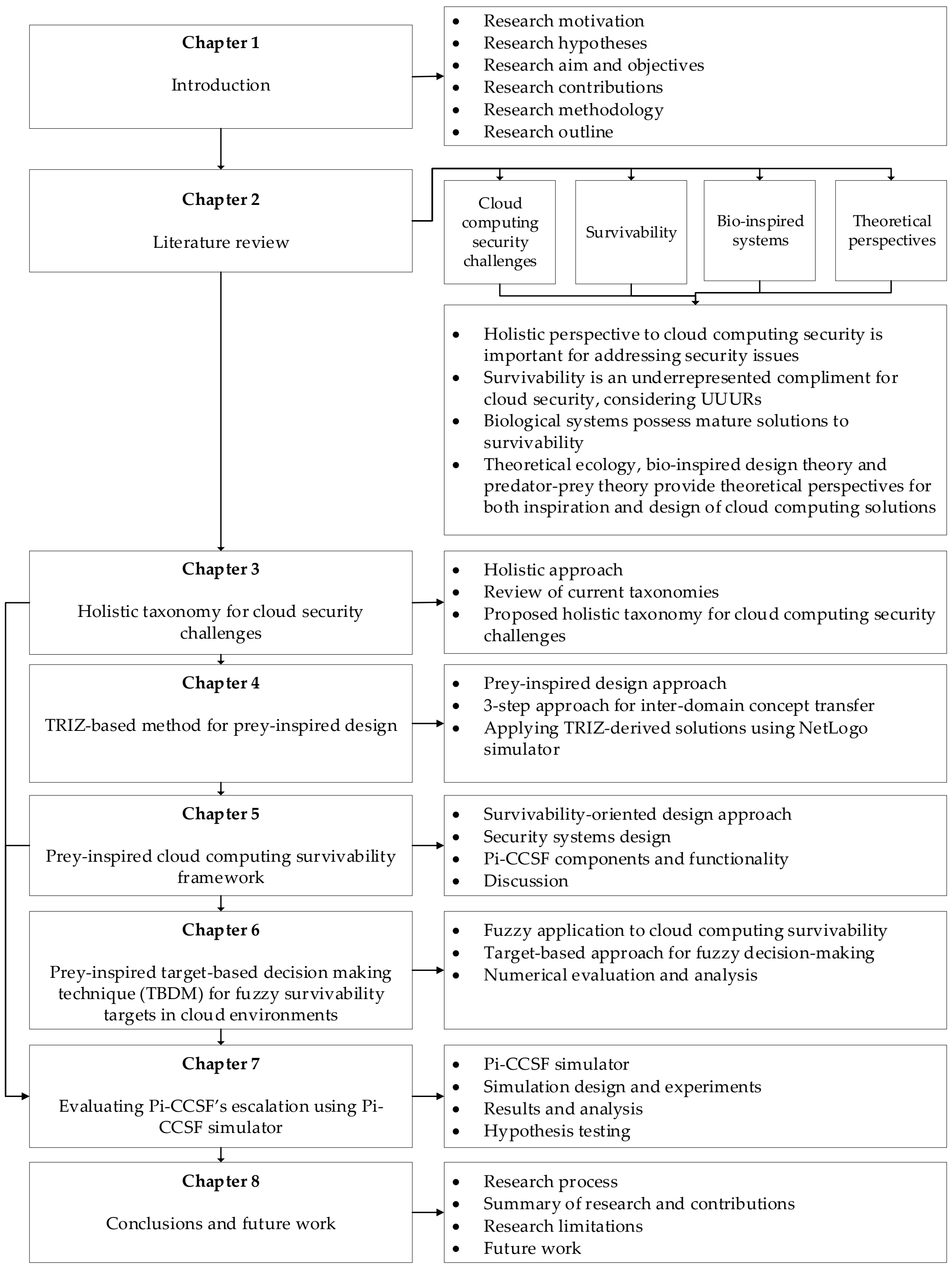
Chapter 7 – Evaluating Pi-CCSF using Pi-CCSF simulator

This chapter presents Pi-CCSF simulator, a custom-built python simulator for testing and evaluating the theoretical concepts of Pi-CCSF and their implications in the context of practical application. The purpose of simulation is to investigate the behaviour of VMs in different states of compromise when prey-inspired survivability actions are applied.

Chapter 8 – Conclusions and future work

This chapter closes the research by reflecting upon the research process and drawing key conclusions on the contributions of the research. It also considers the research's limitations and provides recommendations for future work direction.

Figure 1 illustrates the overall view to this research.



Chapter 1
Introduction
Research motivation
Research hypotheses
Research aim and objectives
Research contributions
Research methodology
Research outline
Chapter 2
Literature review
Cloud computing security challenges
Survivability
Bio-inspired systems
Theoretical perspectives
Holistic perspective to cloud computing security is important for addressing security issues
Survivability is an underrepresented compliment for cloud security, considering UUURs
Biological systems possess mature solutions to survivability
Theoretical ecology, bio-inspired design theory and predator-prey theory provide theoretical perspectives for both inspiration and design of cloud computing solutions
Chapter 3
Holistic taxonomy for cloud security challenges
Holistic approach
Review of current taxonomies
Proposed holistic taxonomy for cloud computing security challenges
Chapter 4
TRIZ-based method for prey-inspired design
Prey-inspired design approach
3-step approach for inter-domain concept transfer
Applying TRIZ-derived solutions using NetLogo simulator
Chapter 5
Prey-inspired cloud computing survivability framework
Survivability-oriented design approach
Security systems design
Pi-CCSF components and functionality
Discussion
Chapter 6
Prey-inspired target-based decision making technique (TBDM) for fuzzy survivability targets in cloud environments
Fuzzy application to cloud computing survivability
Target-based approach for fuzzy decision-making
Numerical evaluation and analysis
Chapter 7
Evaluating Pi-CCSF's escalation using Pi-CCSF simulator
Pi-CCSF simulator
Simulation design and experiments
Results and analysis
Hypothesis testing
Chapter 8
Conclusions and future work
Research process
Summary of research and contributions
Research limitations
Future work

Figure 1. Overview of Thesis structure

Chapter 2 Literature review

This chapter reviews the literatures relevant to this thesis's motivations and the contributions laid out in Chapter 1 . The literature review aims to establish the existing scholarship relevant to this study and identify gap areas and open questions that place the current research into context. Thus, it provides background information, the context and the motivations outlined in Chapter 1, and discusses in each case, the extent to which existing literature addresses these areas.

2.1 Introduction

The systematic literature review (SLR) method is distinguished from a narrative literature review based upon primary evidence retrieved using a clearly defined inclusion criteria (MacDonald, 2003). In addition, a SLR produces a rigorous summary of existing literature, as well as valid and comprehensive evaluation and interpretation of available relevant literature (Keele, 2007). The current study will implement a mixed approach with a clear aim to obtain objective findings derived from a tightly focused research. A narrative bias will be useful for reviewing historically broad topic areas while enhancing the transferability of research findings (Malterud, 2001). Correspondingly, the systematic bias will ensure a comprehensive research that yields transparent and valid conclusions (Keele, 2007). The remainder of this chapter is structured as follows:

Section 2.2 provides a review of cloud computing security challenges and reports on key findings. Section 2.3 presents a review of the research's survivability context and reports on key findings. Section 2.4 reviews theoretical perspectives that provide underpinning basis towards meeting the contributions of this research. Section 2.5 investigates bio-inspired systems and their use in the computing domain. The predator-prey system is investigated, and the key findings summarised. Finally, Section 2.6 provides a summary of the chapter.

2.2 Cloud security challenges

With the advent and wide use of cloud computing, security is arguably more complex, requiring a variety of processes to maintain data and systems stability (Prasad *et al.*, 2011). According to (Zissis & Lekkas, 2012), the cloud computing security perimeter is wider and complex and thus, it is easy to bypass using sophisticated attacks. Along this line, (Cybenko *et al.*, 2014) notes that zero-day attacks are therefore able to plan their attacks and persist within the compromised networks systematically. Logically, this suggests that cloud computing environments enable adversaries to increase their attack surface, which complicates vulnerability management and elevates the attack complexity. Cases in point include Stuxnet, Flame, and Duqu, which obfuscated network traffic to evade detection (Virvilis & Gritzalis, 2013). Beyond this, emerging technologies such as the Internet of Things (IoT) and Big Data complicate traditional firewall deployment due to the challenges of enforcing static security policies over highly mobile environment. Figure 2 illustrates the foregoing, outlining how broad network access enlarges the attack surface, which enables sophisticated threats to easily bypass the traditional security perimeter. Addressing these security problem demands resolving how cloud security management tools, systems, software and platforms are designed and operated in this new environment.

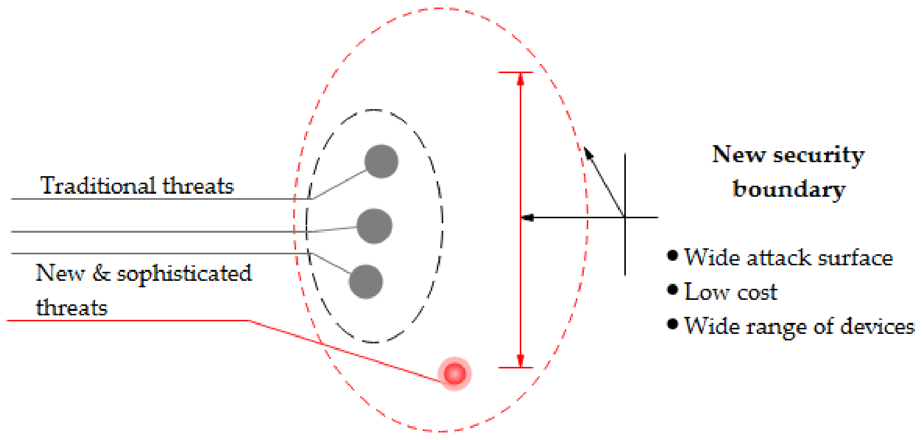


Figure 2. Graphical representation of the cloud security view produced by the author of this thesis

Summarily, the following are the main challenges.

- Traditional approaches i.e. reactive and preventative methods that rely on an identified intrusion signature cannot adapt to new sophisticated threats.
- Cloud environments are complex and unpredictable, yet current security solutions such as intrusion detection and prevention systems, firewalls and antiviruses lack the complexity to cope.
- Cloud computing environments exist outside the traditional security boundary, which limits the extent to which traditional controls can be effective. Security controls and approaches are the recommended set of actions that provide specific and actionable ways to stop attacks (SANS, 2016). These are detailed in Appendix A.

Several works have questioned the effectiveness of traditional security techniques such as intrusion detection systems (IDS) in cloud environments (Cser, 2016)(Albanese, Jajodia & Venkatesan, 2018). For instance, security concerns in cloud environments are increased by circumventing security systems or exploiting vulnerabilities of APIs in cloud software vendors (Ahamed, Shahrestani & Ginige, 2013). Consequently, insecure APIs pose a greater risk by providing execution privileges to unauthorised users (Henning, 2007). According to (Hashizume, Yoshioka & Fernandez, 2013), vulnerabilities are elevated due to the physical and logical structure of the cloud.

Virtualisation is a method to achieve logical abstraction of compute resources; computation, network, storage, operating system, and so on, from their physical constraints (Liu *et al.*, 2015). Due to this logical abstraction, cloud computing resources can dynamically expand vertically up or down, and horizontally according to requirements. Abstraction encapsulates a software layer, e.g. a virtual machine monitor around an operating system, to offer similar interactions and behaviours as from a physical system (Sahoo, Mohapatra & Lath, 2010); (Pearce, Zeadally & Hunt, 2013). Since abstraction enables VM independence from the physical device and runs multiple virtual machines from the same physical hardware, VM introspection; a system-level technique to monitor the state of VM at run-time, introduces reverse engineering challenges (Chen, Paxson & Katz, 2010).

According to (Baars & Spruit, 2012), at least sixty potential security domains exist within the cloud architecture. The impact of security concerns in cloud computing is reported by Verizon data breaches (Verizon Business, 2011; Solutions, 2015; Verizon, 2016). VM vulnerabilities to side-channel attacks are among some of the highly ranked security issues (Cloud Security Alliance, 2013a);(Cloud Security Alliance, 2013b). In fact, (Aviram & Ford, n.d.); (Xu *et al.*, 2011) argue that it is easy for malicious users to observe or send data through exposed side-channels passively. By monitoring cache usage, adversaries can easily identify a target VM (Ristenpart *et al.*, 2009) while behaviour analysis aids cross-VM information leakage (Ahamed, Shahrestani & Ginige, 2013).

For brevity's sake, Table 2 summarises commonly discussed cloud security issues. Security challenges are described according to a corresponding general issue(s) and a specific security issue(s) (bullet points). While some of these challenges are discussed, others are added to supplement existing literature for the interested reader.

Table 2. A summary of commonly discussed security challenges

Challenge	**Issue**
Control	Outsourcing, PaaS, SaaS, and IaaS. Ease of use by end user's degree of information security & control, i.e. control of data & control over the security • The data owner has no full physical control over their data • No control over OSs, network & servers, storage & applications in SaaS • No control over Oss, network & servers in PaaS of control over networking components in IaaS
Trust	Securely establishing trust between servers & clients & trusting cloud environments. • Trust between servers & the client's misuse of cloud services • Impose security policies
Virtualisation	Updating security countermeasures is paramount to preventing data breaches & leaks • The risk to the integrity of saved VM images • Malicious insider • Risks to confidential data stored in virtual machines
Malware	Intrusive and hostile software • Sophisticated malware such as Stuxnet & Flame. • Zero-day exploits
Attacks on Web Services	Prominent attacks & immature coding exploit online vulnerabilities • SQL injection flaws & cross-site scripting • Signature wrapping attacks • Malware, CSS, and Denial of Service (DoS)
Denial of Service	Compromise the availability of services • Semantic & flooding DDoS attacks

	• FRC attacks falsely use cloud resources: impacts application-layer servers • Exploits are financially detrimental to a cloud consumer • Cause the Operating System (OS) kernel to crash
Weak Identity, Credential & Access Management	Insufficient scalability in identity access management systems • Spoofing attacks, DoS attacks, Elevations of privileges and Repudiation
Data breaches	Unauthorised access/use of confidential/sensitive data. • Vulnerabilities in applications, Malicious insider and Information disclosure
Data loss	Insecure & unnoticed configurations or vulnerabilities result in potential exploits & data loss. • Side-channel attacks expose IaaS, PaaS and SaaS to breaches
Insecure interfaces and APIs	Poorly designed APIs in cloud software vendors • Provide execution privileges to unauthorised users
Account & service traffic hijacking	The exploitation of software weaknesses and personal information • Phishing attack • Service or account hijacking
Malicious insider	Breach of confidentiality by manipulating multi-tenancy • Co-residency attacks • Malicious insider
Abuse of cloud resources	Where IaaS providers lose some control • Hackers & spammers take advantage of free limited trials
Insufficient due diligence	Choosing and moving functions to cloud environments entails careful consideration
Shared technology vulnerabilities	Vulnerabilities due to intrinsic & prevalent core cloud computing technologies
Privileged user access	Where 3rd parties process data outside the enterprise • Outsourcing bypasses means that an enterprise's security controls (physical or logical).
Data location	A cloud vendor provides location of data & processes, and provides routine maintenance, • No information to the cloud consumer about the location of data, as well as the processes performed
Regulatory compliance	Challenges when dealing with LSAs & other process & regulatory issues • Assurance • Process and regulatory issues
Data segregation	Logical storage of data in multi-tenant environments • Cross channel attacks • Malicious insider
Recovery	Low-cost disaster recovery & data storage solutions
Long term viability	Concerns a cloud consumer's data if a CSP loses their business

2.2.1 Survey of countermeasures

There are overwhelming efforts, both in academic and industry, dedicated to addressing security challenges in cloud computing environments. Much of these are summarised in Table 2 above. Recent notable work is the collation of user-data security and countermeasures survey by (Basu *et al.*, 2018). This section aims to interrogate some of these works.

Much of the existing literature shows a growing trend towards hybrid countermeasures to enhance visibility in industry security solutions; integrating deceptive controls, intrusion detection systems (IDS) and security information and event management (SIEM). However, there are far less proactive solutions, seemingly due to the threat of legal liabilities associated with their use. According to McGee et al. (2013), possible legal liabilities as one of the factors limiting the adoption of purely proactive solutions. Proactive approaches ensure that security systems are continuously operational and aware of changes in their environment throughout their operation (Djenouri, Khelladi & Badache, 2005). On the other hand, reactive elements in hybrid solutions (semi-preventative control element within intrusion detection systems), means that existing hybrid solutions remain susceptible to obfuscation techniques (Virvilis & Gritzalis, 2013). This is particularly important considering that in both industry and academia, hybrid solutions are currently dominant. Software patching and static perimeter security which are argued to motivate hackers, remain prevalent but ineffective in cloud computing environments (Subashini & Kavitha, 2011). An addendum (See Appendix A) provides in-depth discussions and definitions of hybrid, proactive and reactive approaches and security controls.

Unlike academia, a downside of industry solutions is a general lack of technical design detail in the public domain. Where is it available, details are mostly general rather than specific, presumably to protect the competitive advantage. Clearly, the impact of security upon business investment seems to be a driver to how countermeasures in industry are designed. As (Rong et al. (2013a) suggests, security is considered as a balance between business viability and competitive edge. Since reactive security approaches, including antivirus, monitoring and detection tools remain dominant in industry, they limited in cloud environments due to inherent deficiencies (Subashini & Kavitha, 2011). Besides these functional deficiencies, commercial solutions are often expensive and complex and currently reported to as inefficient for the cloud. As argued by (Virvilis & Gritzalis, 2013), malicious intruders can by-pass end-point security protection. In evaluating academic and industry literature, it is evident that security in the cloud is largely handled through legislation, contracts and good practice (fulfilled through SLAs).

Traditional security approaches remain prevalent in industry across most security vendors. However, there is a visible shift towards software solutions, as opposed to hardware, to adapt in the cloud environment, for instance, Amazon's concept (Huang *et al.*, 2015). As noted in a report by Tara (2018), a survey of cyber security professionals overwhelmingly identify outdated security software as an obstacle to cloud security(Tara, 2018). This research suggests the following to be widespread in existing solutions in academia. Foremost, most solutions in academia and industry fall within the hybrid category. As well as in industry, there is limited research and industry implementing of purely proactive approaches. Most solutions in academia focus on developing single solutions, with most countermeasures being effective upon the specific the specific security solution they are developed. A case is point is works against DoS attacks (Lombardi & Pietro, 2011).

Unlike commercial security solutions, solutions in academia carry a lower financial burden. In addition, most solutions proposed in academia focus upon the technical design of the solution, rather than its security strategy. As such, the current researcher posits an argument that solutions in academia lack adequate on strategy and remain largely suitable to the perspective to which security challenge is viewed. Based on the above, it is logical to conclude that most solutions in academia are only applicable to the perspective in which a security incident is viewed, (e.g. the end-user perspective), as opposed to the global cloud security landscape.

Clearly, open challenges in addressing cloud security concerns remain from both industry and academia due to the unique nature of the cloud environment. As noted earlier, with broad network access, the attack surface is enlarged allowing adversaries to intrude a network with no detection. As a possible solution, the "Kill Chain" approach (Hutchins, Cloppert & Amin, 2011) offers proactive and dynamic intelligence-gathering capabilities to enhance continuous security posture awareness. Furthermore, other proactive approaches involving synchronised and real-time discovery, analysis and mitigation are areas for further investigation. Moreover, proactive capabilities can be switched ON and OFF depending upon security requirements and hyper segregation to enhance resistance to attack. Hence, a pertinent question which arises is how to improve intelligence gathering, an alerting mechanism, and a method to anticipate an attack and enable swift response, as opposed to relying on patching strategies? After the above, it is also pertinent to question how incident response time can be improved while also improving cloud computing platforms' resistance to attacks. More, how to proactively detect, identify and stop an adversary before an exploit.

Table 3 summarises security countermeasures in academia whereas Table 4 is a summary of countermeasure in some of the industry's leading vendors according to the criteria set out by (Cser, 2016). For the benefit of the interested reader, a detailed survey and discussions of cloud security countermeasures are added as addendum of this thesis (See Appendix B).

Table 3. Summary of countermeasures in academia

Countermeasure	**Security Control**
A strategic and systematic approach(Park & Ruighaver, 2008)	SP
Trusted Third Party (TTP) (Zissis & Lekkas, 2012)	AD, SP
Cooperative Intrusion Detection System Framework(Lo, Huang & Ku, 2010)	SP, CO
A dynamic and localised security model (Subashini & Kavitha, 2011)	AD, SP
Policies, hardware, and software security view (Mathisen, 2011)	SP, CR
Cloud Security Countermeasures (Jamil & Zaki, 2011)	SP, DT, CR
Self-monitoring Defensive Mechanism (Mazur *et al.*, 2011)	DT, PE, AD
Security SLA Management for Cloud (Bernsmed *et al.*, 2011)	SP, CR, AD
Advanced Cloud Protection System (ACPS) (Lombardi & Pietro, 2011)	DT, SP
Trust-based secure interoperation framework (Mell & Grance, 2011)	SP, CR, PR, AD
Safety Measures for cloud computing (Wang & Mu, 2011)	SP CR, DT
Isolation solution (Behl & Behl, 2012)	PE
Client Trace Back Model (CTB) (Joshi & Joshi, 2012)	SP, DT
Mitigation Strategies (Chow *et al.*, 2009a)	SP, PR
Incident-Based Solution (Ryan, 2013)	SP
Security 360^0 (Rao, J.R., Chari, S.N., Pendarakis, D., Sailer, R., Stoecklin, M.P., Teiken, W. and Wespi, 2016)	PE, CR, DT, & SP,

Key: DT – detective, PE – pre-emptive; PR – prescriptive; SP – semi-preventative; CR – corrective; AD – adaptive; DC – deceptive; DTR – deterrent

Table 4. Summary of countermeasures in industry

Countermeasures	**Security control**
CloudLock: Cisco (CloudLock, 2015a)	DT, PE, PR
Microsoft Cloud App Security Microsoft (Microsoft Mobility Management, 2016)	DT, PR, SP
Cloud Data Life-Cycle Protection (Bluecoat, 2015a)	DT, SP

BitGlass (Bitglass, 2014a)	DT, SP, DTR
Cypher Cloud Trust Platform (CipherCloud, 2016)	DC, SP, DT, CR
IBM's new security paradigm IBM (Gulla, 2011a)	PE, DT, DC
IBM Global Services and IBM research Labs' Parity	DT, CR
Proventia Management SiteProtector & Desktop Endpoint Security	SP, DT
IBM and North Carolina State University's Nuva	SP
Tivoli®NetNiew® Systems Director	SP
Data Centre Security Solutions (Over, 2014; Server, 2015)	SP, DT
Symantec ™ Protection Engine	DT, AD
Identity and Access Management as a Service. (CA Technologies, 2014)	SP, AD
Cloud Security Platform (Brief, 2015)	DT, SP
Multi-Vector Virtual Execution (MVX), FireEye AX series (Security *et al.*, 2018)	SP, DT, CR, PR
NetScaler Firewall (Citrix, 2015)	AD, CR, PR, SP
Fusion	CR, DT, & SP,
Active Defence Harbinger Distribution (Active, Systems & Networks, n.d.)	DC, CR, DT, & SP,

Key: DT – detective, PE – pre-emptive; PR – prescriptive; SP – semi-preventative; CR – corrective; AD – adaptive; DC – deceptive; DTR – deterrent

2.3 Survivability context

Survivability research advanced in the computing domain due to the development of, and need to protect critical infrastructures (Shi *et al.*, 2008)(Chang *et al.*, 2018). Such are telecommunications networks, power grids, etc. (Habib *et al.*, 2013). When considering critical infrastructures, the notion of survivability aligns with the ability to ensure timely delivery of critical services when faced with planned or unplanned faults or failures, and deliberate or accidental attacks. Four survivability themes are identified in the literature; resistance (ability to repel an attack), detection (ability to recognise an intrusion), recovery (capacity to resume complete and essential services after fault, failure or attack), and adaptation (capacity to evolve to cope with new attacks, faults or failures). Since the inception of cloud computing, some work has been done on survivability with a subset focusing upon survivability evaluation, including quantitative and model evaluation techniques and disaster tolerant architectures. This section presents some of these efforts.

Survivability is viewed variably among different communities. From a Software architecture perspective, survivability is viewed as a system quality that is related to other system qualities such as dependability, availability, reliability, fault-tolerance, and trustworthiness (Pokharel, Lee & Park, 2010). Traditionally, survivability is implied around the business objectives and risk management strategies of organisations (Lipson & Fisher, 1999). In this sense, the mission to survive ensures that the business-critical part of the system continues functioning and continues to provide essential services even in its degraded mode (Serageldin, Krings & Abdel-Rahim, 2013). Hence, the survivability mission requires intimate knowledge of the business mission it is protecting (Lipson & Fisher, 1999). Alternatively, a system can continue functioning despite adversity. The goal, therefore, is to maximise availability, maintainability, and reliability of that system (Tahvildari, 2009).

Survivable virtual network embedding techniques are in use, primarily aimed at enabling survivability as the physical layer by increasing redundant resources (Khan *et al.*, 2015). However, besides this notion of survivability as constrained to the hardware layer, this is challenging in circumstances where resources are limited, or resource requirements are unpredictable. The challenge from a holistic perspective of the cloud environment is the question of "is the system going to continue providing services considering the unpredictable loss of availability, durability, reliability, etc.?". The foregoing brings to the fore how much predictable an environment can be, foremost to address the said losses, as well as to assure the continual provision of services. In terms of assurance, several pieces of research in this area have attempted to address survivability assurance methods in one of two ways, each with its strengths and drawbacks. Figure 5 shows the survivability concept under this consideration. An epoch describes a period within a fixed context (Henderson, 2006).

Thus, survivability requirements in epoch 1 are different from the requirements in epoch 2, epoch 3, up to epoch n. Since each epoch has characteristically unique constrains, the state informs design concepts and attributes. In addition, technologies available to it (Richards & Ross, 2009), the timing of survivability decisions in cloud environments of the system. Two threshold types shown in Figure 5 are important; escalation threshold and survivability threshold and characterise the survivability concept under this consideration. The escalation threshold considers the vitality (vulnerability) of a service or system to determine how significant a UUUR is deemed as a threat to survivability and the escalating actions that follow.

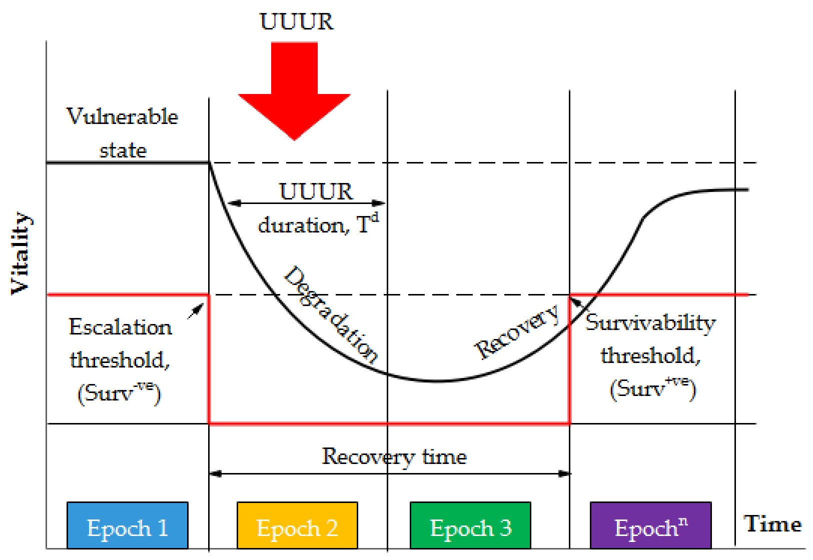


Figure 3. Representation of the survivability concept adapted from (Henderson, 2006)

On the other hand, the survivability threshold considers both the recovery time and duration of a UUUR to implement a decision that must be taken upon ensuing escalating actions. Both thresholds are important to determine the state of the system or service, i.e. is the vulnerability improving or degrading and the actions, tools or procedures relevant to the UUUR event. Now, consider cloud computing's pay-per-use model where CSPs bill CCs according to the length of time they occupy a VM, or how long its provisioned. Suppose a consumer is billed *kNt* based on an amount K for each provisioned VM in as many, N VMs, over a period, t. If an unpredictable threat that can avoid detection and persist in a tenant's environment, this will prolong tenant jobs and increase the cost (Virvilis & Gritzalis, 2013). Thus, the challenge in this sense pertains not least, to how a CSP manages such risks to avoid unnecessary financial burden on a consumer, as well, how the CSP and CC can plan and proactively anticipate such unpredictability.

Yurcik and Doss (2002) coined the term survivability-over-security (SOS) to describe the survivability goals of simultaneously reducing sum vulnerabilities while increasing recovery and flexibility in networked systems. The author of this thesis concurs with (Yurcik & Doss, 2002) and opinions that the survivability of individual system components is vital to the overall survivability of an entire networked system. A case in point in cloud environments pertains to cloud computing storage, in which storage components such as storage isolation, data recovery and storage place, are determinants of long-term data survivability (Liu, 2012).

In their work, Yu et al. (2010) focus on evaluating survivability from the cost-architecture perspective considering Byzantine Fault Tolerance (BFT). Briefly, BFT generally describes a system's ability to continue operating despite some of its failure to act, or malicious actions from among its components. Thus, in their work, Yu et al. (2010) use three different virtual machine-based architectures to investigate how different architectures impact on the survivability of a system (Yu *et al.*, 2010). Three key components are evaluated (static analysis) across each architecture; survivability based on availability, survivability under sustained attack, and the cost of each architecture. Regardless of the BFT's higher costs, a replicated architecture with BFT protocol and diversification is better than replication with isolation (Yu *et al.*, 2010). There are two clear challenges to this work. Foremost, the concept of diversification is loosely defined; does this imply diverse protocols or diverse environments? Assuming both or either, diversification in itself has been shown to increase the vulnerability factor particularly where heterogeneous systems have diverse asset value (G & S, 2013). In addition, static analysis is a major concern due to its susceptibility to false positives, which is exacerbated by the dynamic nature of dynamic attacks (Petukhov & Kozlov, 2008).

In their work Li et al. (2012) quantitatively analyse cloud security risks based on virtual machine vulnerabilities and VM placement schemes. According to these authors, an attacker exploits Type I and Type II vulnerability; exploiting hypervisor vulnerabilities in order to compromise the physical server and direct attack on VMs on the same physical server, respectively. Thus, run Markov Chain analysis over an attack dependency graph to obtain the possibility for each VM to being attacked. Based on a discreet-time Markov Chain (DTMC) analysis, a VM placement algorithm thus defines and distinguishes high-risk VM, i.e. compromised (Li *et al.*, 2012). According to the authors, the proposed VM placement algorithm will consider both preceding placement plan and new placement plan to find an uncompromised node for placement, otherwise, if a high-risk node is the only one available, placement will not occur. Their experimental results yield better survivability and reduced number of compromised VMs. The main drawback in this work is the lack of clarity around the time element of the survivability mission, i.e. what are the implications to survivability if placement does not occur because there is no low-risk node? As many researchers including (Mead *et al.*, 2000)(Lipson & Fisher, 2004)(Bigham, 2010)(Rodríguez, Merseguer & Bernardi, 2014)(Chang *et al.*, 2018) allude, survivability is about mission fulfilment. In addition, (Li *et al.*, 2012) does not address unknown attack paths or graphs, a key component of the notion of unpredictability which central to the current research.

Over the years, several authors including have attempted to address survivability through virtual network embedding (VNE) techniques. VNE is a technique that are aimed to efficiently map virtual resources onto physical network resources to provide among other things, redundancy and survivability. For instance, Xu et al. (2012) propose a Survivable Virtual Infrastructure (SVI) for a service or a tenant using multiple correlated VMs (Xu *et al.*, 2012). Similarly, Rahman and Boutaba (2013)(Xu *et al.*, 2012) propose a survivable virtual network embedding (SVNE) solution by developing a proactive and a hybrid policy heuristic based on a fast re-routing strategy and a pre-reserved quota for backup on each physical link (Rahman & Boutaba, 2013). Also, Liao et al. (2014) propose an efficient algorithm as a solution for survivable multicast service oriented virtual network mapping (SMVNM) (Liao *et al.*, 2014). Furthermore, Buyya et al. (2014) propose an architecture for QoS-aware bandwidth allocation and bandwidth-aware, energy-efficient VM placement for software-defined clouds (SDCs) on data centres (Buyya *et al.*, 2014). Gu et al. (2015) proposed a failure region-disjoint VN mapping scheme to improve VN mapping survivability taking into account mapping costs and load balancing concerns to help improve resource efficiency (Gu *et al.*, 2015). Couto et al. (2016) analysed the benefits of different data centre topologies; Fat-tree, BCube, and DCell, taking into account reliability and survivability requirements (Couto *et al.*, 2016). Along similar lines, Lo and Liao (2016) study the survivable virtual data centre allocation problem (SVAP), which aims at allocating survivable virtual data centre (SVDC) to each tenant to guarantee resource demands even after failures (Lo & Liao, 2017).

In other works, researchers have developed decentralised algorithms for mapping virtual links to virtual nodes across vast geographical locations (Houidi *et al.*, 2011). While these solutions have proved to be successful, two main challenges limit their use in the current context. Foremost, the heuristics suggested above rely on the assumption that the substrate network is always operational, which is an operational impossibility. In addition, the mapping techniques assume prior knowledge of node requirements and link capacity. This is a near impossibility when considering UUURs. Figure 4 is an illustration of the foregoing, in which different cloud networks are mapped to different data centres. Due to limitations of VNE and the limited capacity challenges of nodes within VNE's substrate network, other works have proposed VNE heuristics to leverage the embedding phases (Lischka & Karl, 2009)(Mosharaf *et al.*, 2009)(Chowdhury, Rahman & Boutaba, 2012).

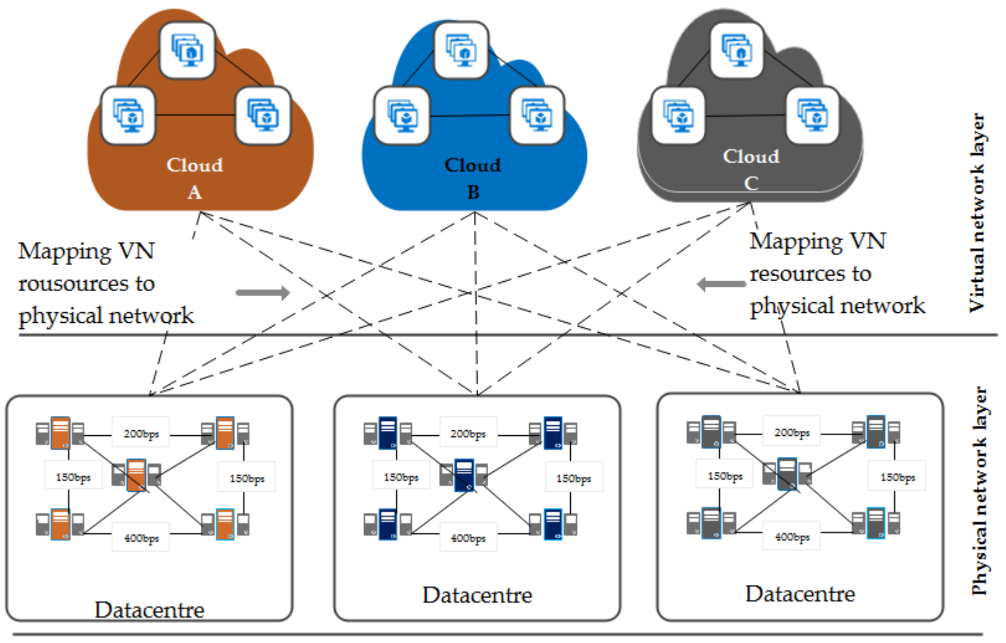


Figure 4. A VNE concept produced by the author of this thesis.

Liu and Wang (2012) proposed a system work model for maintaining and improving system survivability in cloud computing (Liu & Wang, 2012). Based on analysing security strategies for cloud computing, these authors propose the following as key components of the survivability model: Self-monitoring of the cloud state and security threats, service judge evaluation of cloud services by the cloud customer, notify attack which assesses compromised services and the recovery component of proxy service redirection and substitution (Liu & Wang, 2012). However, this work provides basic detail, giving no useful and specific detail of the model components or empirical analysis of survivability in question.

Xu et al. (2013) proposed a VM placement (VMP) and virtual link placement (VLP) techniques to address the problem of how to map survivable virtual infrastructure (SVI) to a data centre network with minimum operational costs while satisfying each VM's resource requirements and bandwidth demands between VMs before and after failures (Xu *et al.*, 2013). According to these authors, simulation results based on the real VM workload show that their algorithms perform significantly and increase survivability. The challenge with this approach is that SVI is modelled as a virtual graph, which the context of this research, subsumes knowledge of bandwidth and other requirements under any circumstance. This has been proven to be an operational impossibility when considering UUURs. Assuming their algorithm is efficient, cascading failures increase network resource cost (Yu *et al.*, 2011), a known survivability metric.

As argued by Afrin and Yodo (2019) efficient and robust recovery must aim to minimise constraints including time and cost (Afrin & Yodo, 2019). Most recently, Aibin et al. (2017) focus on addressing the resource routing problem in cloud computing. These authors proposed a software-defined adaptive survivability approach achieves the best trade-off between the efficiency of path protection and cost of routing (Aibin, Walkowiak & Sen, 2017). According to these authors, results from performance evaluation and assessment of their proposed approach show in significant improvement of network performance (Aibin, Walkowiak & Sen, 2017).

Existing works have studied virtual cluster backup provisioning with fixed primary embeddings but have not considered the impact of primary embeddings on backup resource consumption. To address this issue, (Yu *et al.*, 2017) study how to embed virtual clusters survive in the cloud data centre, by jointly optimising primary and backup embeddings of the virtual clusters. They formally define the survivable virtual cluster embedding problem and propose a novel algorithm to compute the most resource-efficient embedding considering a tenant request. The authors further propose a faster heuristic algorithm able to achieve high performance (Yu *et al.*, 2017).

Several researches attempt to address survivability assurance methods in one of two ways, each with its strengths and drawbacks. First is the qualitative method which evaluates survivability around a recovery time objective (RTO) or recovery point objective (RPO). RTO and RPO as shown in Figure 5, in this sense define the maximum permissible time for an outage and the maximum time within an outage where data may be lost (Sterbenz *et al.*, 2010a). While these methods are commonly used in the traditional context, their main drawback for cloud computing is the assumption that infrastructures (or services) are periodically provisioned according to a 'start' and 'finish', the duration of which an infrastructure (or service) is assumed as functionally healthy. Where cloud environments handle business-critical or time-sensitive data, it is anticipated that the RPO and RTO should always be kept close to zero to enhance survivability.

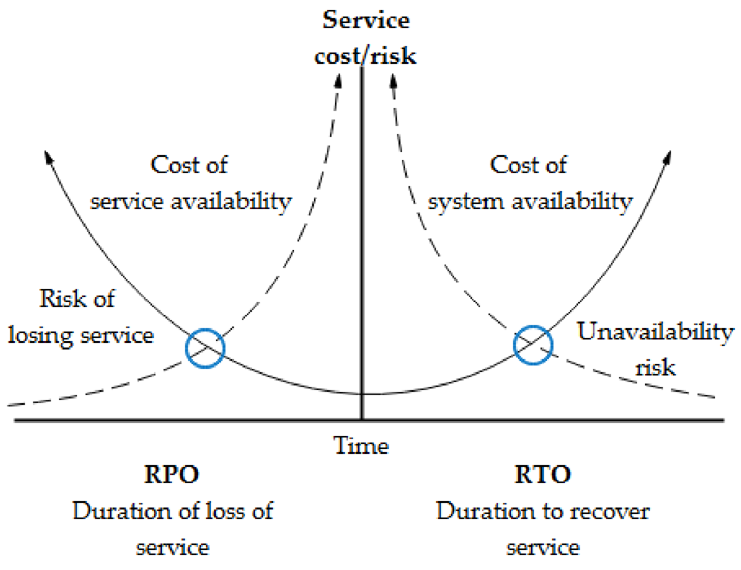

Service cost/risk
Cost of service availability
Cost of system availability
Risk of losing service
Unavailability risk
Time
RPO
Duration of loss of service
RTO
Duration to recover service

Figure 5. An illustration of the RPO and RTO concept produced by the author of this thesis

Considering the notion of UUURs motivated earlier, RTO and RPO are challenging to implement in practice as recovery time is implicitly a random quantity and unpredictable due to the uncertainty of UUURs. Moreover, the focus on maximum and minimum recovery time mean that these approaches other survivability states of a system whose values may be crisp may take an interval form or completely fuzzy. Hence, due to the above, RTO and RPO fail to adequately inform low-level implementations that match the dynamics of survivability under unpredictable and uncertain changes. Along similar lines, Richards et al. identify survivability simply as passive or active abilities to maintain value delivery despite disturbance or the ability to changes the environment through adaptable changes, respectively (Richards *et al.*, 2007). The table below summarises the authors' passive-active survivability notion.

Table 5. Passive vs Active Survivability, according to (Richards *et al.*, 2007).

	Passive	**Active**
Philosophy	Something a system has	Something a system does
Characteristic	Proactive, resistance, robust	Adaptive, flexible, reactive
Design principles	Hardness, stealth, redundancy, diversity	Regenerate, evolve, relocate, retaliate
Forecast	Presupposes state of environment	Presupposes uncertainty in future prediction
Architecture	Closed (static)	Open (dynamic)
Design focus	Resist disturbance, system-level defense	Avoidance and deterrence, architectural

Failures	Linear	Non-linear
Discipline	Reliability, risk analysis, domain-specific technologies	Process design, domain-specific technology, real options, organisational theory

2.3.1 Analysis

Quantitative approaches studied in recent years (Sterbenz *et al.*, 2010a)(Munaretto *et al.*, 2011)(Eiger, Luss & Shallcross, 2011)(Eiger, Luss & Shallcross, 2012)(Panigrahi, 2013) focus on survivability in the context of "timely recovery" for specific contexts. In their work, Trivedi and Xia (2010) specifically focus on the general method of quantifying survivability and present a framework for such evaluation. While these methods capture more details compared to the traditional RTO and RPO, they do not incorporate the cascading effect of threats such as side-channel attacks (Caron *et al.*, 2013). It is conceivable that managing unpredictable changes to cloud systems and survivability requirements, requires effective feedback control and escalation of survivability actions as the system evolves.

A critical challenge for cloud computing is how to meet survivability outcomes while keeping the operational cost of attaining such survivability low. In multi-tenant environments, the degree of tenant concurrency, resources heterogeneity and UUURs render survivability guarantees a major challenge (Floratou, Potti & Patel, 2014). Whereas traditional computing systems with predictable system evolution and typically predictable survivability requirements (Chowdhury, Rahman & Boutaba, 2012), cloud computing attributes (the ability to scale resources for instance) enable computing systems' evolution to deviate from what is typically predictable. When viewed in relation to new and sophisticated threats, the unpredictability of cloud system states becomes the central facets of cloud computing security**.**

Figure 6 is an illustration of a system's evolution from its input (icon at the apex of the graphic with unshaded black arrow) and output state (black icon at the base of the graphic with shaded black arrow). State transitions shows how predictable (blue) and unpredictable (red) intermediary system states and requirements affect the final state (system output) of the system. In a traditional sense (represented on the right side of Figure 6), from input to output, the state of the system and the system requirements are always predictable and therefore can typically be estimated. However, an unpredictable change or phenomena such as a UUUR event induces an unpredictable state which further induces unpredictable requirements to the system.

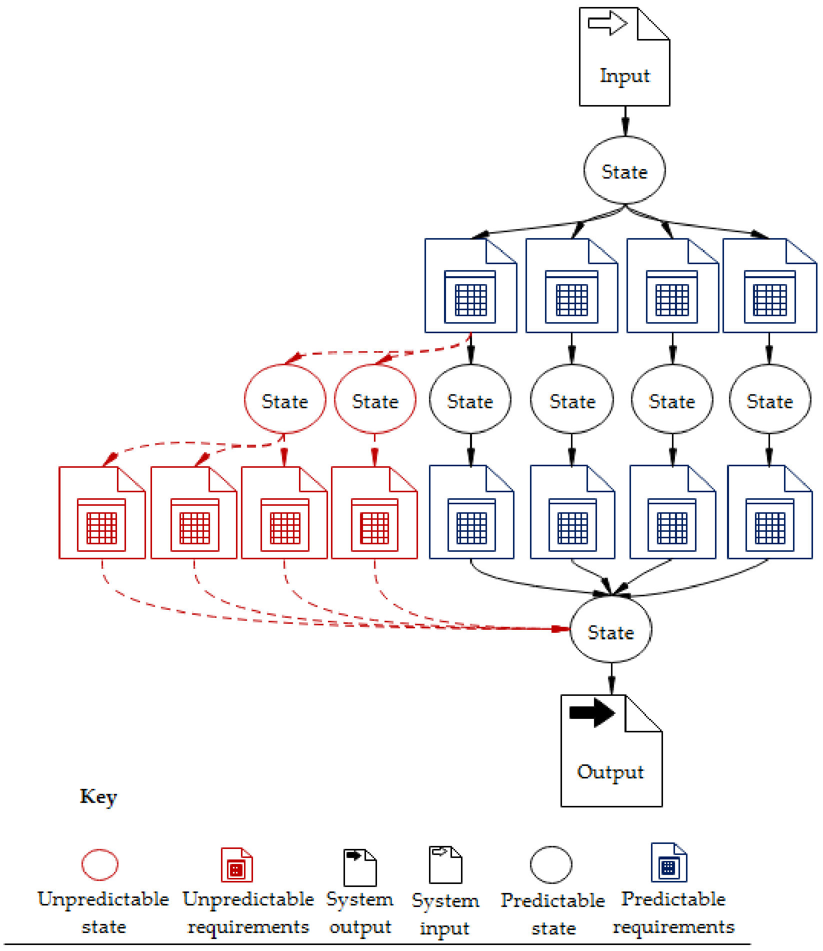


Figure 6. An illustration of predictable and unpredictable system states and requirements produced by the author of this thesis

If Figure 6 represents an entire system, unpredictable requirements (red) will introduce unpredictability in the final state of the system and consequently alter its output. Since unpredictable requirements impact the system's output (and final state), this unpredictability complicates decision-making, particularly at the output. Logically, decision-making under unpredictable changes is an important consideration particularly for managing the system's output state and requirement.

Ensuring an effective survival mission in cloud environments requires optimal decision-making on unpredictable and dynamic survivability objectives. However, decision-making requires monitoring and is critical but challenging where information is unknown with certainty and future systems behaviours are unpredictable. Moreover, such decision-making may rely upon monitoring a system whose observations may produce ambiguous or imprecise information. The challenge here is, therefore, how to make optimal decisions based upon estimate survival actions, estimate system states and estimate survivability requirement and parameters. It can be said that lack of knowledge on the states of VMs means that the general state of the cloud systems at this point is not known. In addition, heterogeneity and uncertain and emergent phenomenon in clouds mean that complete knowledge of the survivability states of cloud systems is unrealistic. As a result, monitoring only relies upon the probabilistic inference of future states.

The control notion is observed in intrusion detection system's wherein during operation, an alert is thrown to indicate the existence or absence of a threat, based upon what a detection model evaluates as true. In this sense, alerts are calibrated to correspond with a range of deviation from the detection model threshold. There are two challenges introduced by this approach. Design-wise, selecting a threshold value for the "true state" requires absolute accuracy, otherwise, a large or small threshold value results in false negatives or false positives, respectively. In cloud environments, UUURs makes this accuracy a near impossibility. Moreover, training a detection model to improve accuracy requires consistent monitoring of the target system; this is a known operational challenge. Even with an optimised detection model, the unpredictable element of UUURs means that countermeasures will likely be misplaced (or inadequately implemented), e.g. selecting a costly defensive response to address a non-extreme intrusion and vice-versa. This impacts upon resource requirements and cost, both inherent survivability tread-offs.

Large-scale networks including cloud computing are inherently complex (Wen *et al.*, 2017). While a complete and unanimous definition of complexity is somewhat contentious across many domains, the discussions in this chapter revolve around the scientific definition posited by (Foote, 2007), "phenomena, structure, aggregates, organisms, or problems that share common themes" (Foote, 2007)". Moreover, they are inherently complicated or intricate, are rarely completely deterministic, mathematical models of the system are usually complex and involve non-linear, ill-posed, or chaotic behaviour, and the systems are predisposed to unexpected outcomes or emergent behaviour (Foote, 2007). Figure 7 illustrates the formation of a complex system from components to a network of connected and interdependent components. On the left, it shows the building components of a system as a set of autonomous components whose transformation (graphic on the right), represents a complex system in which numerous autonomous components grow exponentially through connections (dotted lines) and interdependence (black note at edge of dotted line).

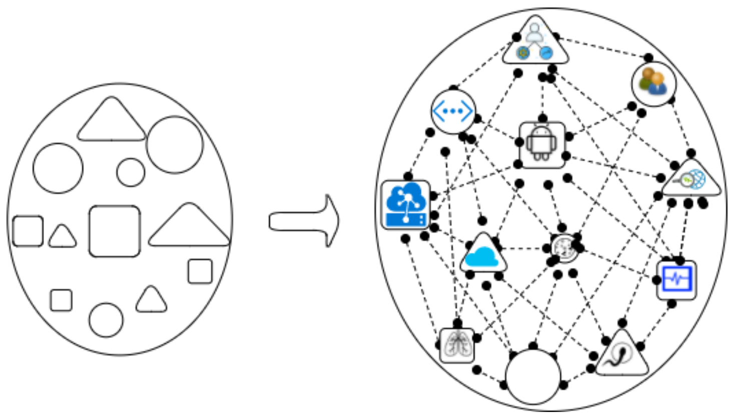


Figure 7. An illustration of a complex system formation produced by the author of this thesis

The nature of their connectivity defines the complexity of the system (in the global sense) rather than its characteristics. Moreover, autonomy enables components to adapt through local instructions and collectively synchronise (through cooperation and coordination) individual statuses resulting in a bottom-up form of order.

With current innovations and the proliferation of new devices and platforms, and multiparty collaborations involving third parties, coordinating and controlling interactions among cloud computing parties is often a complex task. The complex, unpredictable and dynamic nature of cloud computing requires intelligent systems control. Cloud security research should perhaps seek inspiration from well-established complex adaptive systems such as those in nature. While complexity in natural systems develops on a microscale through evolution, elements within complex systems are generally subject to selection and those best suited for the environment are chosen. In a free market economy, products are selected through market forces, whereas politicians in democracy are selected through elections/voting. Similarly, animals are selected through natural pressures such as predation and competition. Literature demonstrates that nature's superiority is multifaceted, enabling biological systems to adapt and survive and conjure solutions in a large search space with limited initial information. According to Mateos el al. (2013), the above makes biological systems prime alternatives in environments where efficient prediction is a challenge (Mateos, Pacini & Garino, 2013). Nonetheless, the use of bio-inspired approaches in cloud computing security is an under-researched area in contrast to the general cloud computing research domain (Stoykov, 2015). As suggested by Balusamy et al. (2015), bio-inspired algorithms have limited application in cloud computing. The following section investigates biological systems and application areas in computing security.

2.4 Bio-inspired systems

Nature has contributed to a range of domains, including finance (Brabazon & O'Neill, 2006) and robotics (Oates *et al.*, 2009), to name a few. The past ten years has seen growth in research interests around bio-inspired systems due to the growth in demand on networked systems and reliance on internet connectivity provided through an assortment of devices and infrastructures (Zheng & Sicker, 2013). For instance, biological self-organisation applied in wireless ad hoc networks enables clustering routing nodes to enhance the scalability of data forwarding protocols (Zheng & Sicker, 2013). Or Intrusion detection systems (IDSs) design inspired by the negative selection of T-cells that bind and kill infected or harmful cells (Zheng & Sicker, 2013). The immune system can adapt and self-protect by dynamically creating and destroying mutated or infected body cells, as it learns new threats and protects itself and its protective components (Sobh & Mostafa, 2011).

Biological/natural systems are abounding with attributes in distributed systems emanating from interactions among autonomous agents (Sayed, 2014). Their strength resides in the ability of autonomous entities to make local decisions, and continuously coordinate and share information to maintain a global form of order (Sayed, 2014). Hence, natural systems demonstrate effective self-attributes as successful phenomenon (Dressler & Akan, 2010). Table 6 summarises some bio-systems and classifies them according to their application area, and the strengths and weaknesses of each system. This section will review some notable biological systems and their application in computing. In addition, seven of the most successful natural preys are investigated to understand the survival dynamic; predation avoidance and anti-predation behaviours and mechanisms.

Table 6. Some example bio-systems application: strengths & weaknesses

Bio-inspiration	**Application Area**	**Strengths**	**Weakness**
Immune System	Anomaly & behaviour detection	Rapid detection of new & known antibodies	Slower detection
Foraging species	Adaptive capability	Efficient resource usage, robust & scalable	Communication overheard
Predator-prey	Highly adaptive Networks	Evolution/co-evolution & adaptation result in the survivability of species	Stability of the system depends on the arms race.
Immune (Idiotypic Network)	Self-organisation	In-depth self-learning capabilities	Stunted scalability

Biological systems have been a subject of research across the computing continuum stretching back to the 1980s (Priami, 2009). Surveys including (Zheng & Sicker, 2013) and (Ribeiro & Hansen, 2012), dedicated their efforts towards evaluating biologically inspired algorithms in computing-related applications. For instance, negative selection of T-cells' strategy to bind and kill infected or harmful cells in an Artificial Immune System (AIS) is a basis for designing IDS (Zheng & Sicker, 2013). Similarly, the adaptation of the memory and self-learning mechanism employed by B-cells in identifying and destroying pathogens inspired the design of IDSs (Zheng & Sicker, 2013). Among many works, (Dressler & Akan, 2010) present a comprehensive survey of bio-inspired networking protocols citing a substantial number of sources. These authors allude to the fact that immune-inspired algorithms form the basis for network security, specifically anomaly detection. They associate epidemiology to content distribution in computer networks, including the analysis of worms and virus spreading on the internet. Meisel at al. (2010) concurs and associate intrusion detection and malware propagation to AIS and Epidemiology, respectively, as complimentary bio-inspired domains.

Artificial immune system is are applied in a variety of areas and particularly lauded for success in IDS (Liang & Fengbin, 2013) based upon immune system detectors which determine the performance of the detection component of the immune system (Ji & Dasgupta, 2009). Several works including (Yang *et al.*, 2009) (Kephart, 1994) (Kim *et al.*, 2007) (Gonzalez & Dasgupta, 2003) to name a few, employ immune inspired approaches for developing computer security mechanism based on the self-adaptive, self-learning, self-organizing, parallel processing and distributed coordination attributes of AISs. In addition, the authors in (Fang *et al.*, 2012) propose AIS phishing detection that is inspired by part of the immune system's response mechanism to pathogens; immature T-lymphocytes life cycle. Nonetheless, the authors contend to the fact that using a static instead of dynamic fire-threshold value on their detectors, their system suffers from deficiencies (Fang *et al.*, 2012). Similarly, (Saudi *et al.*, 2008) explore the use of an immune system inspired concept of apoptosis for in computer network security based upon the immune system's programmed action of destroying infected or mutated cells (Saudi *et al.*, 2008). A comprehensive review of phishing email filtering techniques is presented by (Almomani *et al.*, 2013), while works by (Gupta, Arachchilage & Psannis, 2017) reviews current literature and present a range of solutions proposed against identified attacks.

Genetic algorithms (GA) are stochastic search methods inspired from principles in biologicals systems where problem-solving is indirect through an evolution of solutions, with subsequent generations of solutions, in turn, yielding the best solution to a problem (Hordijk, 2005). Along similar lines, (Sun & Cheng, 2009) proposed GTAP gene-inspired algorithm for user authentication where users from a "family" are identified by a unique gene certificate (synonymous with unique signatures), and users are authenticated upon a positive analysis of their gene code (Sun & Cheng, 2009). According to the authors, simulation results for GTAP demonstrated superiority in safety and security by countering the deficiencies in safety passwords and ambiguity of subject information in certificates presented in traditional mechanisms (Sun & Cheng, 2009).

In other works (Isasi & Hernandez, 2004), genetic algorithms are implemented in cryptography to evaluate and enhance the complexities of encryption systems. An interested reader is referred to a complete guide for cryptographic solutions for computer security presented by (Gupta, B., Agrawal, D.P. and Yamaguchi, 2016). In cryptanalysis, where an attack mechanism is implemented to assess the strength of an encryption system, GA is argued to be highly successful in substitution cyphers (Verma, Dave & Joshi, 2007) and transposition cyphers (Toemeh & Arumugam, 2015). While genetic-inspired approaches are superior for efficiency and specificity for selecting the unique features, for instance, for trust assurance, authentication, authorization, or access control, or in intrusion detection systems (IDSs), they are often complex and can only be used in a specific, rather than general manner (Modi *et al.*, 2013). Thus, where solutions have been proposed for cloud security (Olumide *et al.*, 2015) (Jinyin & Dongyong, 2013), (Wang & Yan, 2010), they are applied as pure solutions for encryption, managing the security of data in storage, intrusion detection, trust management, etc.

Inspired by the reliability of gene identification and assignment inherent in biological systems, Wang et al. propose the Family-gene Based model for Cloud Trust (FBCT) to address existing limitations inherent in PKI-based systems, which include challenges in identifying nodes within cloud environments, access control, and third party authentication system (Wang *et al.*, 2010). By adopting biological principles in family genes, their model provides solutions for trust in the cloud computing domain.

Although neural networks are generally popular in pattern recognition and classification, and noise filtering, they are useful in other areas including the use of biometrics in security (Hordijk, 2005). Key to their success is their accuracy in feature extraction and efficiency in classification, i.e. low rejection rate and high positive classification (Hordijk, 2005). Along these lines, (Shorov & Kotenko, 2014) proposed the Network Nervous System as a mechanism for effective protection against distributed denial of service (DDoS) attacks, grounded on the biological metaphor of the human central nervous systems; distributed information gathering and processing, coordination, and identification activities. Their work rests on the basis that traditional security tools fail to cope with the escalating power of attacks on computing infrastructures (Shorov & Kotenko, 2014).

Ant colonies have been applied for routing traffic optimisation, for instance in works by (Baran & Sosa, 2000) who evaluate an optimisation algorithm; AntNet, in which agents concurrently traverse a network and exchange information synonymous with stigmergy in insects. According to the authors, this algorithm exhibited superior performance in contrast to its competitors (Caro & Dorigo, 1998). (Rafsanjani & Fatemidokht, 2015) Proposed FBeeAd-Hoc as a security framework for routing problems in Mobile ad hoc networks (MANET) using fuzzy set theory and digital signature (Rafsanjani & Fatemidokht, 2015). Other models including the Trust Ant Colony System (TACS) (Marmol, Perez & Skarmeta, 2009), AntRep algorithm based on swarm intelligence (Wang, Zeng & Yuan, 2006), Time Based Dynamic Trust Model (TBDTM) (Zhuo, Zhengding & Kai, 2006) to name a few, have been proposed for distributed systems. Nevertheless, it is imperative to emphasise the need for comprehensive testing and evaluation before their use in cloud environments (Firdhous, Ghazali & Hassan, 2012).

Works by (Gupta & DuVarney, 2004) extends on the predator model, to propose countermeasures against automated mobile malware in networks. The authors propose models for self-propagating, self-defending and mobility attributes found in predating animals. Their works premises on the notion that traditional countermeasures do not scale to solve security challenges existing in distributed systems (Gupta & DuVarney, 2004). (Grimes, 2001) suggest the use of predator models as inspirational solutions against viruses and worms.

Works by (Finstadsveen & Begnum, 2011) explore the use to biological metaphors as a basis for designing, modelling and implementing a cloud-based web service. According to these authors, a cloud-based web service can deal with counter stability issues that arise from long-running processes and security attacks. Also, they argue that a zebra herd-inspired approach simplifies not only complex technical challenges, but also enhances new designs for automating self-management processes for system administrators.

Table 7. A summary of bio-inspired algorithms proposed for cloud (C) and non-cloud (NC) environments

Algorithm	**Biological parentage**	**Domain**
Multiple Sequence Alignment (MSA) algorithms	Protein structure	NC
IDS detector optimisation algorithm	Co-evolution in populations	NC
Data Security strategy	Immune systems	C
Secure Data Storage	Physiological & behavioural patterns	NC
AIS for phishing Detection	T-lymphocytes life cycle	C
Integrated Circuit Metrics (ICMetrics)	Human properties & features	NC
Biologically inspired Resilient	Cells & organisms (sea chameleon)	NS
Data Hiding for Resource Sharing	DNA sequences	C
Organic Resilience Approach against attacks and failure	Immune system	C
Security based on Face Recognition	Facial features	NC
Family-gene Based model for Cloud Trust (FBCT)	Genetics	C
Agent of Network Danger Evaluation	Immune System	C
Supervised learning classifier with real-time extraction (UCSSE)	Genetic-based machine learning	NC
Fraud detection & improper use	Immune System	NC
Extension of Predator-Prey Model	Predator-prey communities	NC
Computer Immune System	Innate immune phase	NC
AntNet: Ant colony optimisation algorithm & OS theory	Ant colony	NC

2.4.1 Prey survival against predation

Vigilance, alarm calls, mobbing and group living are anti-predator behaviours shared among vervet monkeys (*Cercopithecus aethiops*). Furthermore, flight, alarm calls and response to alarm calls in vervet are responses to alarm calls associated with specific predator species (Isbell, 1994). Alarm signals in vervet monkeys perform multiple-duties, ranging from predator deterrents, or distress signals to call in mobbers (Smith, 1992).

Social behaviours in Thomson's gazelles such as their alert posture, galloping, stotting, and soft alarm calls are argued to release alertness and flight information to avoid predation (Walther, 1969). Evidence in the literature supports the claim that Stotting in Thompson gazelles is a vital tool for avoiding predation. According to (Caro, 1986), stotting startles or confuses a predator and thus forms an anti-ambush evasion technique. Thomson gazelle's mothers are known to adopt aggressive strategies to divert predators from hunting their fawns (Fitzgibbon, 1990a). As noted by (Fitzgibbon, 1990b), predation avoidance in Thompson gazelles is also associated with grouping behaviours, where larger groups have improved predator detection capabilities, and significantly reduced vulnerability factor against the cheetah, (*Acinonyx jubatus*).

Five predation avoidance strategies are employed by caterpillars against predating birds; restrict feeding to undersides of leaves, forage at night, use leaves for movement while foraging, and distance themselves from an unfinished leaf, or snip it off altogether (Heinrich, 1979). group living is argued to positively enhance protection, as well as warning signals, defensive movement, and regurgitating noxious chemicals may increase survivability. Literature suggests that Zebras flee predating lions according to their proactive responsiveness to a prior assessed risk level, and reactive responses to imminent predation (Courbin, Loveridge & Macdonald, 2015).

The choice of predation avoidance or anti-predation mechanism is hugely important in Meerkat (*Suricata suricatta*) communities as they live under high predation pressures while occupying challenging foraging niche (Thornton & Clutton-Brock, 2011). As such, social learning and effective cooperation initiate key survival behaviours, including fleeing non-specific predators, mobbing against predating snakes, functional referential alarm calls, or running to bolts holes in response to aerial predators (Thornton & Clutton-Brock, 2011). In addition, Meerkat depends hugely on group living through communal vigilance (Roux *et al.*, 2009). Unlike the response to alarm calls, vigilance occurs in the absence or presence of a predator or danger (Voellmy *et al.*, 2014). The presence of strong predation avoidance responses in nature's prey species demonstrates that past species interactions affect present distributions and may play an important role in the ongoing assembly of contemporary communities. Such avoidance behaviours in a growing number of species fundamentally alter the view of the processes affecting species distributions and the process of community assembly (Resetarits, 2001).

Table 8 summarises existing predation avoidance and anti-predation behaviours and techniques in natural prey, including but not limited to those discussed above.

Table 8. A matrix linking prey survival mechanisms (anti-predation and predation avoidance) to prey species

	Attributes: prey survival technique & behaviour	**Prey**						
		Plants	Lizard	Vervet	Gazelle	Moth	Meerkat	Zebra
Anti-predation	Alarm calling	×	×	✓	✓	×	✓	✓
	Chemical defence	✓	×	×	×	×	×	×
	Fight-back	✓	✓	✓	×	✓	✓	✓
	Stotting	×	×	×	✓	×	×	×
	Group living	✓	×	✓	✓	✓	✓	✓
	Mobbing	×	×	✓	×	×	✓	×
	Aposematic	✓	×	×	×	✓	×	✓
	Mimicry	✓	✓	×	×	✓	×	×
Predator avoidance	Camouflaging	✓	✓	×	✓	✓	×	✓
	Masquerade	✓	✓	×	×	✓	✓	✓

The predator-prey dynamic highlights the importance and the consequences of interactions between two species, i.e. it demonstrates how the functions of a community depend on the characteristics of that community. Before biological systems may be applied, several problems should be considered. Regardless, cloud computing solution must be developed based on the foundations harvested from nature. As suggested by (Hariri, Eltoweissy & Al-Nashif, 2011), existing solutions are limited as they do not adapt and escalate their security strategies to counteract the intensity and sheer aggressiveness of an adversary. Cybenko et al. (2014) concurs and suggests security countermeasures are only successful in traditional networks and not in cloud computing environments where persistent adversaries and zero-day attacks can systematically plan their attacks and persist within a compromised network. On the basis of the foregoing, the authors postulate the rise in popularity of Adaptive Cyber Defence (ACD) approaches such as bio-inspired systems, based on their ability to optimise unpredictability and maximise the adaptive configurations in attack surface, thereby raising the cost of an attack for the adversary (Cybenko *et al.*, 2014).

Unlike the single solution approach noted in Section 2.2, Table 8 shows that prey animals possess rapid reactions based upon the ability to make local but synchronised decisions. Escalation therefore enables a system to invoke appropriate proactive responses (ranging from passive to aggressive) based on the nature of the threat. To enhance the survivability in cloud infrastructures, proactive strategies including deceptive and pre-emptive should be implemented to maintain the state of the environment at best or ensures the system copes with any form of destructive encounter. Given the significant success of biological systems, it seems logical to investigate theoretical underpinnings that describe core elements and their application as plausible approaches in the security continuum

2.5 Theoretic perspectives

Theoretical perspectives form one of the most important aspects of the research process. As noted by (Grant & Osanloo, 2014), to construct knowledge for research and develop an adequate scientific rationale for that research, theoretical perspectives provide the blueprint with respect to the structure of the research, the research problem, the purpose and the implications of the research findings. Thus, the purpose of this section is to give an overview of this research's theoretical perspective and clarify how this perspective integrates the selected theory under this investigation, as well as key concepts and definitions relevant to the cloud computing topic area under the current consideration. Different scientific domains apply variable methodologies and definitions of terms for essentially the same aspect of what can be termed "reality". When considering natural and artificial systems, the system-ness of the foregoing is relevant to abstraction or de-abstraction of theories. (Malecic, 2017)'s opinion is that, understanding this abstraction and de-abstraction will enable effective assessment of their casual and explanatory power.

Control theory's classical methodologies and assumptions handle complexity through enhanced adaptation in the presence of unpredictable and dynamic changes (Smith, 1979) (Landau, 1999). Several works demonstrate the efficacy of stochastic control to optimise expected value of cost under extreme uncertainty and unpredictability. This control concept describes self-adaptation as identified among a huge state-of-the-art in a classification presented by Andersson et al. (2009).

The General systems theory (GST) suggests that functions that convert inputs into required system outputs can be designed and controlled provided that all contexts are known (Checkland, 1981). This assumption suggests that regardless of internal composition, all systems; natural, man-made, abstract, conceptual or concrete, have common characteristics (Checkland, 1981). Thus, quite generally, GST aims to formulate generalised theories; system dynamics, goal-oriented behaviour, hierarchical structures, and control processes, develop methodological ways for describing the functions and behaviours of systems objects, as well as to expand generalised models of systems (Skyttner, 2010). However, as has been explained earlier, with UUURs, there is a limit to the amount of initial information affected systems have.

Game theory is a widely applied in security research. While there exist several quantitative pieces of research on risk analysis and security modelling, quite precisely, fewer studies quantitatively focus on the survivability of cloud systems in view of UUURs. In the former, examples include (Cox Louis Anthony (Tony), 2009)(Wang *et al.*, 2012c)(Furuncu & Sogukpinar, 2015), and in the latter, examples include (Fan *et al.*, 2013)(Kamhoua *et al.*, 2014)(Xiao *et al.*, 2018). However, (Jormakka & Mölsä, 2005) contend with the impossibility of probabilistic analysis of attacks in special cases where time dependency associated with lack of knowledge makes the endeavour a practical impossibility. Game theory has also been applied for survivability as a game mode in which a game strategy is a central requirement. As noted by (Mezzetti & Samuelson, 2006), the Game theory is guided by the principle that clear, stable preferences motivate choices over decision outcomes, and strategic action considers the relationship between one's choice and the decisions of others. For instance, one where a bold strategy results in domination (terrorist game), or a mixed defense strategies where domination can be reduced (evildoer game), or where domination lasts for limited duration (vandal game), or where altering observations and orientation of an adversary (meta-strategies) gives the advantage to a defender (Shehabat & Mitew, 2017). The current author concurs with (Jormakka & Mölsä, 2005), in particular considering the notion of unpredictable, latent, unobserved or unobservable risks proffered by (Ma, Krings & Sheldon, 2009; Ma & Krings, 2011; Ma, 2010). Thus, game theory as applied in cloud survivability becomes fundamentally deficient when survivability threats are unpredictable and where risks are latent or unobservable.

2.5.1 Theoretical ecology

In theoretical ecology, the core concepts around the predator-prey system are interactions and their movement across a habitat. Within this domain of theory, functional and numeric response are common concepts (Fryxell *et al.*, 2007). Functional response is the prediction of the rate at which a predator consumes prey as a function of predators and the density of prey (Petchey, 2000). On the other hand, numeric response is associated with the rate of reproduction as a function of food density (Petchey, 2000). Classical predator-prey theory has its origins around Malthus-Verhulst logistic equations and Lotka-Volterra equations, which describe predator and prey's functional and numeric responses to changes in their habitat (Vitanov, Dimitrova & Ausloos, 2010). Some common problems when applying the classical predator-prey theory arise due to contradictions caused by the principle of mass action to predator-prey interactions. The principle of mass action (Kloeden & Pötzsche, 2010) postulates that the rate at which a predator consumes prey should only be prey dependent. Boutin (1995) further suggests the effects of predation on prey as generally dependent upon prey communities, rather than individuals.

The selection principle explains the adaptive tendencies of an immune system to antigens (Castro & Zuben, 2002), self-organisation in evolution (Kauffman, 1993), and is prominent in various "arms-race" analogies. In their work, Dawkins and Krebs (1979), exemplify selection using the life-dinner principle, wherein an entity with better selection out-evolves its competition. According to these authors, the life-dinner principle describes an inherent asymmetry between a predator and prey in relation to the success of predation, the success of evading predation, and vice-versa. Noting to a cost-benefit analysis modelled by Abrams around the life-diner principle, (Vermeij, 1994) drew the observation that an arms-race was highly likely where predator and prey interact, the outcomes of which were determined by birth and death rates.

Cope's Rule as postulated in works by (Hone & Benton, 2005) (Kingsolver & Pfennig, 2004) is an evolutionary concept which postulates that entities with evolving lineages adapt by growing in size over time. According to these authors, such adaption increases predation success, increases defence against predation, improves intelligence, and improves survival and adaptation to climate changes (Hone & Benton, 2005). Defence against predation has been a subject of study in a wide variety defence systems employed in insect egg (Eisner *et al.*, 2000), and defence studies in plant science (Hay et al. 1987). As in most defence systems, the objective for defence is to improve the survival chances of prey, against a predator. For the predator, the objective for survival is to become a successful hunter while for the prey, it is about avoiding predation. In this case, the Cope's Rule, the Life-Dinner Principle, and the concept of arms-race outlined above come to the fore.

In studies to investigation ecological patterns and evolutionary implication of predation among primate communities, group size and adaptation are identified as notable demographical and behavioural factors of predation, respectively. On the other hand, the notion of inferred evidence is associated with the size of a group. According to Isbell (1994), vigilance is higher in larger groups and this improves the reaction to approaching predator. On the contrary, Isbell (1994) suggests small likelihood of avoiding predation in smaller groups, based upon the likelihood to be detected. Thus, it is perhaps logical to draw parallels between predation avoidance capabilities in prey communities to the security capabilities in cloud computing. The following logical assumptions have their theoretical foundations in the foregoing and a basis for analogical reasoning presented in Chapter 4:

Parallel 1: As direct observation and inference provide primary information and alarms calls and flight are adapted to increase survivability against specific predator species in primate communities, it is local that Swarm intelligence can be used for collaborative proactive countermeasures including monitoring, response and security event management towards the survivability of a cloud computing infrastructures.

Parallel 2: Just as mechanisms such as concealment, flight, vigilance, and alarm calls in prey are attributed to predation avoidance (Isbell, 1994), proactive cloud security approaches including deception, deterrence, obfuscation, pre-emptying and counter-attack can be adapted to increase the survivability of cloud infrastructure.

Analogy 3: Just as threats of capture triggers last resort anti-predation behaviours such as mobbing and fighting back in prey (Isbell, 1994), threats such as malicious users, advanced persistent threats (APTs) etc. in cloud environments necessitate the use of proactive countermeasures able to escalate from passive to aggressive methods, based upon the persistence of a threat.

While bio-inspired approaches have found use as artificial alternatives to mitigate deficiencies a range of diverse areas, bio-inspired approaches have clear distinctions based upon their application domain. Three main classes shown in Figure 8 and briefly explained below provides the basis for the bio-inspired approach follows in Chapter 4.

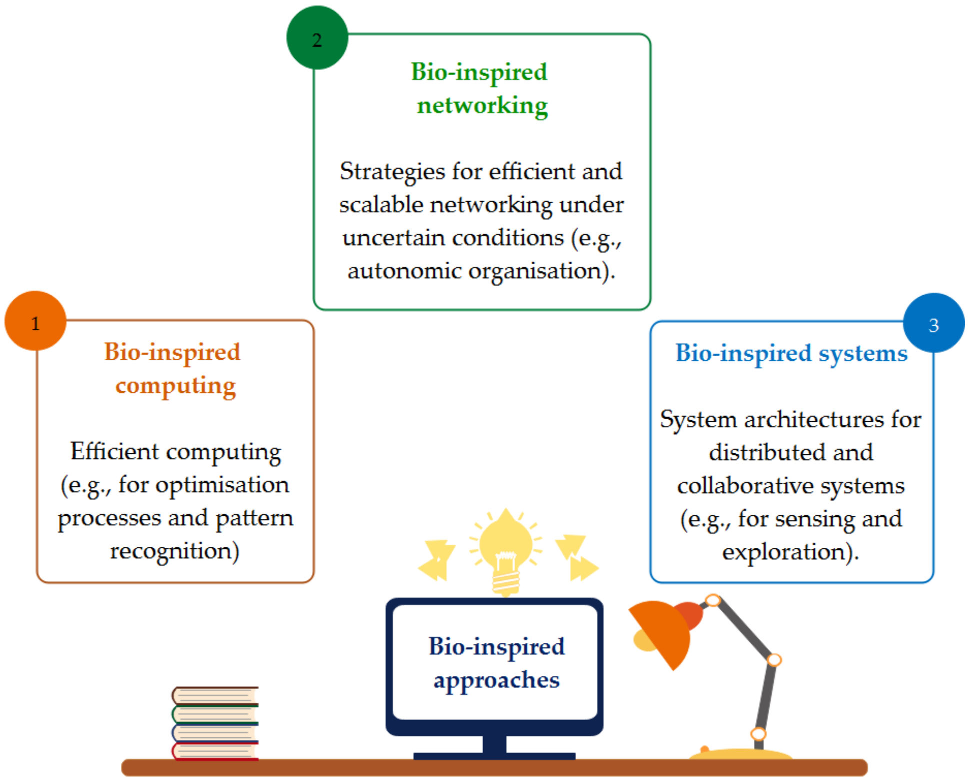

2
Bio-inspired networking
Strategies for efficient and scalable networking under uncertain conditions (e.g., autonomic organisation).
1
Bio-inspired computing
Efficient computing (e.g., for optimisation processes and pattern recognition)
3
Bio-inspired systems
System architectures for distributed and collaborative systems (e.g., for sensing and exploration).
Bio-inspired approaches

Figure 8. An classification of Bio-inspired approaches produced by the author of this thesis

Bio-inspired systems: Are comprised of a domain of architectures for extensively distributed and collaborative systems. As observed by (Dressler & Akan, 2010), exploration and distributed sensing are some of the common applications.

Bio-inspired networking: Is comprised of a domain for addressing phenomenon under uncertainty. For instance, applying autonomic organisation in large-scale distributed systems to facilitate efficiency and scalability.

Bio-inspired computing: This is comprised of a domain of algorithms for efficient computing, for instance, when applied for optimisation solutions.

Bio-inspired design entails a systematic process for developing and mapping analogies for the survivability phenomenon, and the development of bio-inspired tools and platforms for cloud computing.

2.6 Summary

The review of cloud computing security challenges identifies the findings summarised below.

- Multiple perspectives imply that specific solutions remain largely prevalent. Theses must be integrated into a holistic representation of security challenges, towards developing solutions that holistically addresses cloud security issues.
- The cloud computing environment, while possessing some attractive attributes, enable adversaries. It is important while developing future solutions, to address this contradiction; address security risks yet preserving cloud computing's attractive attributes.

The review of the survivability context identifies the findings summarised below.

- There is limited research focusing on survivability architecting specific to cloud computing environments. This indicates an opportunity for designing survivability solutions for cloud computing's many security challenges.

The review of bio-inspired systems identifies the findings summarised below:

- Despite being an under-researched area for cloud computing survivability, the predator-prey system shows clear survival concepts that are exploitable for cloud computing survivability.
- To adopt or adapt biological concepts, there is need for a systematic method to transfer natural concepts to computing. In addition, an approach to create innovative solutions by resolving contradiction.

Chapter 3 A holistic taxonomy of cloud security challenges

This chapter presents a holistic taxonomy of cloud computing security challenges (contributing publication PR2). It structured as a literature review in relation to reviewing and analysing existing cloud security challenges classifications. This chapter is significant to the research hypothesis as it helps conceptualise the complete cloud systems. Whereas there maybe reference to specific security issues, this chapter is not interested in the in-depth analysis of specific security attack paths or attack graphs.

3.1 Introduction

Much of existing research focuses on evaluating and classifying cloud security challenges, resulting in numerous and disparate perspectives and excesses of perspective-based taxonomies. For instance, the perspective of security vulnerabilities in web applications (Johns, 2011), a virtualisation perspective (Lombardi & Pietro, 2011), the perspective of service delivery models (Subashini & Kavitha, 2011), the perspective of a service-oriented organisation (Roy *et al.*, 2015), and so on. Disparate perspectives limit the extent to which the complete cloud security landscape is viewed and understood. Critically, commonly shared but unexplored cloud security challenges remain unresolved and entrenched when perspectives do not overlap. This chapter is motivated by a lack of research focusing on consolidating these disparate perspectives to benefit cloud security countermeasure research and development. Hence, it interrogates how cloud security challenges are viewed and classified in the existing literature, which helps to identify gap areas and their possible implications. Clear gap areas will form a useful gateway to a new classification encompassing all perspectives of the cloud and cloud's functionalities.

Through a meta-synthesis, textual classifications are re-conceptualised into hierarchical tree-like structures, which simplify visualisation for the reader and clearly illustrates entity relationships amongst cloud categories (Polash, Abuhussein & Shiva, 2014). The taxonomy design concept is premised around (Howard & Longstaff, 1998)'s notion of a satisfactory taxonomy; mutually exclusive, unambiguous, repeatable, acceptable and useful. From a high-level, it distinguishes cloud security challenges based on their source and/or origin, as follows: security challenges adopted by the cloud, security challenges inherent to the cloud architecture, and security challenges triggered by cloud implementations. For the purposes of the current chapter, inherent delay is identifying a specific threat is assumed as a worst-case scenario with significant adverse impacts to cloud security. Hence, the holistic approach proposed is innovative and insightful as it introduces the possibility of delay analysis as an element of a holistic cloud security countermeasures design.

The remainder of this chapter is as follows: Section 3.2 presents the holistic approach; outlines the requirements and design approach towards the holistic notion. Section 3.3 reviews existing taxonomies and reconceptualising each into a hierarchical structure. Section 3.4 presents the holistic taxonomy for cloud security challenges and discusses its implications. Section 3.5 concludes the chapter.

3.2 Holistic approach

This section presents a holistic taxonomy design approach (**Error! Reference source not found.** illustrates the general holistic approach.) to gather, interpret and re-conceptualise cloud computing security knowledge. The interpretation of existing taxonomies yields newly transformed taxonomies, which are easier to visualise. Furthermore, the analysis identifies deficiencies that exist among perspectives in a non-ambiguous manner. Ultimately, this approach will ensure that all cloud computing security categories are comprehensively presented, which is significant for security countermeasure design, development and maintenance.

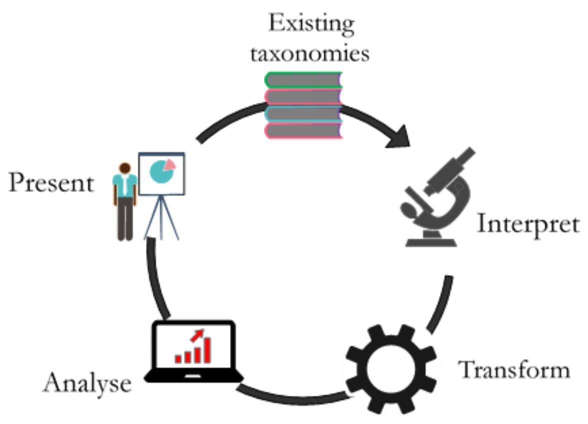


Figure 9. A illustration of the holistic design approach produced by the author of this thesis

Figure 10 shows the workflow processes for the generic holistic approach. The input model (top left) defines cloud computing security's multiple perspectives, while the output model (top right) is the holistic tool for the cloud computing security challenges.

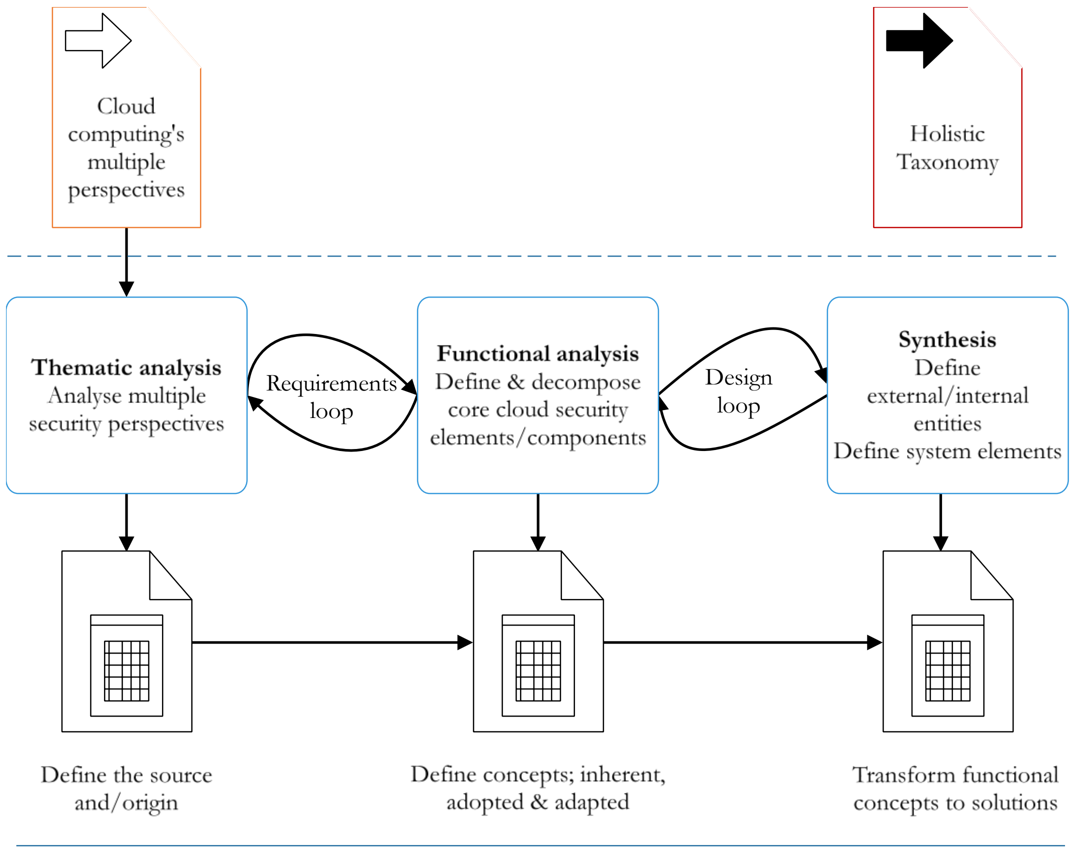

Cloud computing's multiple perspectives
Holistic Taxonomy
Thematic analysis
Analyse multiple security perspectives
Requirements loop
Functional analysis
Define & decompose core cloud security elements/components
Design loop
Synthesis
Define external/internal entities
Define system elements
Define the source and/origin
Define concepts; inherent, adopted & adapted
Transform functional concepts to solutions

Figure 10. An illustration of the holistic workflow processes produced by the author of this thesis

The output model has significance for holistic baselining, security specifications and configuration or security planning. A taxonomy requirements loop handles thematic modelling and functional analysis of the cloud security paradigm; deconstructing, defining and assigning categories to central concepts. A design loop models the continuous processes of the functional analysis mentioned above, and the transformation of functional concepts to physical cloud countermeasures or solutions. Moreover, it facilitates the definition and extension of additional security concepts and elements (these will be represented as categories, subcategories, sub-subcategories, etc.) of the holistic taxonomy. Analysis and control enable the evaluation of the analysed cloud security concepts, and exploration of gap areas and trade-offs, both useful inputs for the synthesis model. For the interested reader, other commonly applied taxonomy development methods exist in literature; (Nickerson, Varshney & Muntermann, 2013) presents an interesting cross-domain review.

3.2.1 Requirements

In this research, an abstraction of the holistic view to cloud security challenges is hypothesised as one which considers that security challenges have a source or origin. This notion is aimed to facilitate the comprehensive organisation of key security issues into high-level concepts that are extensible for the further investigation and corresponding countermeasure development.

The following are suggested as necessary requirements: (1) Acceptability and completeness (Amoroso, 1994, Howard, 1997). (2) Mutual exclusivity, repeatability, unambiguity and usefulness (Howard, 1998), Determinism (Krsul, 2014), and (3) Compliant security terminology (Lindqvist & Jonsson, 1997) and well-defined terms (Nasr, Abou El Kalam & Fraboul, 2011). Thus, this research formally expands upon the definition a holistic taxonomy for cloud computing security challenges summarised by (Hansman & Hunt, 2005).

Definition 1. *A taxonomy satisfies the holistic property if and only if, all aspects of the source or origin of a security challenge are considered. Also, requirements and dimensions inform the design of main behaviours in a security challenge domain, i.e. the dimensions of a security target, a specific vulnerability and payload are considered.*

Figure 11 illustrates this conception, with each of its components discussed in turn below.

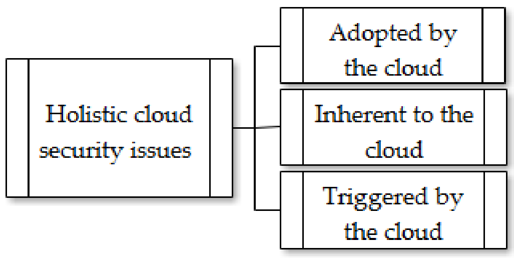


Figure 11. A high-level holistic view of cloud security challenges produced by the author of this thesis

This definition abstractly suggests that some cloud security challenges are directly adopted by the cloud from other technologies. In addition, that others are inherent to the cloud, embedded among key features such as scalability, broad network access, pay-per-use, etc. Moreover, it also implies that other security concerns are triggered by various forms of cloud computing implementations.

3.3 Review of existing classifications

Figure 12 is an illustration of unique cloud security challenges that are inherent to cloud computing. For instance, control mechanisms are well-established in a traditional "on-site" computing model; where data and services are housed within a perimeter to ensure availability. Even where data must leave the premise, for instance, with remote workers, security policies and mechanisms exist to ensure that data remain secure (confidential, integral & available), and accountability for whom, where and how data is being held and processed are mature. However, due to the de-perimeterised nature of the cloud, and the ambiguity in data location information, traditional these policies cease to be effective, causing a great deal of discomfort primarily due to the loss of control (Subashini & Kavitha, 2011).

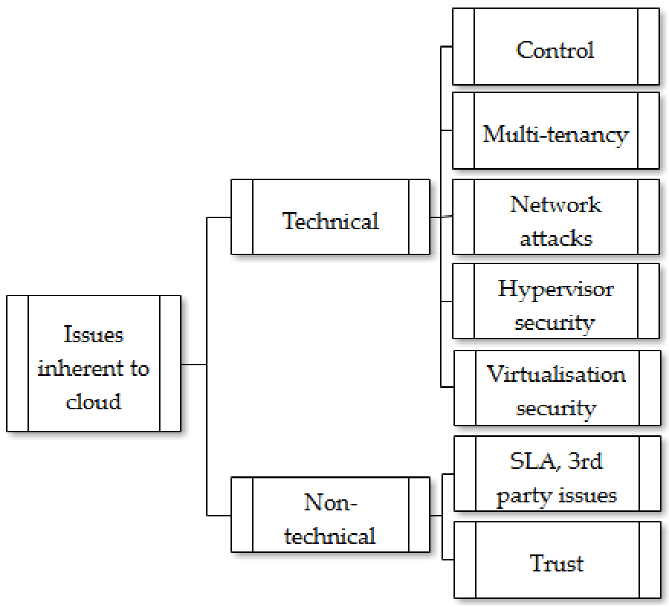


Figure 12. Security challenges inherent to cloud computing produced by the author of this thesis.

Figure 13 is an illustration of cloud computing security challenges that arise due to implementations in the cloud.

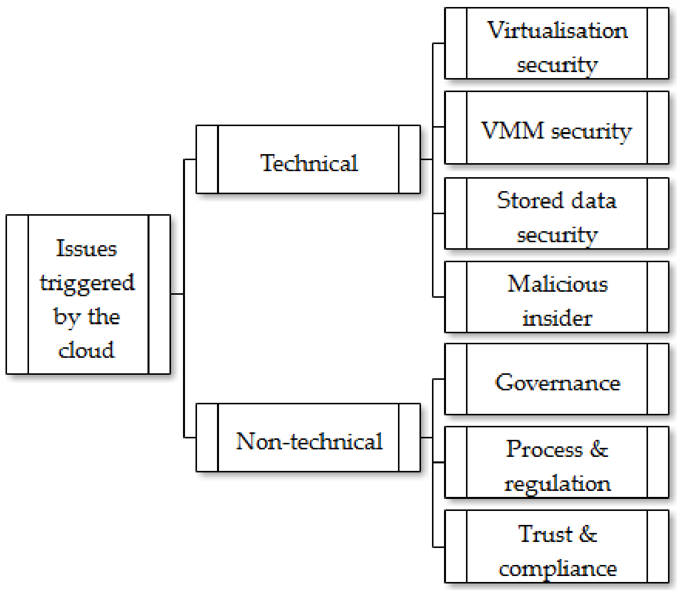


Figure 13. Security challenges triggered by cloud computing produced by the author of this thesis.

Bigdata's compute-intensive business analytics; high velocity, high capacity and high variety data, is an upwards trend (Chardonnens *et al.*, 2013)(García *et al.*, 2016). With cloud's rapid elasticity, resource pooling, on-demand access, transference of risk, etc. there is an inevitable convergence of technologies (Assunção *et al.*, 2015). From a security point of view, big data analytics is critical as it can improve the visualisation of the security domain by correlating security-relevant data (O'Connell *et al.*, 2014). However, despite its benefits, big data poses a threat to the cloud primarily due to the high volumes of data. As an example, privacy concerns arise where large databases collected through data mining are compromised. An example would be Google's infrastructures that collect and analyse data for advertising (Chow *et al.*, 2009b). While virtualisation enables essential cloud features such as location independence, resource pooling, multi-tenancy and rapid elasticity, it inadvertently elevates traditional security challenges. For instance, cloud customers depend on an internet connection to access cloud resources. In a traditional sense, DoS attacks could therefore focus on network entry points with high IP packets (Sabahi, 2011). However, this same attack poses a devastating impact in cloud environments. Moreover, malicious agents consider what resources they can gain in an attack, how much effort is required to compromise a target and how much access they have to the target (Grobauer, Walloschek & Stocker, 2011). The fact that cloud services are offered as a service, where a user pays for what they use, it means that malicious agents have easy access into the cloud infrastructure and act a constant threat in a multi-tenant environment.

Figure 14 illustrates this research's conceptualisation of unique issues adopted by the cloud. The argument that some cloud security challenges are not new to information security is shared by several authors including (Zissis & Lekkas, 2012)(Subashini & Kavitha, 2011)(Chen, Paxson & Katz, 2010)(Pearson & Benameur, 2010). Traditional challenges commonly refer to augmented and well-known security issues that pose security challenges for the cloud, mainly due to the physical or logical aspects of the cloud (Yu *et al.*, 2015). Such security challenges are non-specific/unique, however, they render existing security mechanisms ineffective in cloud environments (Hashizume, Yoshioka & Fernandez, 2013). For instance, by extension, the Bring Your Own Device (BYOD) phenomenon, which enables access to information on devices that may be outside the ownership and control of an organisation, introduce security concerns around regulatory compliance, data leakage, data breaches, data theft, etc. (Morrow, 2012)privacy and ultimately the right to privacy is broadly speaking a human right from a legal perspective in the United Kingdom and Europe. Privacy laws, e.g. Privacy and Human Rights Act 1998 in a way guarantees that everyone's right to such privacy (Commission, 2016). Hence, privacy in the context of personal data is arguably enshrined and enforceable under such Acts.

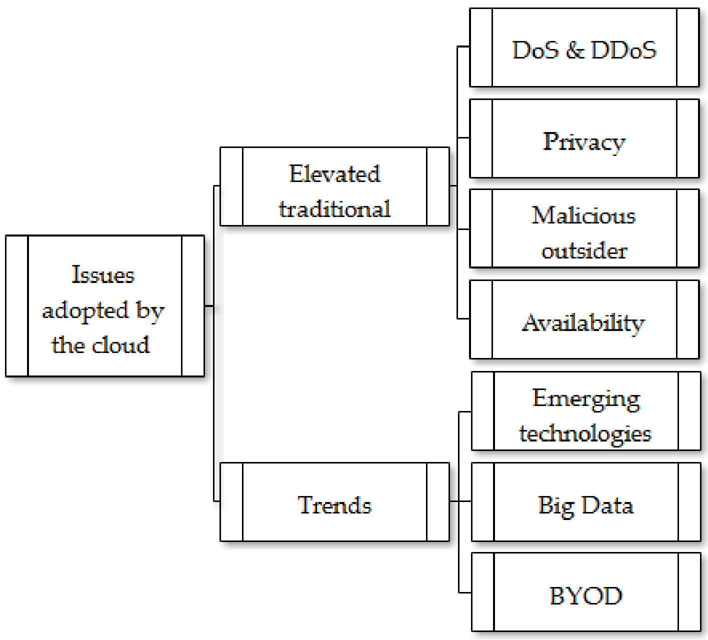


Figure 14. Security challenges adopted by cloud computing produced by the author of this thesis.

However, when considering privacy in the cloud perspective, it is important to acknowledge the following contexts; Clouds are geo-dislocated, meaning that private data can be held and processed in any location, known or unknown, arguably implying that the enforcement of privacy laws becomes a challenge due to jurisdictional discrepancies. In this context, privacy challenges that existed in the traditional context, whose legal mechanisms for enforcement exist/existed, are leveraged in this new environment. As mentioned earlier, the holistic notion in this paper is informed by the view that cloud security issues and challenges can primarily be distinguished according to their source or origin. The source/origin notion is suggested by (Wallom *et al.*, 2011) to emphasize the concept of VM trustworthiness when performing critical computation. While many similarities in existing cloud security taxonomies revolve around similar security issues or challenges, these highlight critical areas for research and solutions design for a cloud-wide perspective of security.

Figure 15 shows unique security issues unique to private clouds. Since private cloud infrastructures are generally managed on-site by the organisation and in some cases by a 3*rd* party/ external organisation, a CSP is generally able to specify security configurations while retaining some level of control. In some cases, issues arise as the private provider relinquishes control over how and if configurations are fully implemented since this control remains under the infrastructure owner's domain. Consequently, limited control leads to trust and compliance challenges. To mitigate these challenges, consumers access services through a trusted base. The offerings of public cloud versions over a private cloud, while presenting some flexibility and cost benefits, introduces security challenges associated with the public cloud.

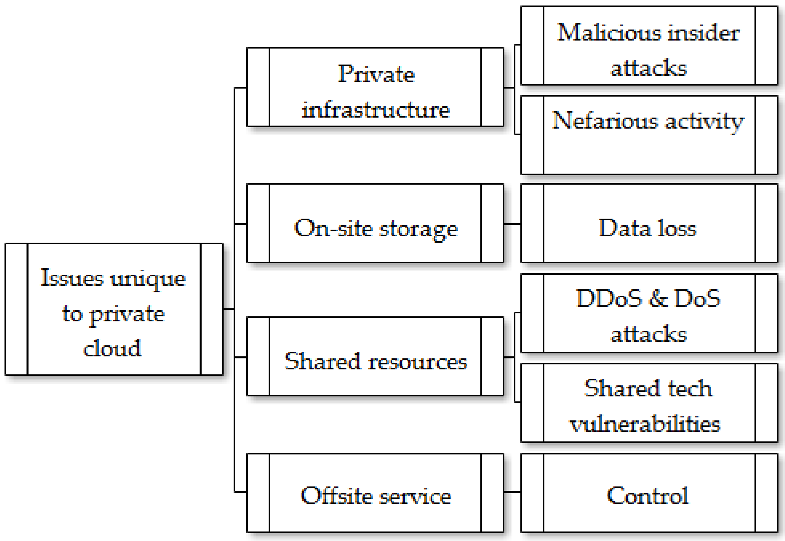


Figure 15. Security challenges that are unique to private cloud produced by the author of this thesis.

Figure 16 shows the security issues unique to public clouds. In the public cloud scenario, the infrastructure is managed and owned by a 3*rd* party and located off-site, in which consumers access services through an untrusted base. Public clouds are deemed as financially viable; cheaper model option compared to its alternatives. In addition, as a cloud subset, public clouds benefits from a shared pool of resources. Physical security risks in SaaS are greater in the public cloud (Subashini & Kavitha, 2011). Dependence on an internet connection to access off-premise infrastructure introduces risks to networks: for instance, attacks associated with the traditional security model such as man-in-the-middle and DoS attacks, threaten the security of data in transit. While traditional security systems such as firewalls, IDS and IPS are well-established and mature (Gonzalez *et al.*, 2012), efforts for developing similarly effective firewalling and filtering systems for public cloud is an ongoing process. Consequently, control, trust and insurance issues exist as consumers expect their data to remain secure, and providers give assurances of the security of their services.

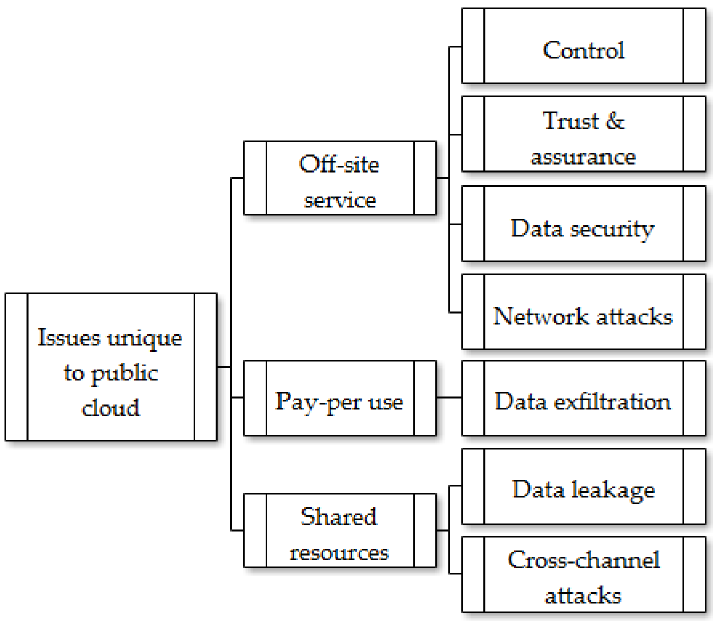


Figure 16. Security challenges that are unique to public cloud produced by the author of this thesis.

Figure 17 illustrates unique security challenges specific to hybrid clouds, where model infrastructures are managed and owned by both the organisation and a 3*rd* party. In this model, services are located both on-premises and off-premises. The hybrid option is highly dynamic as it benefits from the flexibility and scalability of the public cloud, and the efficiency and security of an on-premise model. Multiple external integrated components yield the hybrid nature of this cloud. However, performance, security and reliability procedures rely on the strength of the integrated services.

Integration of services from the private and public options means that multiple platforms are in operation, which results in compliance and insurance issues, particularly when dealing with SLA and other process and regulatory issues. While heterogeneity aids in evading challenges associated with platform lock-in, the same concept introduces integration challenges, i.e. security integration on a multi-vendor platform (Takabi, Joshi & Ahn, 2010).

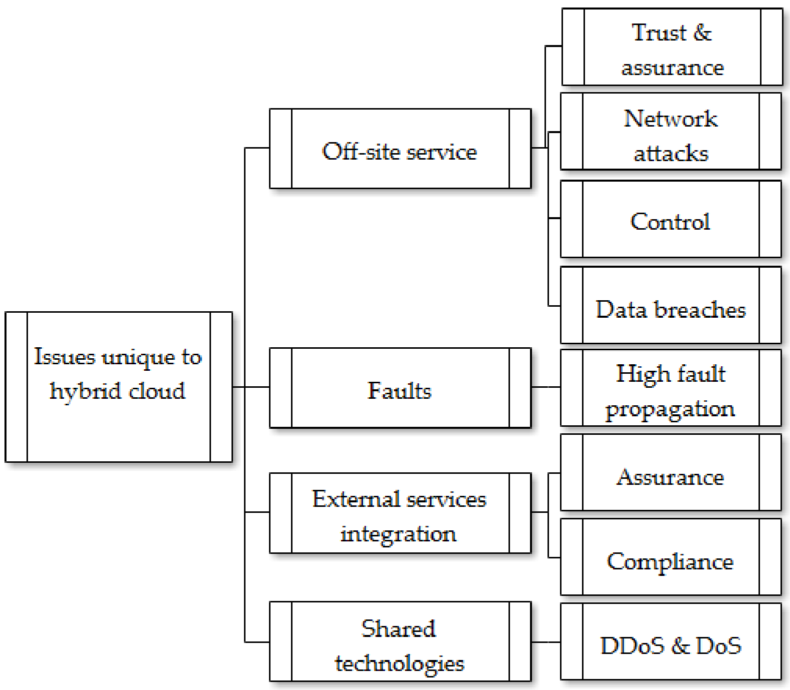


Figure 17. Security challenges that are unique to hybrid cloud produced by the author of this thesis

Figure 18 shows security challenges unique to service provision. Since virtualisation facilitates the provision of resources as services, security at the service level critical.

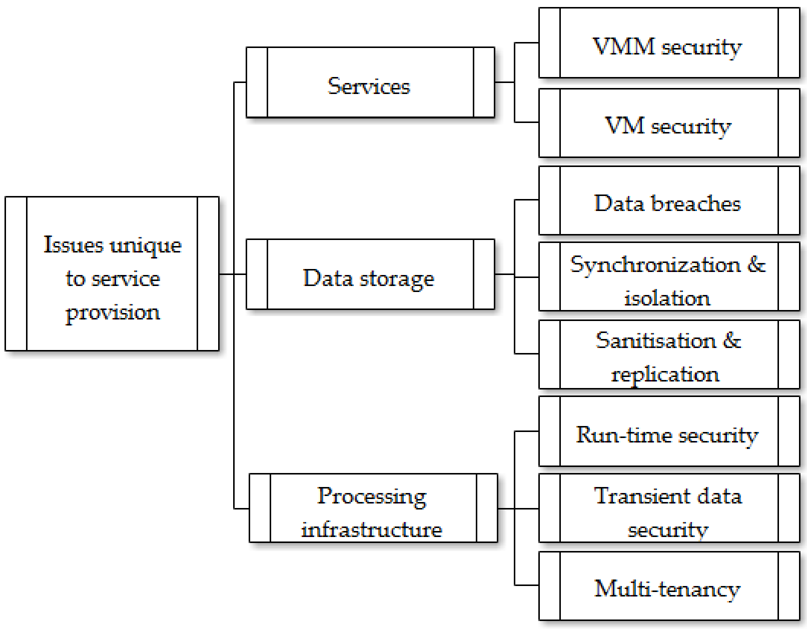

Issues unique to service provision
Services
VMM security
VM security
Data storage
Data breaches
Synchronization & isolation
Sanitisation & replication
Processing infrastructure
Run-time security
Transient data security
Multi-tenancy

Figure 18. Security challenges that are unique to service provision produced by the author of this thesis

This includes the security of the hypervisor and the virtual machine since services (SaaS, PaaS, IaaS, etc.) reside in middleware. PaaS impacts both the cloud user and clouds provider at the runtime of applications and deployment of applications. One of the contentious issues on cloud security pertains to cloud-stored data where an end-user who owns data has limited control or knowledge of where their data is stored. Hence, cloud service providers implement homomorphic encryption algorithms with distributed verification to mitigate storage security concerns including sanitization, availability, leakage, snooping, etc. One of the common challenges facing service provision is establishing trust where tenants share the same physical space, they have a secure and dedicated space. A trusted third-party solution is generally introduced but trust nonetheless remains an outstanding and unique issue in the infrastructure level of service provision. Consequently, the security of transient data becomes critical considering the prevalence of data hijacking (Baars & Spruit, 2012); (Demchenko *et al.*, 2011).

Figure 19 below illustrates unique cloud security challenges according to (Ryan, 2013). This figure is a conceptualisation of the classification as understood by the author of this thesis. These author view security issues from a perspective of their uniqueness to the cloud in contrast to their existence in the pre-cloud era.

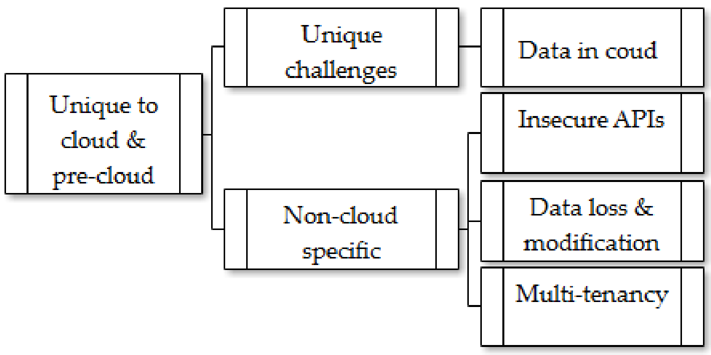


Figure 19. Security challenges from the perspective of uniqueness to the cloud and their existence pre-cloud produced by the author of this thesis

Their classification identifies multi-tenancy issues as a traditional challenge, suggesting the notion that the multi-tenancy concept predates cloud computing and has existing solutions and hence no longer be a scientific challenge (Ryan, 2013). To substantiate this claim, the author identifies the operating system (OS) process isolation and the VMM as mature multi-tenant environment management solutions. Moreover, the author identifies threats to cloud resident data as the only cloud-specific challenge due to the involvement of third parties such as the cloud provider, its employees and sub-contractors. In its current form, this is a simple classification which is easy to understand. This perspective demonstrates the relationship between cloud entities and the commonalities that exist with respect to data stored in the cloud. While this classification highlights important areas for cloud resident data, this perspective limited as it narrowly focuses on one perspective. Thus, this perspective is non-exhaustiveness as it misses key cloud components and issues around the VMM, trust, control, legal, process and regulatory issues.

Figure 20 illustrates cloud security challenges from an outsourcing perspective identified by (Shahzad, 2014). This figure is a conceptualisation of the classification as understood by the author of this thesis. In their state-of-the-art survey, the authors identify the outsourcing element in cloud computing environments as a unique to cloud computing and thus a unique source of security concern. According to these authors, an outsourcing perspective is critical since cloud service providers retain control of data. Along similar lines, the authors in (Dorey & Leite, 2011) concur and specifically identify unique risks introduced by 3rd parties through end-to-end interactions.

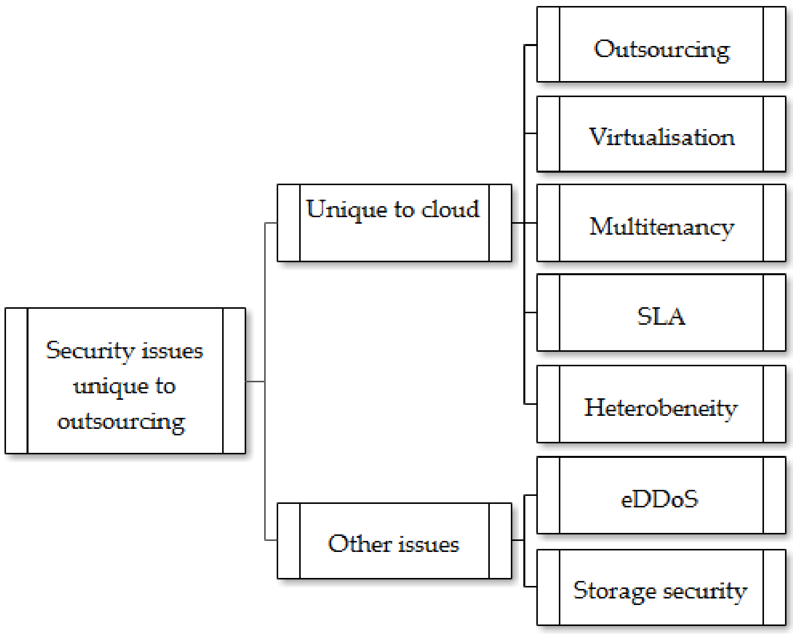
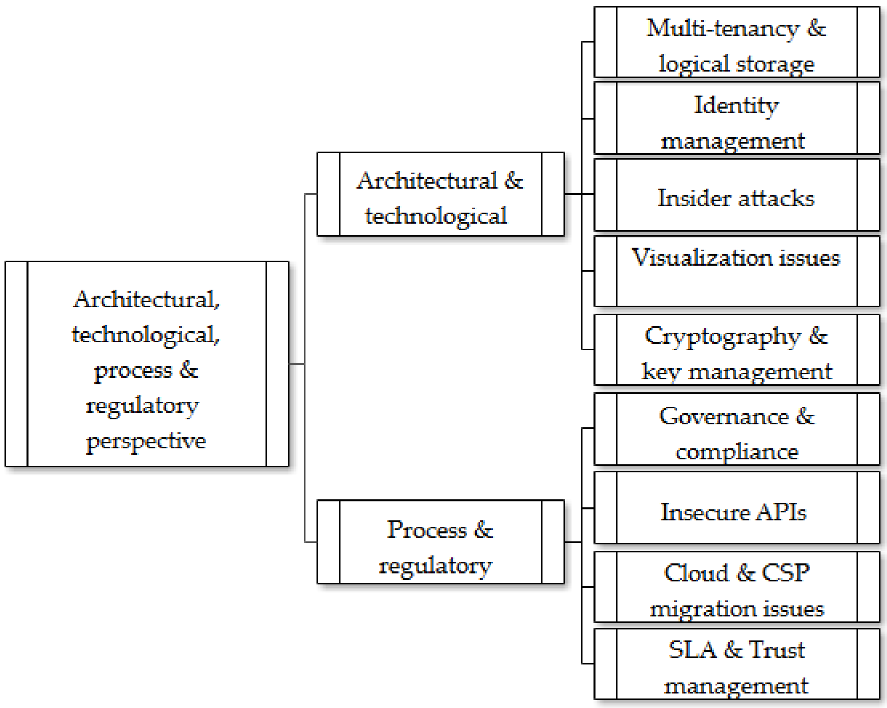


Figure 20. Security challenges from an outsourcing perspective produced by the author of this thesis

Unlike the works by (Ryan, 2013), the authors in (Shahzad, 2014) also consider multi-tenancy, virtualisation and service level agreements (SLA), as unique to the outsourcing perspective with eDDoS attacks and cloud storage data security as some of the main issues of concern. These authors share a common view that security challenges affecting the cloud are either unique to the cloud or traditionally existed. However, this is broadly too general a view of security issues in an otherwise complex environment. As noted by (Ali, Khan & Vasilakos, 2015), this complexity builds over time as cloud system entities interact and internal organisation changes. Hence, the taxonomy from an outsourcing perspective in its current form lacks the necessary complexity for general use. It is difficult to identify the boundaries upon which (Shahzad, 2014)'s observed security issues impact the cloud, which layer of the cloud, at what level, etc. Nonetheless, due to its simplicity it is easy to understand and perhaps useful when applied to address or understand specific outsourcing scenarios.

Figure 21 illustrates security challenges according to the state-of-the-art analysis by (Srinivasan *et al.*, 2012). This figure is a conceptualisation of the classification as understood by the author of this thesis. Srinivasan et al. (2012) distinguish cloud security challenges according to the architecture, technology, process and regulation perspectives of cloud environments.



Architectural, technological, process & regulatory perspective
Architectural & technological
Process & regulatory
Multi-tenancy & logical storage
Identity management
Insider attacks
Visualization issues
Cryptography & key management
Governance & compliance
Insecure APIs
Cloud & CSP migration issues
SLA & Trust management

Figure 21. Security challenges from an architectural, technological, process and regulatory perspective produced by the author of this thesis

In this regard, they identify two main categories; challenges that have architectural and technological aspects and those that are process and regulation oriented. Architectural challenges in this regard encompass multi-tenancy, identity management, virtualisation, cryptography and key management issues. On the other hand, process and regulatory issues include governance and compliance, API security, migration issues, SLA and trust management challenges. The current author finds the simplicity of (Srinivasan *et al.*, 2012)'s classification as beneficial as it enhances the taxonomy's usability. While this taxonomy broadly incorporates most of the security issues discussed in the literature and identified in this research, the omission of traditional issues limits the scope of this taxonomy. As demonstrated earlier in this research and indeed throughout available literature, traditional challenges such as the DoS, malware, malicious insider, etc. remain a threat to confidentiality, integrity and privacy in the cloud. (Cser, 2016) suggests that as traditional security strategies become outdated and ineffective in the cloud, enhanced behavioural and malware detection, preventative strategies and effective security ecosystems among security providers will ultimately deliver adequate security.

Figure 22 illustrates the classification of security challenges according to a survey by (Rong, Nguyen & Jaatun, 2013a). This figure is a conceptualisation of the classification as understood by the author of this thesis. Rong et al.(2013a) classifies cloud security challenges into traditional and cloud categories.

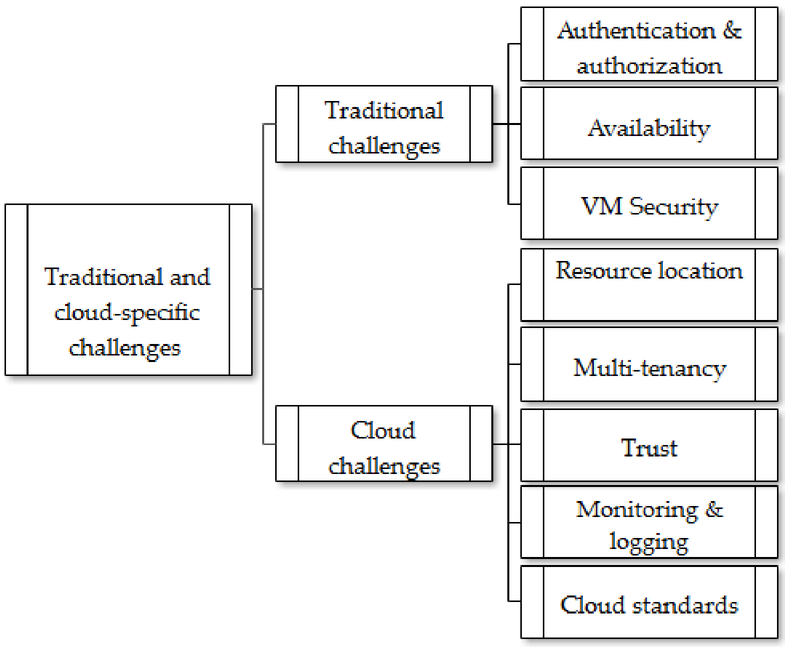


Figure 22. Security challenges from a traditional and cloud-specific perspective produced by the author of this thesis

According to the authors, traditional challenges describes those issues common to traditional communication systems such as availability, while cloud challenges described those that are uniquely introduced by the cloud. These include resource location, multi-tenancy, trust, monitoring and logging and cloud standardization (Rong, Nguyen & Jaatun, 2013a). The simplicity of (Almorsy, Grundy & Müller, 2016)'s classification makes it easy to understand, but nonetheless with limited usability across the entire cloud system. Remarkably, the authors view VM security as a traditional challenge whose solutions is available, despite clear evidence in current literature that VM security is indeed a cloud security factor that is absent in traditional systems (Almorsy, Grundy & Müller, 2016). This research posits challenges in VM security are unique to the cloud due to the operational dependencies between cloud models. This notion is articulated by (Ali, Khan & Vasilakos, 2015) who distinguish operational dependencies in the virtual layer; Software developers (SaaS) need a platform (PaaS) (Ali, Khan & Vasilakos, 2015). One may conclude that since the outsourcing element is a core concept of the cloud, 3rd parties will introduce trust and insurance challenges, which are unique to the cloud.

Figure 23 illustrates the classification of security challenges according to a co-residency perspective postulated by (Roy *et al.*, 2015). This figure is a conceptualisation of the classification as understood by the author of this thesis. (Roy *et al.*, 2015) classify cloud security issues in the infrastructure, data, communication and external services integration perspective. Co-residency attacks breach confidentiality when a malicious insider manipulates multi-tenancy to compromise other tenants on the same physical infrastructure. Each of the four categories is further divided into a security area sub-category, up to security issues (e.g. hypervisor & trusted computing base as sub-divisions of infrastructure security, resulting in integrity, confidentiality and availability issues). This research finds this classification useful as it highlights specific attacks in cloud computing. This classification is also simple and easy to understand, but nonetheless not exhaustive in its presentation of security challenges. For example, while an insecure trusted computing base is a risk to confidentiality, integrity and availability, it consequentially introduces issues of trust, privacy etc.

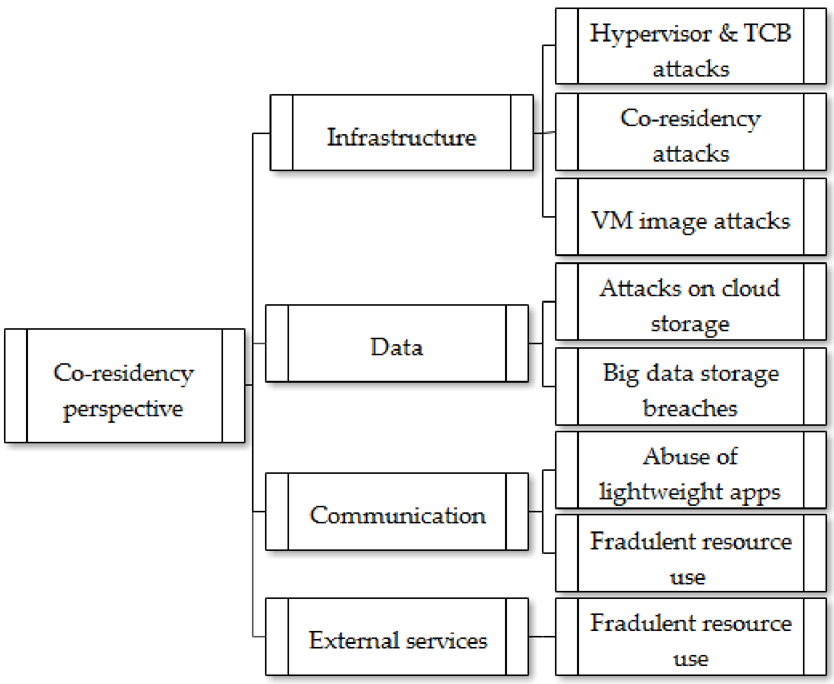


Figure 23. Security challenges from a co-residency perspective produced by the author of this thesis

Figure 24 illustrates a three-tier cloud layer classification by (Chraibi, Harroud & Maach, 2013). This figure is a conceptualisation of the classification as understood by the author of this thesis. In their work, (Chraibi, Harroud & Maach, 2013) view security issues in cloud environments pursuant to the level they belong. Thus, the authors view challenges from six levels; hardware, VM manager, guest OS, applications, network and governance. At the hardware level, the authors suggest availability, integrity, privacy and accountability as pertinent security issues. Similar security concerns exist at the application level, and all but accountability remain pertinent with network and governance. A guest operating system (OS) can access resources in the VMM by compromising insecure design, thereby compromising the integrity and privacy of that VMM and its tenants (Srinivasan *et al.*, 2012).

This is a useful taxonomy as it includes some of the core concerns found during the current research. However, this taxonomy is limited only to general use due to the authors' high-level outlook on security challenges. Moreover, this research finds this taxonomy lacking details, for instance, whether the security challenge exists internal or external of the cloud. For instance, machines are hardware, and faults exist on the server-side or user-side. From a cloud perspective, faults on the user-side do not necessarily cause the same security concerns as faults on the server-side, this taxonomy would benefit from further detailing.

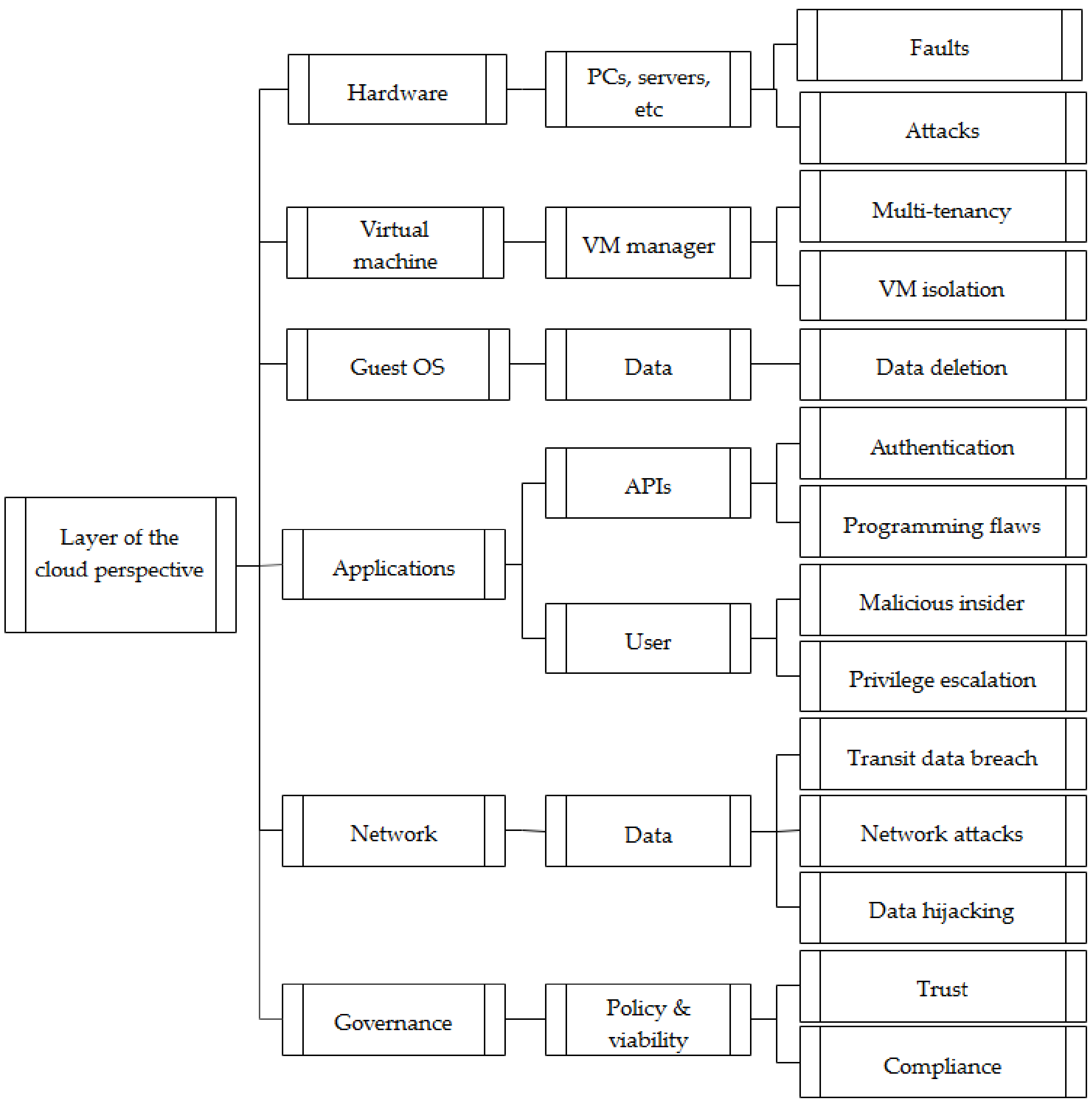


Figure 24. Security challenges from the perspective of the layer of cloud produced by the author of this thesis

Figure 25 below illustrates a cloud development perspective to cloud security challenges according to (Singh, Jeong & Park, 2016). This figure is a conceptualisation of the classification as understood by the author of this thesis. (Singh, Jeong & Park, 2016) view the reliance on web services and technologies, along with software applications and development languages as central areas which introduce security vulnerabilities. Hence, this research infers this classification as suggesting a development perspective. Along with other security issues illustrated in the figure below, this taxonomy highlights security challenges around metadata and web services.

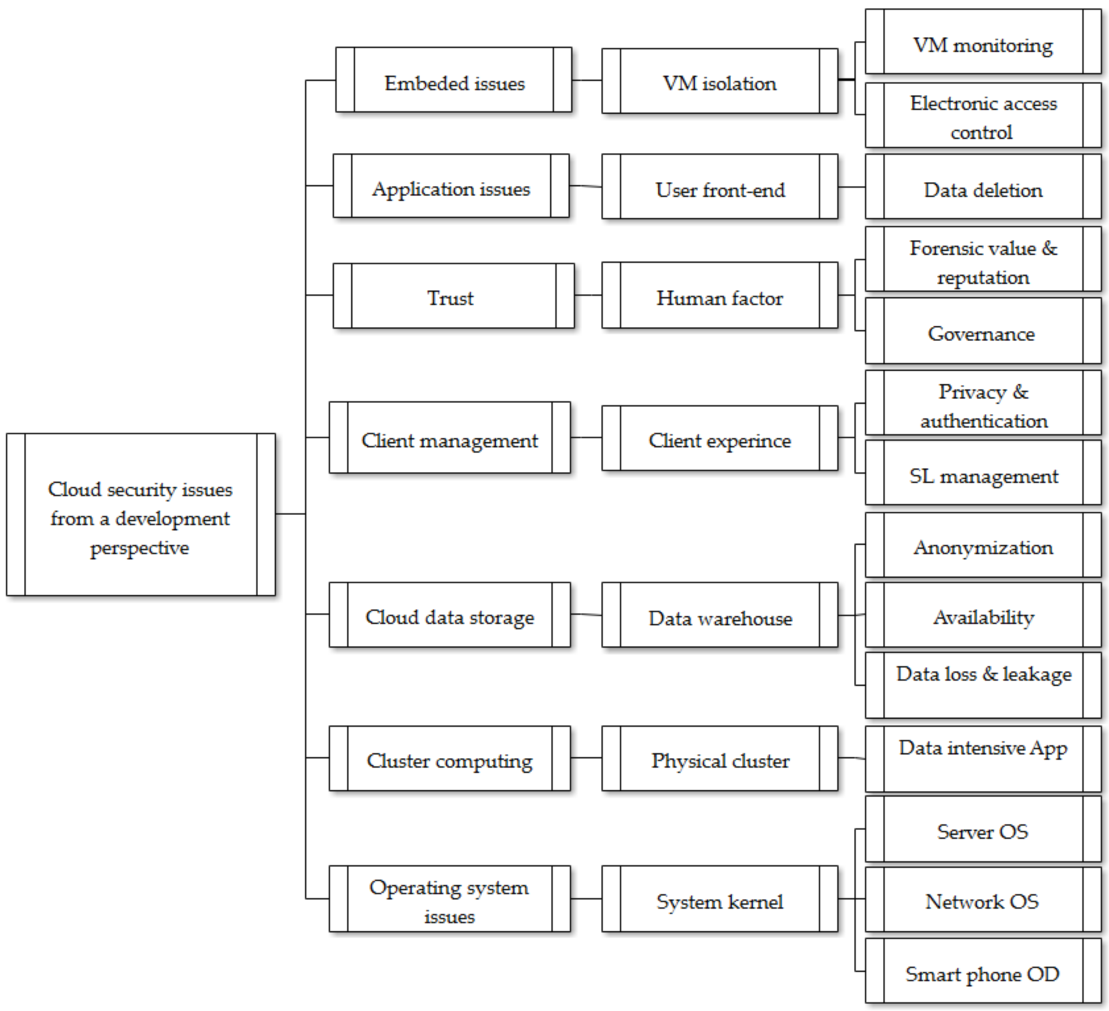


Figure 25. Cloud security challenges from a development perspective produced by the author of this thesis

Web services (Web 2.0) is suggested to introduce inherited security issues (Singh, Jeong & Park, 2016), a line of argument similar to traditional issues posited by (Rong, Nguyen & Jaatun, 2013a). Metadata, on the other hand, is highlighted as it contains data about cloud-stored data. In their view, sanitization, maintenance, separation and location protection, therefore, become the most critical challenge. While this taxonomy highlights critical security issues, particularly those relevant to a development perspective, there is no clear demonstration of complete consideration for other related entities. For instance, the role of the cloud broker as shown by (Bohn *et al.*, 2011) aims to ease the task of managing complex service integration through intermediation, aggregation and arbitration. The broker's role which entails monitoring and managing security policies between cloud consumer and cloud provider is of immense significance to cloud computing in general, and specifically to a development perspective. Moreover, where many different development languages may introduce potential vulnerabilities (Singh, Jeong & Park, 2016), the cloud broker's intermediation role is significant.

Figure 26 illustrates a classification of cloud security challenges presented in survey research by (Ali, Khan & Vasilakos, 2015). This figure is a conceptualisation of the classification as understood by the author of this thesis.

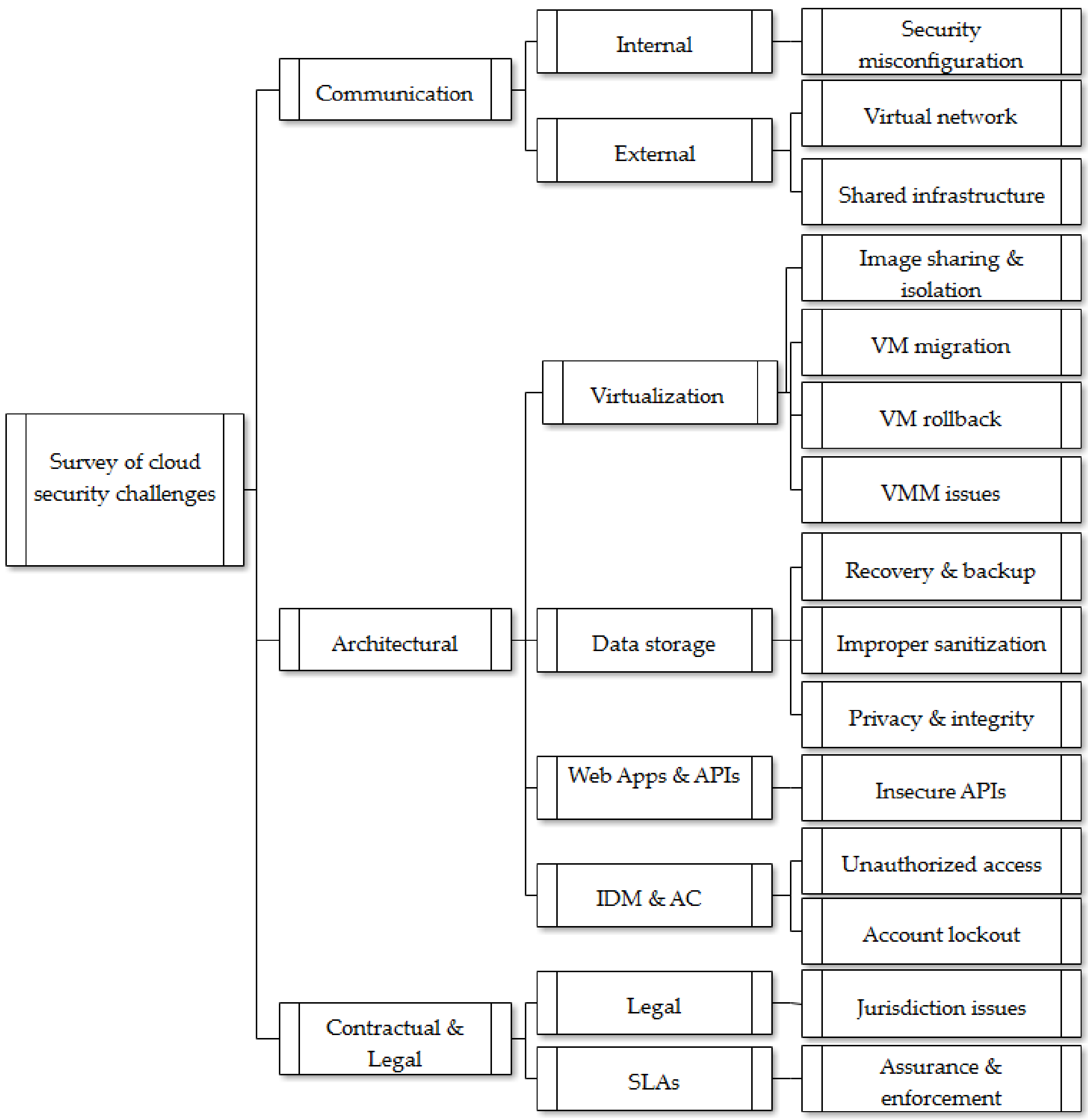


Figure 26. Cloud security challenges from a development perspective produced by the author of this thesis

(Ali, Khan & Vasilakos, 2015) classify security challenges according to the following: resources pooling, underlying technologies such as virtualisation, and the operational dependencies of cloud services on the cloud infrastructure. They identify external communicational challenges due to the significance of the internet for communication between the cloud provider and the customer. Internal challenges however, are viewed as those interactional issues that arise because of cloud characteristics and technologies, for instance, between virtual machines at a network level and another shared communication infrastructure Architectural challenges, on the other hand include virtualisation issues associated with a shared virtual environment in multi-tenant setup, for instant, VM isolation. Furthermore, they view data storage issues, insecure APIs and web applications and identity management, as architectural challenges. Contractual and legal challenges include agreements, i.e. enforcement and insurance, between the CSP and consumer, including performance insurance and consequences in times of breach. Moreover, legal challenges also encompass issues around discrepancies in the application of the law due to physical locations and/or jurisdiction.

Whereas this taxonomy comprehensively covers a wide range of cloud security concerns, it fails to incorporate the outsourcing components of cloud computing. This is a critical aspect of the cloud since 3rd parties form a core component of the cloud ecosystem. As noted by (Dorey & Leite, 2011), associated risks, control, supplier sustainability and integrity are some of the critical concerns to the cloud computing architecture. These among others, demonstrate the deficiency of (Ali, Khan & Vasilakos, 2015)'s perspectives to cloud computing security.

Figure 27 illustrates this research's conceptualisation of security challenges from architectural complexities perspectives according to (Dorey & Leite, 2011). According to these authors, identity management, data security and trust & assurance are priority areas for security improvements (Dorey & Leite, 2011). The authors argue that cloud security challenges exist due to the architectural complexities and thus, trust & assurance, identity management and data security as priority areas for security improvements in cloud environments. Moreover, they propose that cloud security challenges exist due to the architectural complexities within the cloud environment.

It is the opinion of the author of this thesis that multi-tenancy, control issues related to third parties and security issues inherent to the cloud architecture such as monitoring, and auditing are equally critical to security in cloud computing. Data storage concerns identified here resemble a data life-cycle perspective motivated by the authors in (Gonzalez *et al.*, 2012). According to these authors, privacy focuses on the complete data life cycle, from the point when data is generated through to its destruction, while legal issues pertain privacy relate to guidelines in the cloud. The authors define a compliance category relating to governance and control issues. Furthermore, they define the architecture category to include elements such as the network, interfaces and the virtualisation issues (Isolation, hypervisor, data leaks, and VM identification and cross VM attacks) arising due to cloud the architecture and infrastructural implementations. According to the authors, other challenges include decision-making, user access and authentication. This taxonomy is deemed as limited as it does not include some critical security challenges

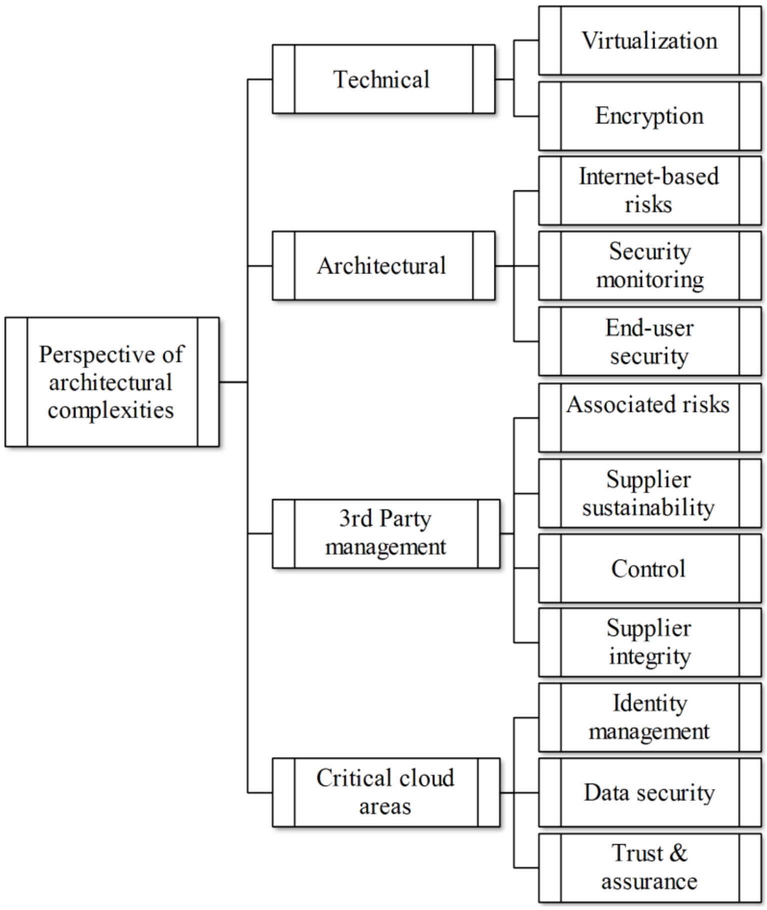


Figure 27. Cloud security challenges from the perspective of architectural complexities produced by the author of this thesis

Figure 28 illustrates cloud security challenges from an organisation's outsourcing components perspective according to (Jansen, 2011). This figure is a conceptualisation of the classification as understood by the author of this thesis. Jansen (2011) classifies cloud security issues concerning the outsourcing portions of organisations as trust, architecture, software isolation, identity management, data protection and availability. The author postulates that loss of direct control of the security aspects in outsourced environments introduces risks as it gives the cloud service provider "unprecedented levels of trust" to (Jansen, 2011). Likewise, risk management is a challenge in an outsourced environment, as organisations possess limited control to prepare for incidents, setting priorities and contingency plans.

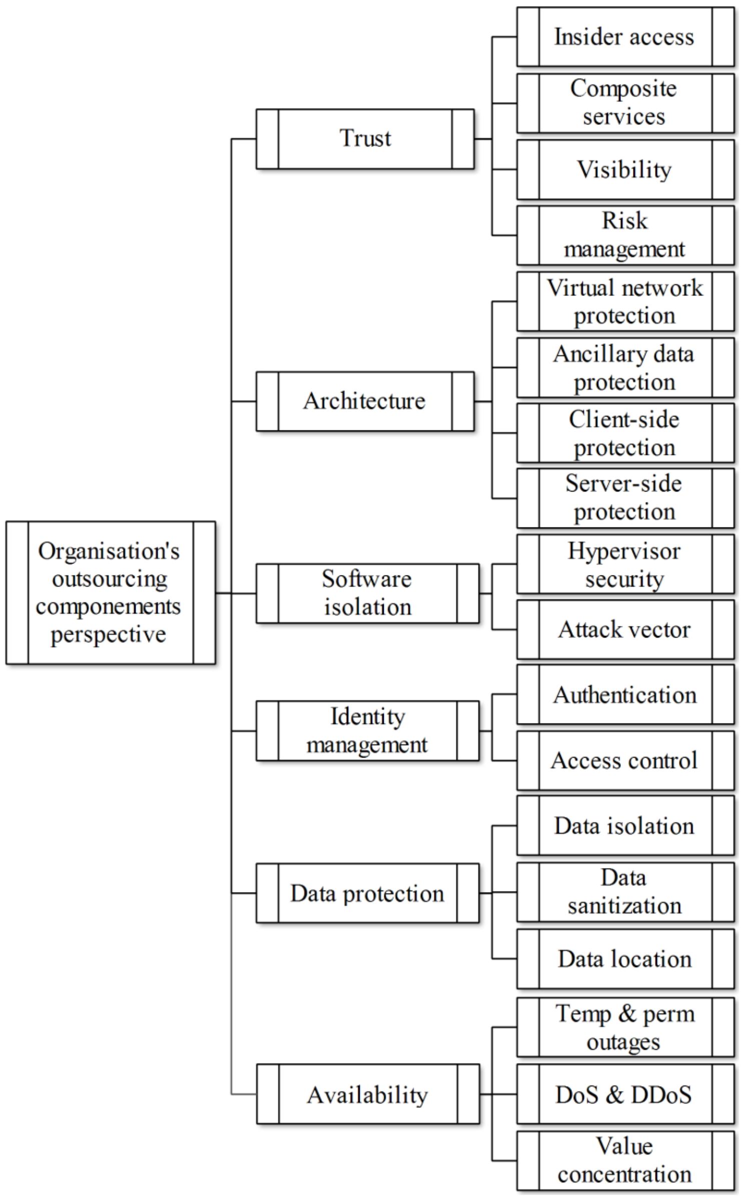


Figure 28. Security challenges from an organisation's outsourcing components perspective produced by the author of this thesis

This classification is deemed to be useful and insightful. For instance, outages as a critical cloud security challenge to availability. In addition, this thesis considers architectural challenges including client-side security to be a valuable consideration; cloud interactions begin at the client workstation, through the internet to the cloud. However, this classification could benefit from illustrating wider coverage, for instance, considering trust issues pertinent to the consumer vs trust issues pertinent to a 3rd party such as a sub-contractor. Moreover, it may also consider issues that are a result of implementations in cloud infrastructure; the physical and organisational structures and facilities that constitute the cloud (Liu *et al.*, 2011). For instance, where cloud consumers utilise SaaS applications to process data, often without the knowledge of how data is processed or where it resides, compliance and privacy are some of the main security issues that arise (Subashini & Kavitha, 2011). From jurisdiction perspective, a range of legal challenges including e-discovery become prominent across jurisdictional boundaries. From a technical perspective, virtualisation introduces vulnerabilities associated with isolating VMs on shared physical infrastructure, hypervisor vulnerabilities that may result in data leakages, malicious attempts to sniff traffic, compromised cryptographic keys and other confidential data (Subashini & Kavitha, 2011).

Typically, one of cloud computing's main advantages for an organisation is the reduction in management and maintenance costs associated with computing infrastructure, including the security. This implies that trust is assumed as shared between the cloud provider and the consumer (Sun *et al.*, 2011). However, considering vulnerabilities due to multi-tenancy, virtualisation, 3rd parties and/or shared infrastructures, trust concerns arise due to the subjective, context-based, imprecise and often transitive nature of trust (Sun *et al.*, 2011). As noted by the authors in (Zissis & Lekkas, 2012), depending on the cloud model, trust tends to be an obscure property due to loss in the governance of data and applications associated with outsourcing services.

Figure 29 illustrates this research's interpretation of cloud security challenges from the end-users perspective identified by (Zissis & Lekkas, 2012). These authors classify security challenges according to an end user's concerns of security issues at the application, virtual and physical layers of the cloud. They suggest a logical notion of the end-user as broadly encompassed in a cloud subscriber, software developer and the person or organisations who owns a cloud infrastructure. According to these authors, an end-user-centric proposition presents a compelling argument for a Trusted Third Party (TTP), as a solution to trust issues at a horizontal level of service (Zissis & Lekkas, 2012).

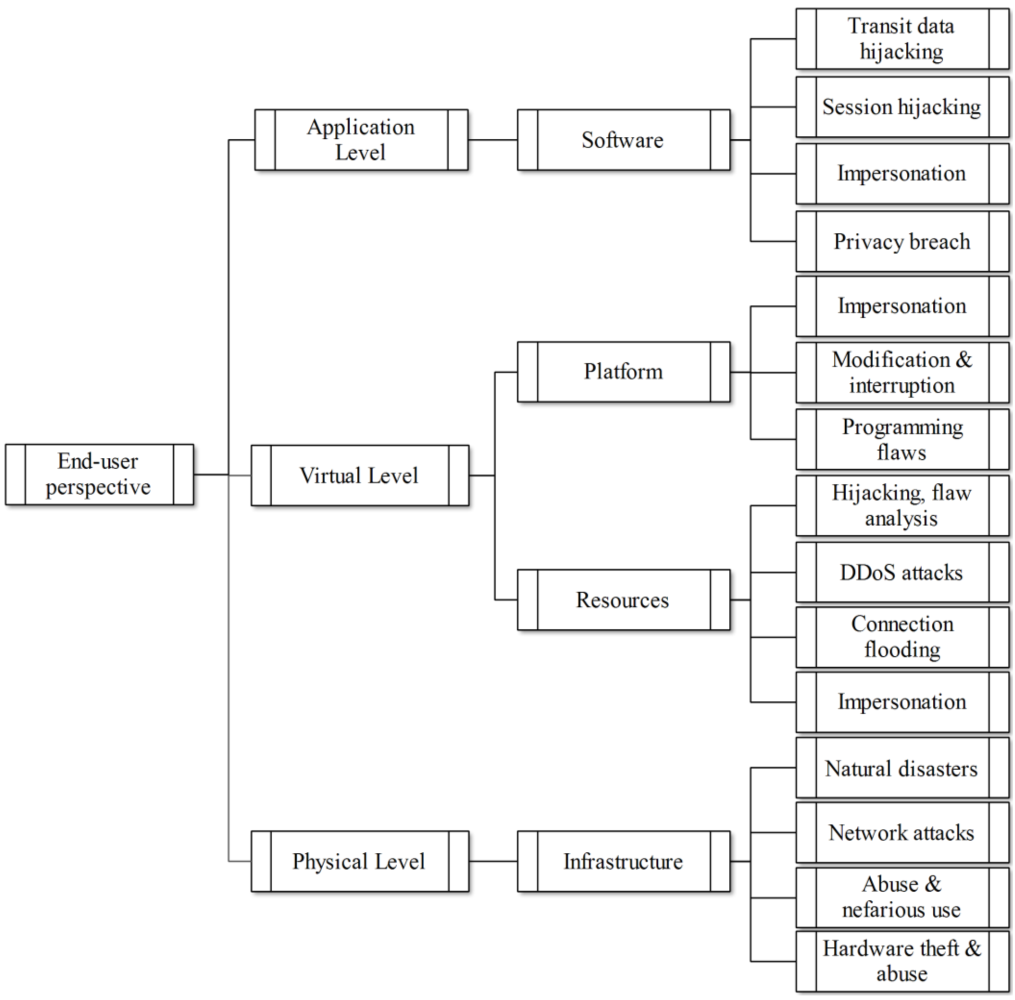


Figure 29. Cloud security challenges from the end-users perspective produced by the author of this thesis

This classification offers a useful technical proposition towards addressing one of the most prevalent security challenges in the cloud. More so when considering the importance of the trust element when integrating cloud computing with other technologies such as the IoT (Botta *et al.*, 2016)(Díaz, Martín & Rubio, 2016).

Nonetheless, this perspective suffers from the common deficiency of single-perspective approaches, i.e. it is only limited to local concerns for the end-user and perhaps fails to address the wider security challenges of other dissimilar perspectives. One major drawback, as a result, pertains to developing countermeasures. Applying (Zissis & Lekkas, 2012)'s end-user perspective implies that only clearly identified local security concerns are focused upon, and thus countermeasures predominantly offer narrow end-user-centric solutions. In fact, this taxonomy does not consider the top-down, bottom-up or left-to-right view, and hence excludes other dimensions of the complete sense of the cloud that clearly constitute areas of concern, e.g. other cloud entities such as 3rd parties as noted in (Shahzad, 2014).

3.4 Proposed holistic taxonomy

The literature survey shows that existing taxonomies are not generalisable across the entire cloud security domain. Figure 30 below clearly demonstrates this multiplicity of perspectives to what constitutes cloud security challenges. While other perspectives are considerably detailed, their complexity introduces some level of ambiguity. The foregoing gives the notion that while existing perspective-driven taxonomies may be simple and useful, representing specific perspectives of the cloud security domain, they remain solely limited to that perspective and therefore fail to identify and communicate security challenges from a holistic view. Arguably, this limitation also perpetuates cloud computing's current security dilemma where solutions are designed for specific threats.

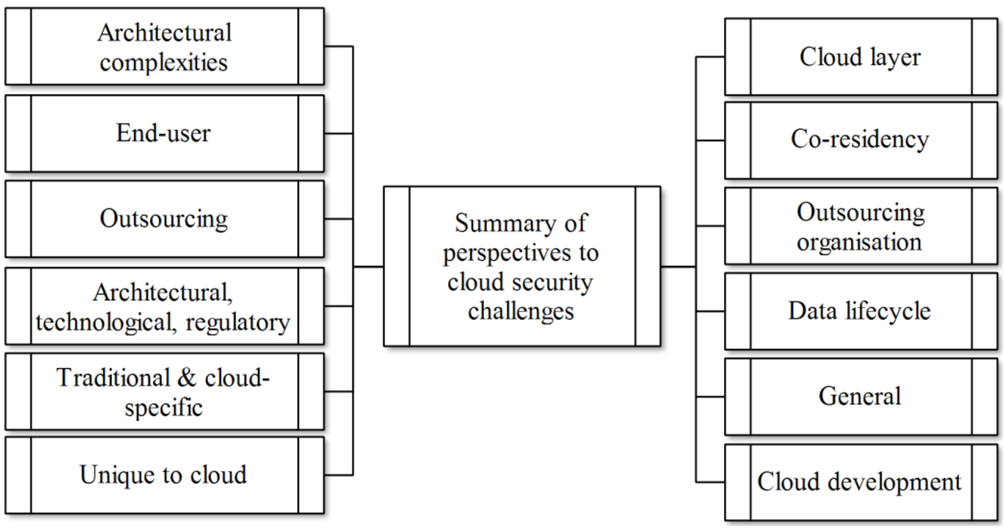

Architectural complexities
End-user
Outsourcing
Architectural, technological, regulatory
Traditional & cloud-specific
Unique to cloud
Summary of perspectives to cloud security challenges
Cloud layer
Co-residency
Outsourcing organisation
Data lifecycle
General
Cloud development

Figure 30. Multiple perspectives to cloud computing security challenges produced by the author of this thesis

In addition, the perspective aspect also implies that current taxonomies possess inherent weaknesses such as ambiguity, non-exclusivity, etc. Thus, the current section proposes a holistic assertion that incorporates the entire cloud paradigm, regardless of perspective. Moreover, the holistic notion proposed further shows and supports that cloud security challenges can be classified based primarily on the source/origin of the security incident rather than a perspective.

Table 9. A summary of cloud security topical areas.

	Author										
	(Dorey & Leite, 2011)	(Jansen, 2011)	(Zissis & Lekkas, 2012)	(Gonzalez *et al.*, 2012)	(Srinivasan *et al.*, 2012)	(Rong, Nguyen & Jaatun, 2013a)	(Ryan, 2013)	(Chraibi, Harroud & Maach, 2013)	(Shahzad, 2014)	(Ali, Khan & Vasilakos, 2015)	(Khalil, Khreishah & Azeem, 2014)
Tenancy	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Cloud specific	✓	×	×	×	✓	✓	×	×	✓	×	×
Traditional	×	×	×	×	✓	✓	×	×	✓	×	×
Compliance	×	×	×	✓	×	×	×	×	×	×	×
Physical	×	×	✓	✓	×	×	×	✓	×	×	✓
App/software	×	×	✓	×	×	×	×	✓	×	✓	✓
Network	×	×	✓	✓	×	×	×	✓	×	×	×
Availability	×	✓	×	×	✓	×	×	×	✓	×	✓
Architecture	✓	✓	×	✓	×	×	×	×	×	✓	✓
Governance	×	×	×	×	×	×	×	✓	×	×	×
Process	×	×	×	×	×	×	×	✓	×	×	×
Infrastructure	×	×	×	×	×	×	✓	×	×	×	✓
VM manager	×	×	×	×	✓	×	×	✓	×	×	✓
Data security	✓	✓	×	×	×	×	✓	✓	✓	✓	✓
Communication	×	×	×	×	×	✓	✓	×	×	✓	✓
External services	✓	×	×	×	✓	×	✓	×	×	✓	×
3rd parties	✓	×	✓	×	×	×	✓	×	×	×	×
Technical	✓	×	×	×	×	×	×	×	×	×	✓
Privacy	×	×	×	✓	×	×	×	×	×	×	✓
Trust	×	✓	✓	×	✓	×	×	×	×	✓	✓
Legal & SLA	×	×	×	×	×	×	×	×	×	✓	✓
IDM	✓	✓	×	✓	×	×	×	✓	✓	✓	✓
Data Isolation	×	✓	×	✓	✓	✓	×	✓	×	✓	×
Risk	×	✓	×	×	×	×	×	×	×	×	×
Malicious insider	✓	×	✓	✓	×	×	×	✓	×	×	✓

By simple inspection, Table 9 suggest that, while some areas, e.g. external services and 3rd parties overlap, they are nonetheless diversely viewed among various perspectives. Intuitively, this points to the importance of specificity when tackling security issues. Hence, this thesis places emphasis on simplicity to be a positive attribute towards the usefulness of a taxonomy. Gap areas in current taxonomies (suggested by adjacent x) point to several unexplored security areas of interest which exist across different perspectives. In this sense, simplicity limits deeper detail and hinders the adequacy of taxonomies. Whereas the concept of the 'source' forms the basis upon which a taxonomy evolves, a comprehensive approach would inform the development of the holistic taxonomy for cloud computing security. The matrix populates earlier taxonomies against relevant areas of the cloud reviewed by authors. This is clearly not an exhaustive list.

Having analysed the limitations of existing taxonomies with respect to cloud computing's important features and related security issues, this section presents a holistic taxonomy with clearly defined categories and provides descriptions explanations of each. Unlike taxonomies in current literature, the proposed holistic taxonomy will facilitate a comprehensive analysis of security issues and the development of robust security countermeasures. As the origin/source hypothesis alludes, the root of the holistic taxonomy comprises three main categories; adopted, inherent and triggered cloud security challenges. Figure 31 is the graphical illustration of the detailed holistic taxonomy.

To capture the multiple perspectives obtained through research, vertical categories, herein named Cat 0, Cat 1, Cat 2, Cat 3, Cat 4 and Cat 5, are integrated. Cat 0 encompasses the three main categories mentioned above. While holistic notion is placed in retrospect to perspective-driven classifications discussed earlier, some contributions including, (Liu *et al.*, 2015)(Almorsy, Grundy & Müller, 2016) and (Huang & Wu, 2018) are substantial to the current proposition. Below are descriptions of the proposed categories, which according to the current author, highlight the basis for new and unique opportunities each affords to the general cloud-wide security domain.

Cat 0**:** Distinguishes security challenges according to source or their origin, in line with (Armbrust *et al.*, 2010) who suggests security issues as being both internal and external of the cloud. Knowledge of the source or origin presents a clear opportunity for applying specific security countermeasures and where necessary, tracing the source. Practically, this implies for instance, that the Same Origin Policy (SOP) can be applied at Cat 0, where monitoring of the original location of a web request, enabling validation of that requests (II & Al-Hamdani, 2011).

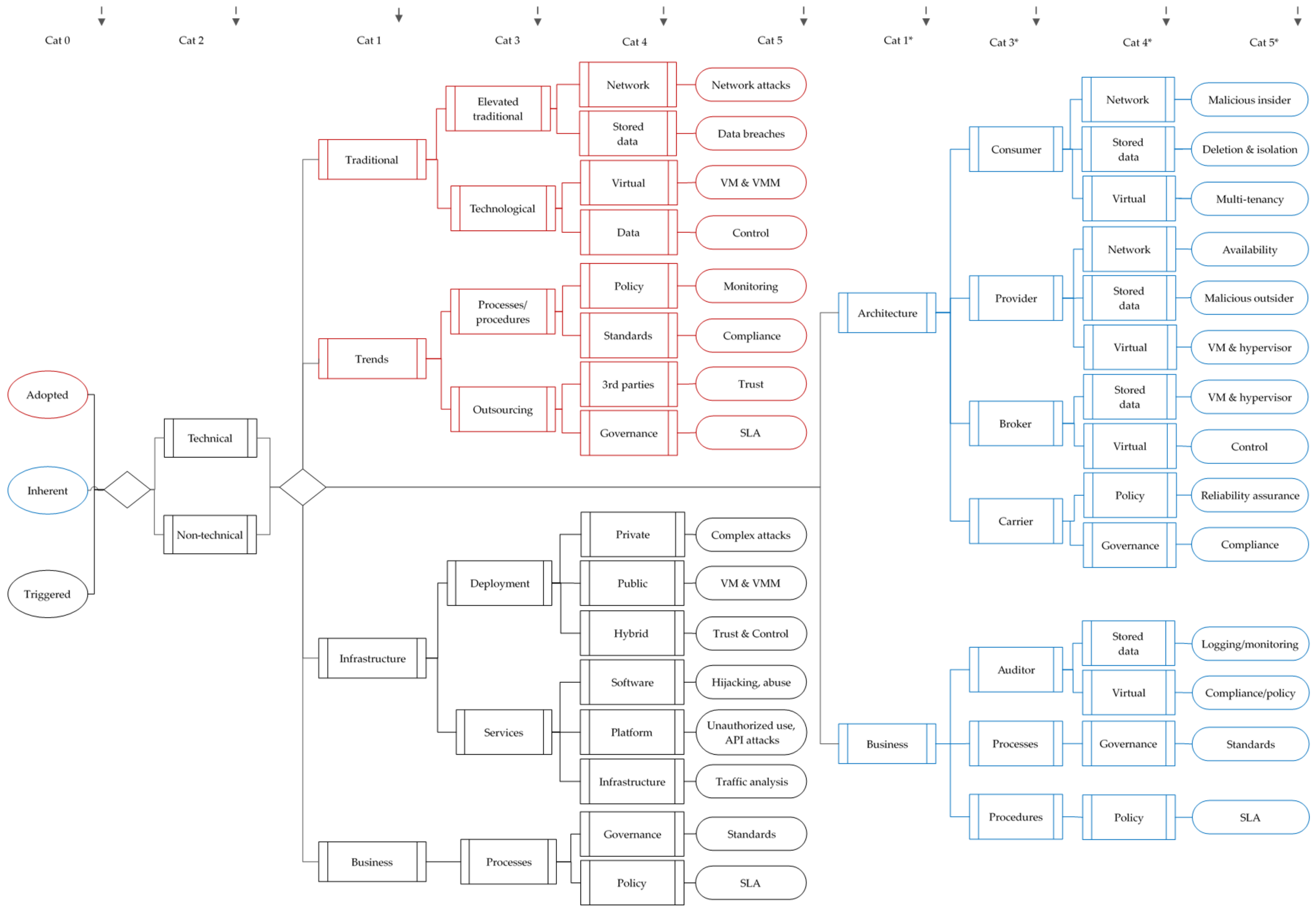


Figure 31. Holistic taxonomy of cloud computing security challenges.

The holistic taxonomy captures security challenges adopted security challenges (red) such as traditional security challenges and trends. It captures security challenges inherent to the cloud architecture (blue) such as business and architectural issues. Moreover, it captures cloud security challenges which arise from implementations in cloud environments (black).

Cat 1**:** Consists of two sub-classes that provides more detail into the nature of the challenges described in Cat 0. When considering cyberspace in general, elements of this category can be extended to capture emergent behaviours

Cat 2: Distinguishes security issues simply as technical or non-technical. Based on this categorisation, it is easier to effect efficient prioritising and decision-making, for instance, non-technical business priorities would attend to concerns such as governance and policy matters.

Cat 3: Provides detailed descriptions of the technical and non-technical aspects of Cat 2. Put into perspective, Cat 3 describes traditional security challenges (Cat 1) emanating from cloud's peripherals (Cat 0) which are technical and/or non-technical in nature (Cat 2), but only exist upon elevation in the cloud (Cat 3). As an example, when considering the architectural perspective, Cat 3 describes technical challenges pertaining to the consumer and provider and non-technical challenges pertaining to the broker, carrier and auditor.

Cat 4**: D**escribes attributes that have a direct impact upon security, including networking attributes in the traditional sense result in network attacks such as the man-in-middle or DoS. Similarly, attributes that describe data result in data breaches, etc. Virtual layer attributes result in multi-tenancy, virtual machine management and hypervisor security issues.

Cat 5: As has been illustrated throughout this paper, there are many security challenges that affect the CSP, consumer, carrier, auditor and the broker. This category identifies a spectrum of security challenges covering the entire cloud domain. This category is not exhaustive, thus, security challenges listed in this category exist for illustrative purposes.

3.5 Analysis

As noted earlier in this chapter, existing literature on classifying cloud security challenges fails to represent a comprehensive view of the cloud security continuum. Moreover, it is quite clear that no research efforts go towards consolidating existing multiple textual perspectives, for the benefit of obtaining useful value for security practitioners. By combining viewpoints of what constitutes current security concerns from academia and industry, this research substantially adds to the growing body of literature on cloud security. The holistic approach employed is unambiguous as it integrates cloud computing's many dimensions; services, entities, attributes, layers, characteristics, etc. and links all its perspectives into non-specific categories (Cat 0 — Cat 5). In this manner, each category represents a branch of potential security incidents.

The proposed holistic taxonomy comprehensively eliminates gaps introduced by perspective-driven views of the cloud, provides a basis for further research and facilitates the development of security strategies. Another clear opportunity is the possibility to enforce accountability in any cloud entity, including the enforcement of punitive or corrective measures through easier identification of the source or origin of a security incident. From an incident response point of view, the ability to identify the source and/or origin of a security incident means response teams spend more time implementing mitigation measures, rather than trying to identify the incident. Furthermore, the proposed holistic taxonomy highlights security boundaries, including where security systems can be placed. Thus, considering UUURs, Cat 1 of the proposed holistic taxonomy can be extended to capture emergent phenomenon and solutions. Figure 31 addresses the obtaining view that cloud security challenges are addressed in isolation. It raises an important concern for decision-making around security solutions; which is itself a critical factor in the security and requires better strategy and planning (Paquette, Jaeger & Wilson, 2010).

To enhance the security posture of cloud computing, the focus should be firmly placed on identifying and placing into context security challenges based on their source/origin foremost, before extending to the actual security incident. By considering source/origin as the basis for a holistic classification, this research highlights planning as a critical phase that is significant when designing and implementing security countermeasures. Hence, the proposed holistic taxonomy enables better planning for security design and provides a contextual illustration of the relationship between cloud computing and its peripherals for research analysis. For instance, Cat 1 enables cloud solution planning by mapping any organisation's objectives including their security requirements.

Trends such as Big Data are at the same level as business processes and procedures and the security issues in the architecture and infrastructure. This implies that in a real environment, Cat 0 and Cat 1 which highlight issues at the perimeter-level of security, enable organisations to make use of their existing security technologies. This is an important addition considering deficiencies in existing taxonomies. Furthermore, the proposed holistic taxonomy integrates Cat 3 following NIST's view of cloud's computing: cloud entities, attributes and models. Addressing security is also contingent upon how organisations are prepared to avoid or minimise the impact of insecurity. This readiness is what determines the appropriate security solutions and the degree of proactiveness that ensure that unforeseen events or black swan events are well-prepared for. Moreover, employing the most appropriate countermeasures in the cloud includes applying the correct mechanisms (tools, techniques, procedures and approaches), correctly presented requirements and adequate policy that defines the lower and up boundaries of what is allowed and what is not allowed. These approaches can be matured by applying the holistic taxonomy proposed in this research. Several techniques and approaches to secure cloud environment exist, some focusing on the technicality of achieving a security design, while others focus on the approach. For instance, CloudProtect is designed to provide privacy and confidentiality in the cloud (Ardagna *et al.*, 2015b). (Rahman & Choo, 2015) Conceptual Cloud Incident Handling model addresses security from the cost perspective of an incident handling investment, cost of incident detection and analysis, the cost of responding to an incident and post response cost.

Yet, some challenges remain open. The ability to visualise deficiencies in security solutions designs requires that visualisation techniques are revised. Evidence in the literature suggests that design outcomes are reliant upon the visualisation technique; from earlier systems with lower dimensions and numbers to complex systems with complex information (Chalmers, 2013). In the latter, is it suggested that visualisation is incomprehensible due to a large amount of information to be represented (Knight & Munro, 1999). Another challenge pertains to the process itself. Manually interpreting corpus textual data is a tedious process and prone to errors and omissions.

3.6 Summary

The work presented in this chapter aims to address the foundational aspects of the scope of this thesis, by effectively placing cloud computing security challenges into context. It is necessary after all, to explore what security issues exist, how they are communicated and understood in academia and in practice, and how this knowledge can be applied to enable the bio-inspired approach proposed. A proposed definition for a holistic taxonomy (See section 3.2.1) identifies the requirements for a holistic notion specific to cloud computing security challenges. The holistic design approach and the holistic workflow process are useful guides to the holistic taxonomy. The holistic taxonomy presented in section exposes security areas of the cloud and proposed as useful to facilitate the design of enforcement or corrective countermeasures based upon the source or origin of a security incident. The proposed taxonomy thus satisfies H1 of the research hypothesis.

Chapter 4 A TRIZ-based approach for prey-inspired cloud computing survivability

This chapter presents a TRIZ-based prey-inspired survivability design method to facilitate bio-inspired design as applied from prey survival to survivability in cloud computing. This integrates both normative and descriptive perspectives. Based upon a problem-driven approach, a 3-step process facilitates the interdomain transfer of concepts (retrieval, mapping and transfer), whereas TRIZ's generic approach suggests specific solutions for cloud computing challenges based upon systematically deduced resolutions. Hence, this chapter addresses the hypothesis with respect to the applicability of 'escalating survival behaviours and mechanisms' to enhance survivability. Since this method presented are systematic, the approach proposed here is generalisable for use in other domains. This chapter is based on contributing publications RP3, PR4 and RP6.

4.1 Introduction

Cloud computing presents a new dimension to the longstanding bio-inspired design challenge which impacts upon innovation itself, i.e. how to create innovative solutions and resolve design requirement contradiction. While several bio-inspired design approaches (Nagel & Stone, 2011a) (Vincent *et al.*, 2006) exist, there is limited empirical studies to compare them. The design problem for cloud computing concerns contradictions between survivability and security requirements and cloud environments' attributes. For instance, how to address security challenges due to the cloud's enlarged attack surface while preserving broad-network access which enables cloud computing to be accessible anywhere on any device. In this case, how to improve the design requirements in question while preserving the cloud's attractive attribute becomes the issues of concern.

Analogical design, functional modelling and other reasoning methods (Glier & McAdams, 2011) attempt to resolve this problem. However, these approaches are built on functional design tools and methods which require in-depth interdisciplinary knowledge to develop solutions (Nagel & Stone, 2011b). Bio-inspired design, for instance, relies upon a designer's understanding of the underlying biological system for efficient inter-domain transfer of information. An engineering-to-biology thesaurus (Nagel, Stone & Mcadams, 2010) thus becomes useful through direct translation or abstraction of biological terminology to engineering. Notwithstanding, bio-inspired systems such as self-managed cloud platforms (Hariri, Eltoweissy & Al-Nashif, 2011)(Ali, Robson & Boukerche, 2016) possessing the complexity attributes of inter-networked environments resting on established evolution principles of 'self' (Meisel, Pappas & Zhang, 2010) have been proposed.

Earlier work by the author of this thesis identifies a number of bio-inspired algorithms where the predator-prey system are specifically identified as vastly useful for high adaptation (Mthunzi & Benkhelifa, 2017). As noted by (Andersson *et al.*, 2009), enhancing error-free and automated survivability in dynamic, complex and unpredictable environments entails identifying what is central to adaption. Nonetheless, it is established that the use of biological systems in cloud computing security and survivability is an under-researched area (Stoykov & Yazidi, 2016). This deficiency in research forms the basis for this chapter's work to develop generalisable approaches for prioritising and combining interdisciplinary objectives in a formal manner and offer systematic solutions to the cloud computing security domain.

The remainder of this chapter is as follows: Section 4.2 - 4.3 presents the method; the problem-driven approach applied to identify a biological solution for cloud computing. The 3-step process is a systematic approach for transferring natural concepts to cloud computing, and the TRIZ method for innovative solutions. Section 4.4.1 shows the application of TRIZ. Section 4.5 presents a NetLogo pilot experiment and Section 4.6 concludes the chapter and summarises the main findings.

4.2 Problem-driven design context

The current chapter's contributions postulate that given an old problem (P^{old}) with an old solution (S^{old}), a new problem (P^{new}) can be conceptualised with new partial and null solutions (S^{new}) in the solutions space S^{old} to S^{new}. Hence, prey animals' solution against predation is proposed as core requirements for cloud computing's survivability problem. Thus, solutions are proposed as effective for cloud computing because, (1) each serves to address specific contradictions, (2) solutions exist at varying levels of abstraction. (3) The process suggests core areas for future work in this research. Hence, one of the main contributions of this chapter is the systematic method for biology-to-cloud design, to facilitate the efficient and unambiguous transfer of concept and information. Figure 32 shows the generic processes. Dotted line indicates the boundary of the taxonomy development processes, while the solid line indicates the outer boundary. Arrows indicate the direction of flow between the processes. Briefly, an input model which in the current chapter, represents the multiple contexts of security within cloud computing's security fora.

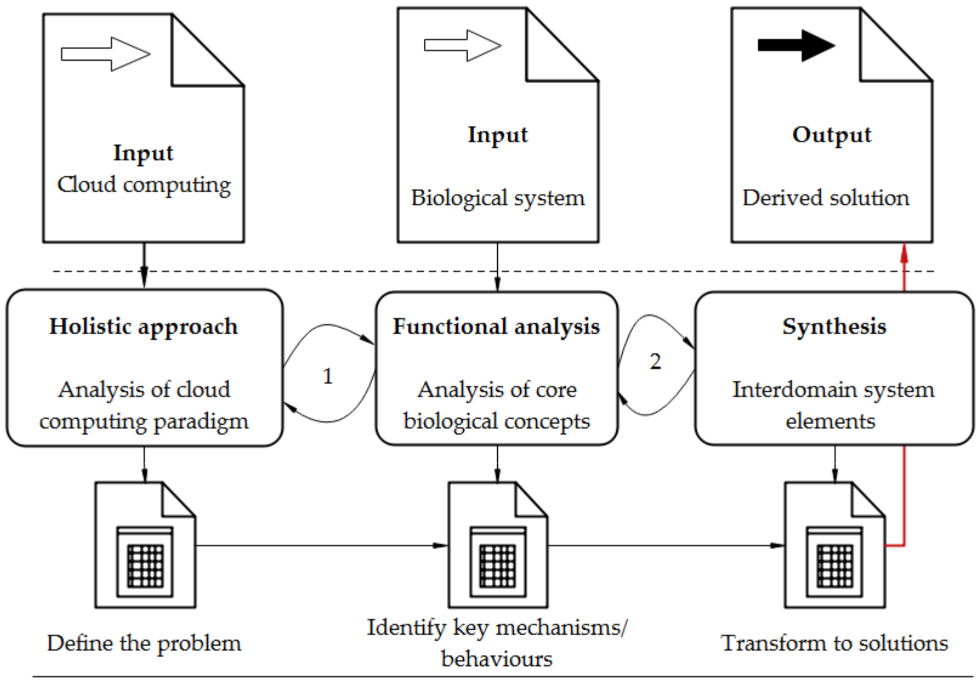


Figure 32. Holistic problem-driven approach for bio-cloud computing design produced by the author of this thesis

On the other end, the output model provides the holistic tool that specifies the security or survivability baselines and contexts. In the current chapter, this is specifically useful for information retrieval (described below). However, the output model can be used for security systems baselining, security specifications and configurations or security planning. Lower level processes comprise the taxonomy requirements loop (1) - thematic modelling of the cloud security paradigm and functional analysis which deconstructs, define and assigns categories to concepts. A design loop (2) models the continuous processes of the functional analysis and the synthesis; to functional concepts to those that resemble the real physical cloud environment. This useful as it facilitates the definition and extension of additional security concepts and elements (these are represented as categories, sub-categories, sub-sub-categories, etc. in – holistic taxonomy) of the holistic taxonomy. Analysis and control facilitate the evaluation of the analysed cloud security concepts, which enables the exploration of gap areas and trade-offs, both useful inputs for the synthesis model. For the interested reader, a range of commonly applied taxonomy development methods exist in the literature (Nickerson, Varshney & Muntermann, 2013) presents an interesting cross-domain review.

4.3 3-step concept transfer process

Although several studies focus on interdisciplinary (cloud and biology) bio-inspired design methodologies, there is a deficient focus on how interdomain concepts are retrieved, mapped and integrated before transfer into a target domain. This section outlines a 3-step process shown in Figure 33, to facilitate efficient concept transfer from biology to cloud computing.

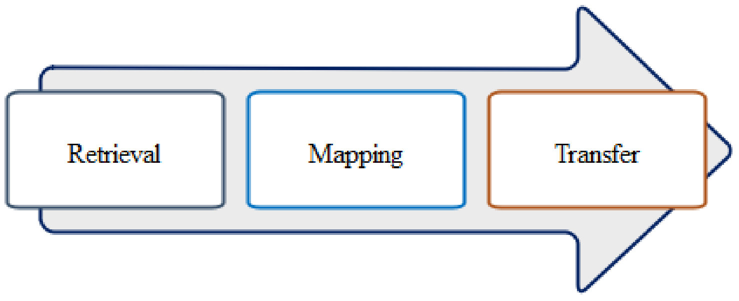


Figure 33. 3-step process produced by the author of this thesis

This is the first part of the broader methodology whose conclusion is the application of TRIZ parameters on contradicting but transferred concepts. Besides the methodological transfer, this process highlights the necessary steps to alleviates common ambiguity challenges in the interdomain transfer. The theoretical significance of this process is in modelling of generic concept extraction and transfer processes necessary for interdisciplinary design in anticipation of limited expert information in at least one domain.

4.3.1 Retrieving concepts

Several methods exist in information retrieval areas such as data mining and text mining, etc. This is an important step for effectively mapping of features and establishing relationships among interdisciplinary knowledge. For the purposes of this research, let us assume an arbitrary finite set of systems or a system of systems, where systems or a system of systems might correspond to natural systems, e.g. plants, mammals, reptiles, etc. They might also correspond to some form of relationship among system components or sub-systems across concepts such as parenting, hierarchy, habitat, etc. It may also be the case that a natural system can be specified using its features or a combination of features.

Starting from a quasi-formal characterisation standpoint, let us suppose that a system X possesses some similarities to system Y in some aspect. Moreover, let us also suppose that system X possesses some unique feature Z, such that system Y possesses feature Z or some other feature, Z*, which are like Z. Given two domains of interest; source domain, N, and target domain, C, to represent a natural system and cloud system, respectively. Each is comprised of a set of objects; a derived set of first order and/or high-order statements $N = \{N^o, N^p, N^r, N^f\}$ are components of the source domain, while their counterparts $\{C^o, C^p, C^r \textit{ and } C^f\}$ are components of the target domain. Terms o, p, r and f define unique relationships between corresponding objects in both the target and source domain. Hence, s-feature is a set $q = < f_1, f_2, \dots f_n >$ of cardinality n, whereupon nature and cloud system are defined according to the following:

- A nature system (nature-System), N, is a set of m-features, where $\{N = N_1, N_2, .. N_m | m \geq 1\}\{N = F_1, F_2, \dots F_m | m \geq 1\}$, implying that nature system is are a set of m-tuples, which is true in N for both the set of nature systems and the features that define N. It may not be the case that n-features are strictly the same in natural systems. For brevity sake, n-features shall from this point on be referred to as features, in respect to concepts and attributes of natural environments.

- A cloud system (cloud-System), C, is a set of c-features, where $\{C = c_1, c_2, \dots c_n | n \geq 1\}\{C = F_1, F_2, \dots F_I | I \geq 1\}\{C= c_1, c_2, ..c_n{}'' n \geq 1\}$. It may as in nature, be the case that not all c-features are the same in all cloud systems. For brevity sake, c-features from this point on will be referred to in respect of objects as attributes.

4.3.2 Mapping concepts

Survivable preys possess unique attributes and are well adapted to their environments. In addition, they exhibit strong and successful predation avoidance mechanisms, which demonstrates that historical interactions have far-reaching implications for future species (Resetarits, 2001). By understanding such behaviours, it is possible to adopt/adapt such processes for use in cloud computing. Mapping aims to plausibly demonstrate that a proposition is true due to known or accepted similarities between a nature-systems and cloud-system, despite known or acceptable differences between both systems.

Figure 34 below illustrates a generic mapping scheme. The output of mapping is a system consisting of unique objectives; *Obj_1, Obj_2..., Objn,* unique attributes*; $Attrib_1$, $Attrib_2$...$Attrib_n$*, which serves unique benefits: *$Benf_1$, $Benf_2$...$Benf_n$*. First, domains are distinguished according to structure, behaviour and function of their systems. In addition, mapping entails that known differences and similarities are clearly identified, particularly in relation to the central mapping concept. In this research, the central concept is survivability.

Hence, with a set of keywords, the task for mapping is to find a graph function, a set of connected sub-graphs and extract an accurate ranking function to produce the preferred features.

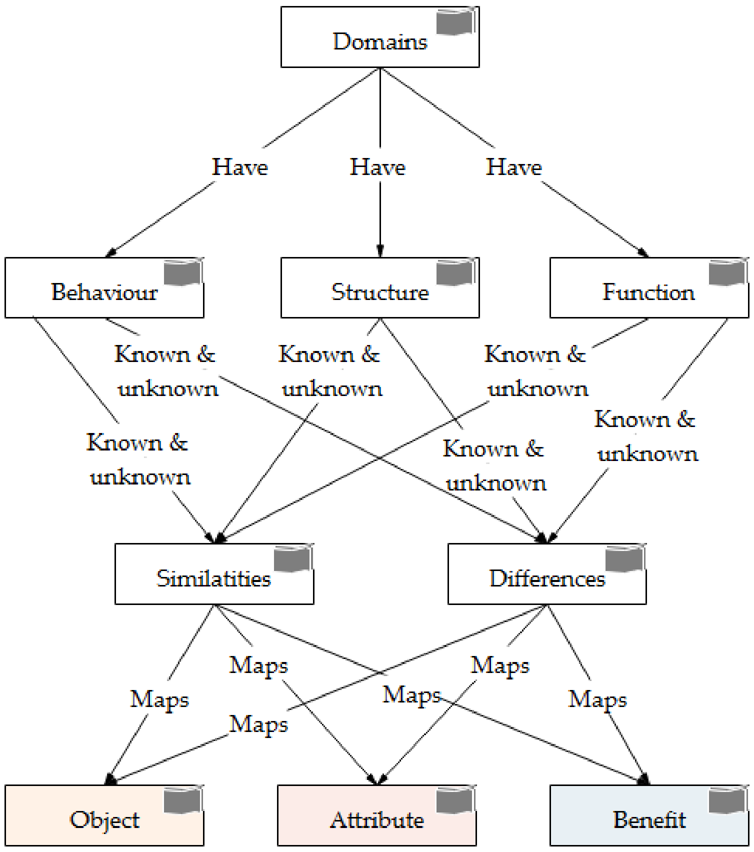


Figure 34. Mapping cloud to biological systems; structure, behaviour and function features, around a central survivability concept

Figure 35 is an illustrative example of a graph function for mapping cloud computing and prey systems. Hence, the definition below is presented add clarity to the remainder of this work.

Definition 4 *(mapping). Links existing knowledge in N and C, despite known or acceptable differences between both systems, i.e. , , , and , where starred symbols represent the inferred similarity.*

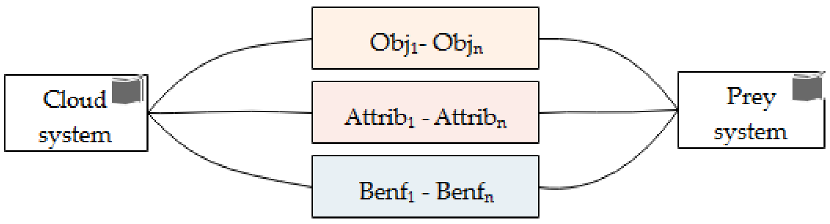


Figure 35. A graph example of cloud-prey system mapping

4.3.3 Transferring concepts

For the purposes of discussion, simple natural language (NL) keywords for the survivability concept are used despite their known challenges. Since the source of biological information is critical, its simplicity (in understanding) is particularly critical for non-biological persons. Thus, using suitable search tools or sources such as in (Sadava *et al.*, 2011) will provide simple understandable background information. Moreover, while searching biological terms tends to be straightforward, a natural language keyword search for analogies tends to be ambiguous, considering that synonyms tend to increase the number of matches per functional keyword (Vakili & Shu, 2001). Along with works by (Sullivan & Regan, 2011), transfer in this section focuses upon verb terms as they objectively enhance a focused search wider biological mechanisms to perform the intended actions for cloud computing problems. Consider the following: Suicide as a biological function of natural prey animals defines:

- Objective – To bring about death
- Benefits – sustain living organisms
- Attributes – predation risk, ageing

In fact, keyword disambiguation methods such as those in (Sommarive & Report, 2013) are suitable to address this challenge. Investigating keyword disambiguation methods is however outside the scope of this thesis. For in-depth details and example techniques and tools, the interested reader is recommended (Ilevbare, Probert & Phaal, 2013). Based upon the cloud computing problem outlined above, transferring concepts from nature to the cloud requires that a designer deduces the general solution using TRIZ. This entails one to decide on the cloud features to improve (i.e. those identified as degrading the survivability concept) and the features to preserve (i.e. those which enhance the survivability concept). For purposes of this work, an interactive online tool (TRIZ40) (Creativity, 2019a; Domb *et al.*, 2011a) facilitates this process. For purposes of this research, three (non-exhaustive) survivability natural language keywords which describe the survivability problem features (implied in UUUR features) are identified as central to improving the cloud computing problem, the difficulty of detection, adaptability and ease of repair.

4.4 TRIZ-based approach

Although TRIZ's inventive principles have found use in cloud computing, these have been largely limited to the business management aspect of the cloud, for instance in (Hsu, Tsai & Chen, 2013). Perhaps this is because the original TRIZ is limited where a domain has no explicit match with TRIZ's forty principles. In recent years, however, TRIZ has been adapted to suit other environments such as information technology (Beckmanna, 2015). Based upon inspiration from prey animals in nature, useful analogies are derived from specifically resolved contradictions. The TRIZ method contributes to identifying specific design requirements for future cloud survivability solutions. In this research, it is aimed to address cloud computing security challenges imposed by UUURs.

While prey animals must be exhaustively investigated to garner broader representation of their natural domain, integrating a problem-driven approach (where survivability problem motivates the search for biological analogies towards a solution) (Sullivan & Regan, 2011), means the current contribution supports the intended interdisciplinary design. To understand TRIZ and its application, below are brief descriptions of TRIZ's concepts which are central to this work. These include, but are not limited to, the general TRIZ process, 40 inventive principles, contradiction matrix and contradiction (concept).

40 inventive principles: Based upon an analysis of 40000 patents, the TRIZ inventor suggested the existence of a pattern to most inventions, implying the existence of a solution to all problems embedded within the inventive principles (Labuda, 2015).

Contradiction: TRIZ's inventor suggested contradictions as the administrative, technical and physical constraints which arise as a result of incompatibility that exists between a design's desired features for improvement, and the system (Hsieh, Chen & Do, 2015). In fact, it is argued that the invention itself entails the solving of contradictions.

Contradiction Matrix: The inventor introduced a matrix of 39 technical parameters which aim to resolve technical contradictions. Parameters in the column represent features which are obstacles to desired solutions while column feature is intended for improvement. An intersection points to the contradiction solution. An example is BioTRIZ's contradiction matrix (Craig *et al.*, 2008) with a specific focus on extended solutions from living nature.

The TRIZ's systematic process is illustrated in Figure 36 is adapted from (Ilevbare, Probert & Phaal, 2013). The underlying TRIZ process enables effective bio-inspired design for cloud computing.

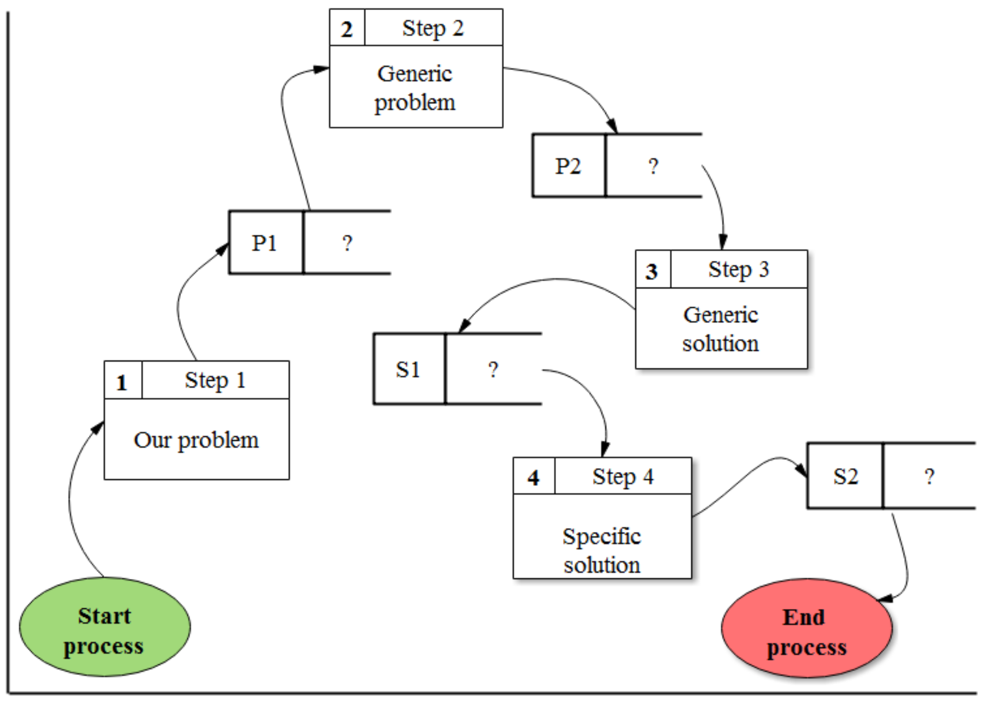


Figure 36. An illustration of TRIZ's systematic process adapted from (Ilevbare, Probert & Phaal, 2013). By following the TRIZ process from start (green) steps represent TRIZ's generic components whereas P1 and P2, S1 and S2 represent TRIZ abstract problems and solutions, respectively.

As (Russo & Spreafico, 2015) postulates, abstraction enables TRIZ principles to be applicable across a wide range of fields. Unlike other problem-solving methods such as brainstorming, lateral thinking, mind mapping, etc. TRIZ provides the additional capacity to suggest conceptual solutions (defined by TRIZ's inventive principles) based upon a specific problem is identified. Moreover, is it possible to further develop specific factual solutions? Defining "our problem" seeks for a specific problem concerning an area to which a solution is required. In addition, defining the "general problem" thus reduces the problem into its elements which are then applied to the contradiction matrix. Based on this latter, a general solution is a combination of TRIZ's inventive principles to eliminate the contradiction. Finally, the specific solution may draw analogies for the specific problem and formulates specific solutions.

4.4.1 Applying the TRIZ

This section shows the application of TRIZ. Although originally designed for solving problems in physics and chemistry, TRIZ has been applied to provide logical, innovative and inventive creations across many domains (Ilevbare, Probert & Phaal, 2013). The application of TRIZ to cloud computing's survivability problem revolves around findings by Altshuller (1999) that interdisciplinary problems and their solutions, i.e. science and other domains, are similar and have repeated patterns of evolution across domains. In addition, the implications of innovation exist beyond the disciplines for which they are developed.

4.4.1.1 Defining the survivability problem

Defining or conceptualising the problem is the first element of the overall TRIZ problem-solving process. Hence, for the purpose of this research, defining the survivability problem is illustrated in Figure 37. As an overview of this process, the following three central questions should be addressed: (1) What is the problem in cloud computing, (2) What needs to the achieved? and (3) What are the obstacles to addressing the problem and achieving the objective?

The survivability problem illustrated below summarily contemplates the following: How to achieve survivability and security in view of UUUR while ensuring resiliency (cost) and reliability on the fly? How to improve the operational challenge of monitoring and detecting UUURs considering unpredictability due to cloud computing's large surface area, broad network access and the complex multi-party interactions? How to improve survivability considering important survivability tread-offs including secure, reliable and cost-beneficial service provision?

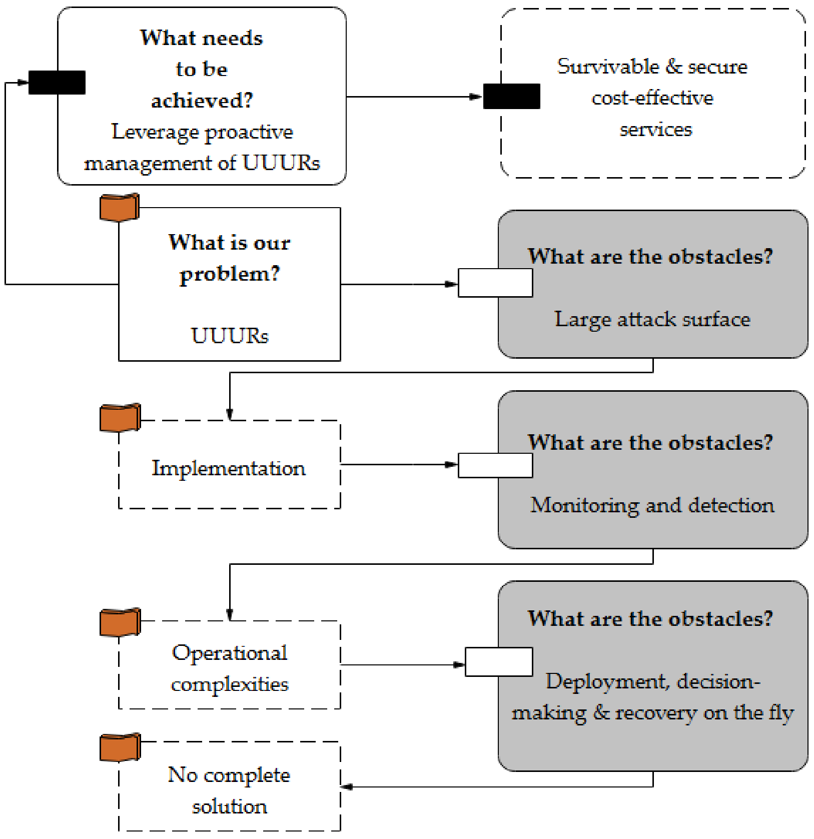


Figure 37. Defining the survivability problem, adapted from (Kasravi & Fellow, 2010)

4.4.1.2 Defining a generic problem

Three (non-exhaustive) survivability features identified as central to improving the cloud computing problem, detection, adaptability, recovery and ease of repair. Figure 38 illustrates the generic problem under this consideration. Briefly, this figure conceptualises the problem for cloud computing in the context of the processes, tools and objects within an environment. For instance, the problem of adapting is shown in relation to the recovery processes as effected by parameter changes. Recovery meanwhile influences some object state, for instance, a virtual machine.

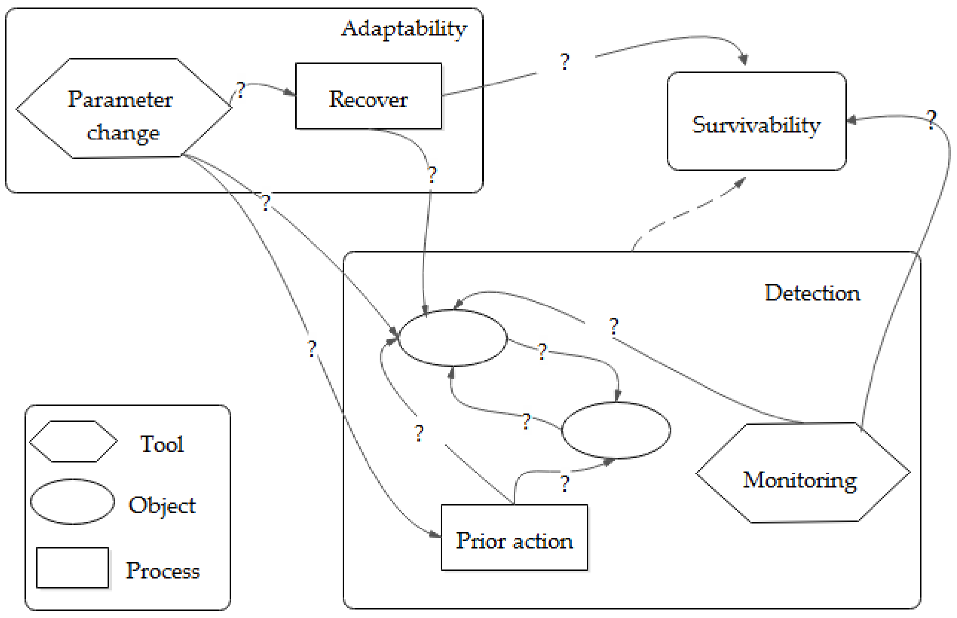


Figure 38. Generic illustration of the survivability problem

Table 10 gives a summary and description of each. An interactive online tool; TRIZ40 (Creativity, 2019b; Domb *et al.*, 2011b) facilitates the TRIZ process of finding solutions by resolving contradictions.

Table 10. Features to improve and to preserve and summary of corresponding TRIZ principles

Feature to Improve	**Feature to preserve**	**TRIZ principles to solve the contradiction**
Difficulty of detection	Area of a stationary object	**2 Taking out**: Separate an interfering part or property from an object, or single out the only necessary part (or property) of an object **16 Partial or excessive actions:** If 100 percent of an object is hard to achieve using a given solution method then, by using 'slightly less' or 'slightly more' of the same method, the problem may be considerably easier to solve **30 Flexible shells and thin films**: Use flexible shells and thin films instead of three-dimensional structures. Isolate the object from the external environment using flexible shells and thin films **39 Inert atmospheres**: Replace a normal environment with an inert one. Add neutral parts or inert additives to an object
	Area of moving an object	**2** As above; **16** As above; **30** As above; **39** As above
	Device complexity	**10 Preliminary actions**: Perform, before it is needed, the required change of an object (either fully or partially). Pre-arrange objects such that they can come into action from the most convenient place and without losing time for their delivery **15 Dynamics**: Allow (or design) the characteristics of an object, external environment, or process to change to be optimal or to find an optimal operating condition. Divide an object into parts capable of movement relative to each other. If an object (or process) is rigid or inflexible, make it movable or adaptive. **28 Mechanical substitutions**: Change from static to movable fields, from unstructured fields to those having structure.
Adaptability	Reliability	**13 The other way around:** Invert the action(s) used to solve the problem, make movable parts (or the external environment) fixed, and fixed parts movable, Turn the object (or process) 'upside down'. **24 Intermediary:** Use an intermediary carrier article or intermediary process, Merge one object temporarily with another (which can be easily removed). **35 Parameter change:** Change an object's physical state, Change the concentration or consistency, Change the degree of flexibility.
	Ease of repair	**1 Segmentation:** Divide an object into independent parts, make an object easy to disassemble, Increase the degree of fragmentation or segmentation.

		4 Asymmetry: Change the shape of an object from symmetrical to asymmetrical, and If an object is asymmetrical, increase its degree of asymmetry**.** **7 Nested dolls:** Place one object inside another; place each object, in turn, inside the other, Make a part pass through a cavity in the other. **16** As above
	Extent of automation	**27 Cheap short-living objects**: Replace an inexpensive object with a multiple of inexpensive objects, comprising certain qualities (such as service life, for instance). **34 Discarding and recovering**: Make portions of an object that have fulfilled their functions go away (discard by dissolving, evaporating, etc.) or modify these directly during operation; Conversely, restore consumable parts of an object directly in operation. **35** As above
Ease of repair	Cost	**1** As above; **10** As above **32 Colour changes:** Change the colour of an object or its external environment. Change the transparency of an object or its external environment.
	External harm affecting the object	**2** As above; **10** As above; **16** As above; **35** As above
	Loss of substance	**2** As above; **27** As above; **34** As above; **35** As above

4.4.1.3 Defining a generic solution

Based upon the generic problems and the identified TRIZ parameters in Table 12, the generic solutions, i.e. analogies, are developed briefly explained in this section. For each generic solution, a cloud computing example is provided for to give closer semblance to a specific cloud computing environment and potentially address the "factual problem directly" (Ilevbare, Probert & Phaal, 2013). For each of the three survivability problems, detection, adaptability and ease of repair, recommended TRIZ parameter solutions are conceptualised as analogous examples to the inspirational prey system. Furthermore, analogies for cloud computing survivability are derived based upon prey's anti-predation and predation avoidance mechanisms.

Detection: Autonomous members of the community cooperate to manage (including eliminating) entity-level activities that negatively impact the global state of the community (parameter **2:** Taking out). Also, rather than waiting to respond after an attack, proactive observation, planning and alarms calls improve the timeliness of pre-planned predation avoidance and anti-predation processes and/activities (parameter **10** Preliminary actions). Self-organising communities enable survivability objectives to be identified, changed and updated dynamically according to changing requirements (parameter **15** Dynamics). Besides, partial or excessive actions allow the execution of a range of actions (at times in combination) that escalate (preliminary-to-post incident) from passive to aggressive and/or vice-versa, according to an on-going process/activity (parameter **16** Partial or excessive actions). In addition, as in prey animals which develop anti-predator mechanisms to ward off attacks, for instance, mechanical, chemical, physical or behavioural, substituting mechanisms in response to stimuli can give advantage to an adversary (parameter **28** Mechanical substitution).

Adaptability: Segmentation means that autonomous entities are designed with properties that enable local-level activity and yet with a collective global goal (parameter **1** Segmentation). In addition, adaptation entails that unusual processes are enabled achieve a global goal (parameter **13** The other way around). For instance, sacrificing an entity in the first instance, rather than implementing costly countermeasures. Moreover, enabling **p**artial or excessive actions means that a combination of actions (preliminary-to-post incident) that escalate from passive to aggressive and vice-versa, according to on-going survivability process/activity objectives (parameter **16** Partial or excessive actions). For instance, instead of creating new entity instances using unnecessary resources to meet the same survivability objectives, rapidly deploy prototypes with only limited requirements (parameter **27** Cheap short-living objects). Moreover, ddiscarding and recovering (parameter **34** Discarding and recovering) allows survivability functions to be modified after an entity instance is deployed, to enable rapid discarding of "costly" entities to fulfil survivability objectives. In addition, reconfigurable survivability parameters can be changed on the fly (parameter **35** Parameter change)

Ease of repair: the ability to change the appearance of an object or the external environment (parameter 32 Colour changes) is uniquely critical for enable recovery. In addition, segmentation, replacement or elimination, preliminary actions; partial or excessive actions, parameter changes including after an entity is created are applied as describe for adaptability and detection.

The following analogies provide some practical examples.

While some analogies postulate that computing infrastructures are homogenously susceptible to attacks, other studies (Gorman *et al.*, 2004) postulate susceptibility to attack as being heterogeneous. In this research, predator-prey analogies aim to capture unique diversification mechanisms that ensure survivability in both homogeneous and heterogeneous prey species. Predation avoidance and anti-predation mechanisms describe the main objectives of diversification, which define how prey species behave to improve its selection and survivability (Jr, Jr & III, 1991). Anti-predation mechanisms describe prey techniques, which reduce the probability of predation, while predation avoidance describes mechanism prey uses to remove itself from the same habitat as the predator. Several works show Lotka and Vito Volterra's model (Luo, He & Li, 2004)(Rozenfeld *et al.*, 2006)(Campillo & Lobry, 2012)(da Silva Peixoto, de Barros & Bassanezi, 2008) as the simplest description of the predator-prey system in which populations changes are a product of the rate of reproduction, the rate of predation and the interaction between species.

Predator-prey systems (PPS) demonstrate complex relationships among interdependent entities; where one depends on the other for food and survival (Colomer *et al.*, 2011). Behavioural, interactional and functions of individual 'networks' impacts on how biological systems change and adapt in time, self-organise and diversify (Zhou, 2009). Thus, a predator is able to assess and learn the quality of prey resources while hunting (Williams & Flaxman, 2012). In contrast, the latter line pertains to exploring the dynamic in entities within a biological system, including how they associate.

4.4.1.3.1 Ecosystem

Figure 39 is a high-level illustration of the ecosystem analogy which characterises the functional and behavioural aspects of cloud systems and the inherent survivability processes summarised as recommended TRIZ. An ecosystem in technology is given the familiar analogy of information environments or digital/media ecologies to include technologies and people in complex dynamical systems (Betz & Stevens, 2013).

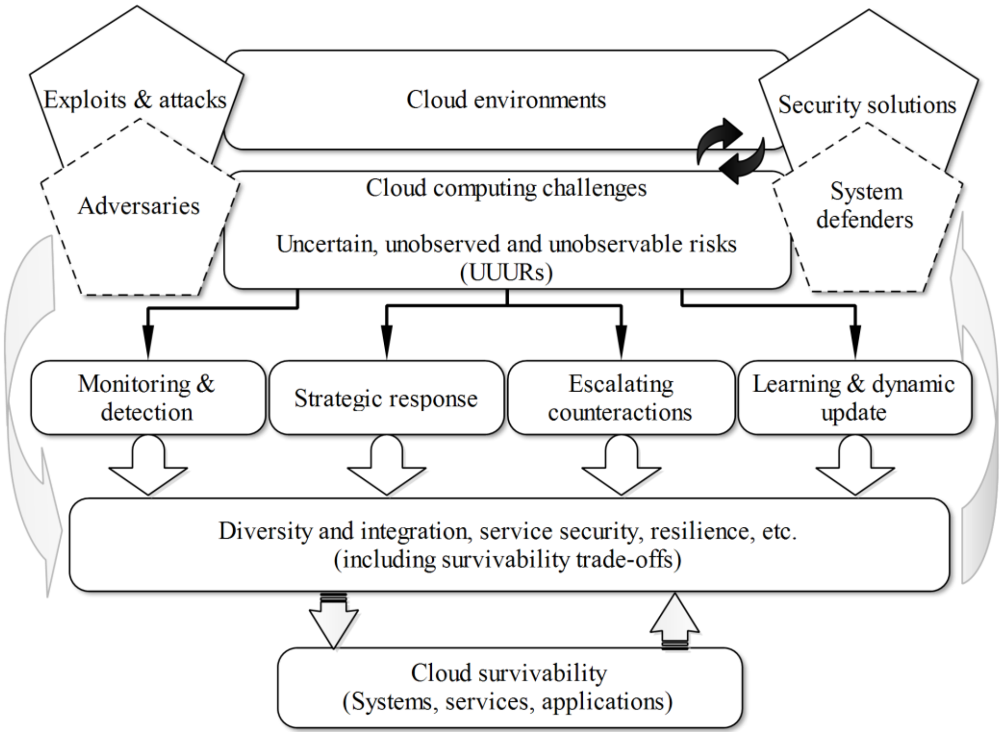

Exploits & attacks
Cloud environments
Security solutions
Adversaries
Cloud computing challenges
Uncertain, unobserved and unobservable risks (UUURs)
System defenders
Monitoring & detection
Strategic response
Escalating counteractions
Learning & dynamic update
Diversity and integration, service security, resilience, etc. (including survivability trade-offs)
Cloud survivability (Systems, services, applications)

Figure 39. Ecosystem analogy

This analogy will enable the analysis of varying survivability concepts to support innovation at all levels; strategic, methodological, operation and organisation. Entities act in response to triggers (internal and external) within their environments, whereupon an action-type leads to outcomes (desired or undesired), e.g. extinction in prey or survivability in the cloud. In nature, prey animals obtain food and information (resource and triggers, respectively) from their environment and respond with action. Actions, in turn, affect the environment, which retrospectively affects the opposing species. Outcomes of encounters/interactions among competing and non-competing animals determine the extinction of species. This research considers the survival of cloud systems where adversaries (attackers) and defenders coexist in cloud environments; whereupon both entities are enhanced by cloud computing's abundant resources (input). Outcomes of adversarial actions affect the cloud, as much as the outcome of interactions between defenders and adversaries determine the survivability of the environmental. Hence in this context, the number of cloud resources, and stimulus (input) from the cloud environment, and the nature of responses (behavioural and/or mechanistic action) are important to survivability. This analogy considers the environment as a physical system with entities who belong to communities and possessing processes (attack-respond analogy, survivability analogy).

Thus, an analogy for the cloud ecosystem that is dynamic and consisting of several entities (Kushida, Murray & Zysman, 2012) is presented. Our premise of the ecosystem analogy is in line with the authors in (Huston, DeAngelis & Post, 1988), who assert that variables including population size are useful to describe organisms in nature, from and individual to the community. It is key to note however, that individuals' organisms in nature have an equal effect on each other since interactions are inherently local (Huston, DeAngelis & Post, 1988). Hence for the current ecosystem analogy, ecological parameters and constraints are assumed in their general rather than explicit sense. As such, a community is conceptualized as being both individualistic and integrated, whereupon an individual is described per its structure.

4.4.1.3.2 Entities analogy

The analogy in Figure 40 is inspired by prey parents who live in groups to improve the collective ability to defend and protect themselves and their offspring against hunting predators. Entity attributes are associated with survivability actions which are in turn associated with the strategic decisions towards survivability. Understanding just how prey parents estimate optimal foraging distance from their offspring is perhaps one behavioural attribute to give insight for future work. Nonetheless, what is indeed known is that prey parents can sacrifice an injured or severely compromised offspring to increase the chances of survival for other offspring. The analogy proposed views virtual machine vulnerabilities synonymous to prey offspring. In this regard, VMs in a cluster represents the offspring of prey animals, while the hypervisor represents the parent, as it manages the greater functions of VMs.

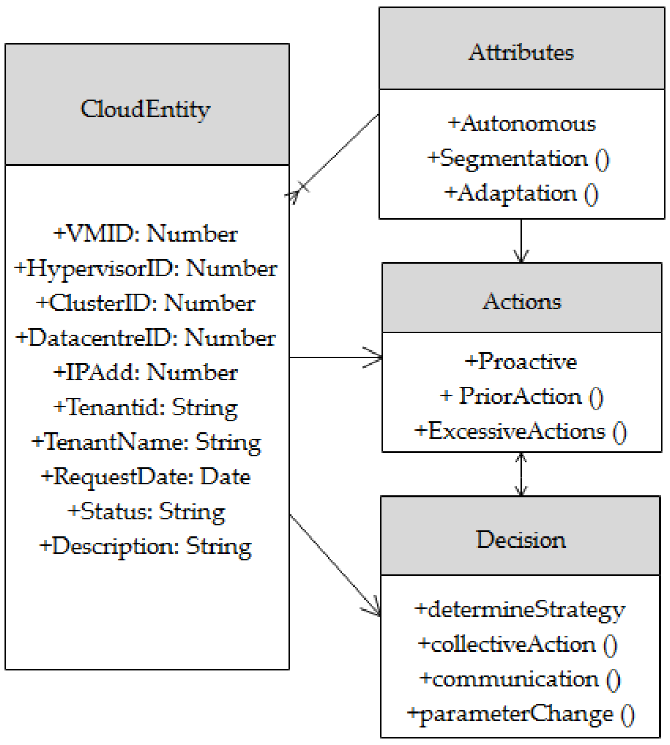


Figure 40. Cloud entity analogy

4.4.1.3.3 *Community analogy*

Research efforts in the predator-prey domain revolve along exploring the importance of behavioural and interactional attributes between two entities, and their implications. In addition, it focuses on how the functions of individual "networks" depend on unique attributes inherent to their communities. The former pertains to evolution; how biological systems change and adapt in time, for instance, the ability to learn and remember information typified by organic self-organisation and individual diversity" (Heiser *et al.*, 2015). Or the ability of a predator to assess and learn the quality of prey (Williams & Flaxman, 2012).

Other research efforts pertain to the dynamic in entities within a biological system, including how they associate. In this regard, attention is upon exploring how animal communities remain the same while the state of individuals changes dynamically due to instance, foraging, death or reproducing, etc. Examples include group size and group formation in foraging primates (Janson & Goldsmith, 1995), feeding habitat selection according to where prey is easier to catch as opposed to where it is abundant, etc. (Balme, Hunter & Slotow, 2007). The examples clearly demonstrate how the autonomous nature of the attribute of natural systems has implications on the behaviours of individuals and vice-versa. The foregoing is typical in nature as epitomized by (Pinol & Banzon, 2011)'s adaptation of Lotka and Vito Volterra's model, to encompass a survival probability or Verhulst Factor.

The community analogy is conceptualised to capture system dynamics (data) in heterogeneous cloud environments, whereupon interactions determine the construction of community. In the previous hypothesis, entities' actions describe concerted mechanisms against attacks. The community analogy which identifies high to low-level dynamics as captured in a cloud infrastructure. Just like prey animals, entities in the cloud although autonomous, belong to a general community. Cloud defenders, for instance, share a common survivability goal. At the local level, interaction dynamics determine decision strategies, including those actions taken to ensure survival. of the local and global levels. This analogy is in the future implementation of virtual machine communication; inter, intra or remote, in cloud computing (Jiang, Xu & Wang, 2006) considering local and global dynamics of an environment. The community analogy juxtapositions multi-tenancy in cloud environments, against virtual machine vulnerabilities and identifies communication or interaction as a powerful component for the survival of entities. This begs the question of how to design VM "communicate" from one topology to the next, from one technology to the next, within and across tenants, within and across different cloud computing environments.

4.4.1.3.4 Attack-response analogy

Ecological literature shows that foraging increases the risk of predation for prey animals (Grand & Dill, 1999)(Sundell *et al.*, 2004)(Grand & Dill, 1999)(Luttbeg & Schmitz, 2000). However, efficient response to alarm calls and chemical responses during an encounter with predator reduces predation losses (Mirza & Chivers, 2001). As been implied in the entity analogy, the ability for parents to defend their offspring is paramount for the survival of their species. In this analogy, prey parents actively engage an attacking predator to distract it away from their offspring, which increases chances for offspring to conceal their presence and/or flee. In extreme cases, fighting a persistent predator is a known behaviour especially among cooperating prey parents. In their work, (Jr, Jr & III, 1991) show in their detailed description of such extreme cases, that is it possible in prey communities to employ poisoning actions as a countermeasure against predators. Similarly (Somayaji, Locasto & Feyereisl, 2007) shows that prey animals are known to perform a sacrifice to improve their survival chances. Logically, this implies global survival is more critical.

The foregoing is hypothesised as the action analogy (proactive prior and partial or excessive) which are implemented against threats and attacks on virtual infrastructures and services. The attack-response analogy describes unusual corrective actions including escalating actions (preliminary-to-post incident and from passive to aggressive), according to an on-going process/activity to improve outcomes performed by cloud security systems or to avoid and recover from compromise (TRIZ principles to address the difficulty of detection and ease of repair). Actions may include protecting assets by hiding their visibility and increasing the complexity of being observed. Increasing the complexity of an asset increases the cost of an attack, which in turn increases the complexity of executing an exploit and gives an advantage to the defender (McQueen & Boyer, 2009). In (Yuill, Denning & Feer, 2006)'s model, for instance, deception is employed to add intelligence to the defender, while thwarting an adversary's capabilities to observe, investigate and learn about a target. In nature, the perception of predation risk, e.g. the persistence of a perceived predator, results in prey developing optimal predation avoidance response, for instance, a hiding prey choosing to flee as the predator gets closer or approaches in a direct movement (Koga *et al.*, 2001). This change in response suggests the analogous escalation of defensive actions in cloud computing environments, i.e. the transition from prior-excessive actions. Escalation mechanisms may exist as specific or non-specific to an on-going event (Matsuda, Hori & Abrams, 1996).

4.4.1.3.5 Survivability analogy

This section presents our interpretation of the survivability analogy based on prey survival against predation discussed in the section above. It is important to emphasise our view that, unlike traditional computing infrastructures where hardening of security systems improved survivability, the ability to remain robust and recover from attack are essential characteristics to pay closer attention to, as the main distinguishing pillars from both the survivability and security context of the cloud. The following scenario outlines the pillars of the survivability analogy.

Threat detection: In nature, prey animals that live in groups have higher survival chances due to increased predator detection. Analogously, the ability to detect threats enhances a timeous response, with the best strategy or countermeasures in cloud environments.

Collective action: Prey's collective foraging strategies, predation avoidance methods and reproduction trends are a result of evolution and adaptation. Analogously, the ability to self-manage, cooperate, adapt and escalate countermeasures increases survivability.

Management: Prey co-habitat to share resources and tasks and reproduce to improve their fitness to avoid extinction. Analogously, diversity, autonomy, integration and self-management of cloud systems and services enhances survivability.

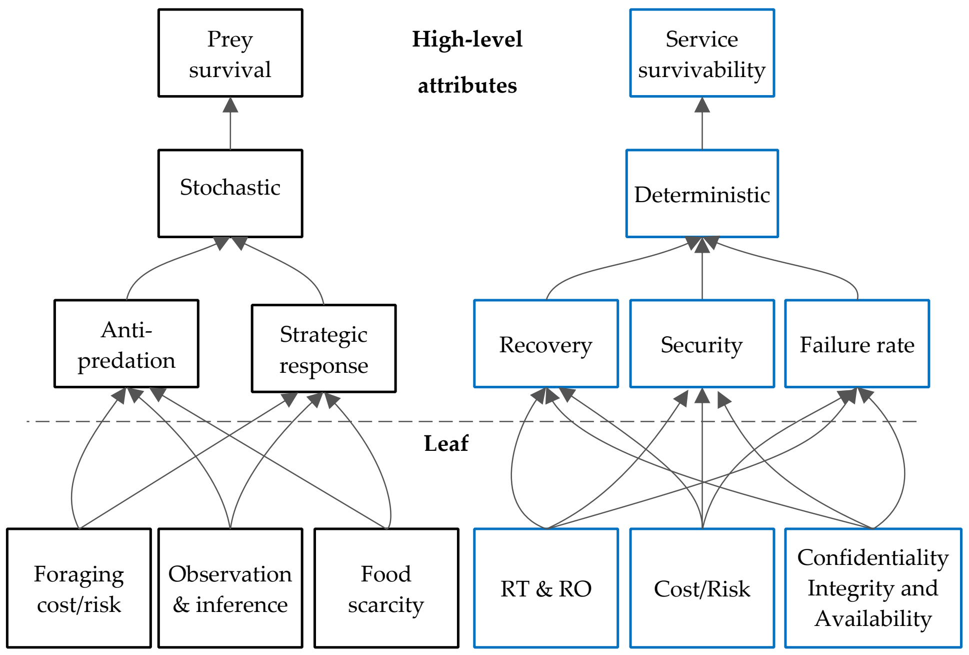


Figure 41. The survivability analogy

Figure 41 is an illustration of the survivability analogy for cloud computing environments based upon prey survival attributes presented. Since the business vision of cloud service providers includes assurances for quality, reliability, the availability of services (Ziring, 2015), leaf attributes such as recovery time (RT) and recovery objective (RO) as well as sustainable pricing to maximise profits are pertinent to cloud service providers. In this scenario, responsibility for security and availability of services including security affecting the customer's infrastructure lies with the CSP (Rong, Nguyen & Jaatun, 2013b). In a traditional sense, security concerns primarily revolved around the confidentiality, integrity, and availability of information.

4.4.1.4 Defining a specific solution

Detection: Since the business vision of CSP include assurances for quality, reliability, the availability of services (Ziring, 2015), RT, RO and sustainable pricing to maximise profits are pertinent responsibilities of CSP. Optimising security and availability of services including security affecting the customer's infrastructure are critical for responsibilities of the CSP (Rong, Nguyen & Jaatun, 2013b). Improving proactive monitoring of the cloud environments, survivability actions and the general system dynamics will address the problem of unpredictability introduced by UUURs. For instance, survivability of a VM-based intrusion tolerant systems based upon inter-communication among multiple VMs is enabled and centrally monitored by a hypervisor (Zheng, Okamura & Dohi, 2015).

Preliminary actions to increase the complexity of an asset increases the cost of an attack, which in turn increases the complexity of executing an exploit and gives an advantage to the defender (McQueen & Boyer, 2009). In (Yuill, Denning & Feer, 2006)'s model, for instance, deceptive measures are employed to add intelligence to the defender, while thwarting an adversary's capabilities to observe, investigate and learn about a target. Moreover, efficient isolation enables compromised VMs to be killed. Thus, proactive capabilities to detect identify & stop an adversary before exploitation, e.g. the "kill chain" approach, is a proactive & dynamic intelligence-gathering method for continuous security posture awareness. As suggested by (Hariri, Eltoweissy & Al-Nashif, 2011), existing solutions do not adapt and escalate their security strategies to counteract the intensity and sheer aggressiveness of an adversary. (Cybenko *et al.*, 2014) suggests that security countermeasures are only successful in traditional networks, while persistent adversaries and zero-day attacks are able to systematically plan their attacks, and persist within the compromised cloud computing environments (Cybenko *et al.*, 2014). On the other hand, the ability to adapt survivability parameters according to changes in the environment, e.g. adapting the current survivability configuration to suit new requirements, adapting survivability objectives to suit new configuration or adapting survivability actions to suit evolving requirements or objectives, etc.

Adaptability: Countermeasures have been suggested including aggressive strategies such as "white worms" (Lu, Xu & Yi, 2013) which actively pursues malicious software with an intent to destroy it, or deceptive techniques such as address hopping (Shi *et al.*, 2007) which masks data in transit from a possible attacker (Gregory, 2011) are suggested to provide dynamic intelligence gathering, security optimisation, and continuous security posture awareness. In addition, deception through systems such as the KIPPO SSH Honeypot, anticipate attacks and swift response (Sochor & Zuzcak, 2014). Discarding and recovering imply adequate policy, guideline & legal framework for integrating, adopting & situational use aggressive countermeasures, e.g. (Rabai *et al.*, 2013) for the "last resort" deployment of aggressive countermeasures.

There is consensus on the core areas closely related to survivability, such areas including resilience, dependability, fault-tolerance, assurance, fault-tolerance, availability, etc. Nonetheless, while survivability is an element of resilience, for instance, one need to pay attention that improving survivability does not adversely affect resilience. For instance, if hardening makes a node highly survivable at a larger financial cost, and increases the overall processing time for infrastructure, but can nonetheless be compromised by a persistent threat, increasing survivability, therefore, counteracts resilience. In fact, survivability should aim to assure the continuity of service or mission with resilience, i.e. at low cost, within the schedule, affordably, etc. In the systems engineering perspective, survivability generally defines a level to which a system is able to continue to provide timely services even during an attack (Ellison *et al.*, 1997a)(Redman, Warren & Hutchinson, 2005)(Mekdeci *et al.*, 2011)(Adams, 2015).

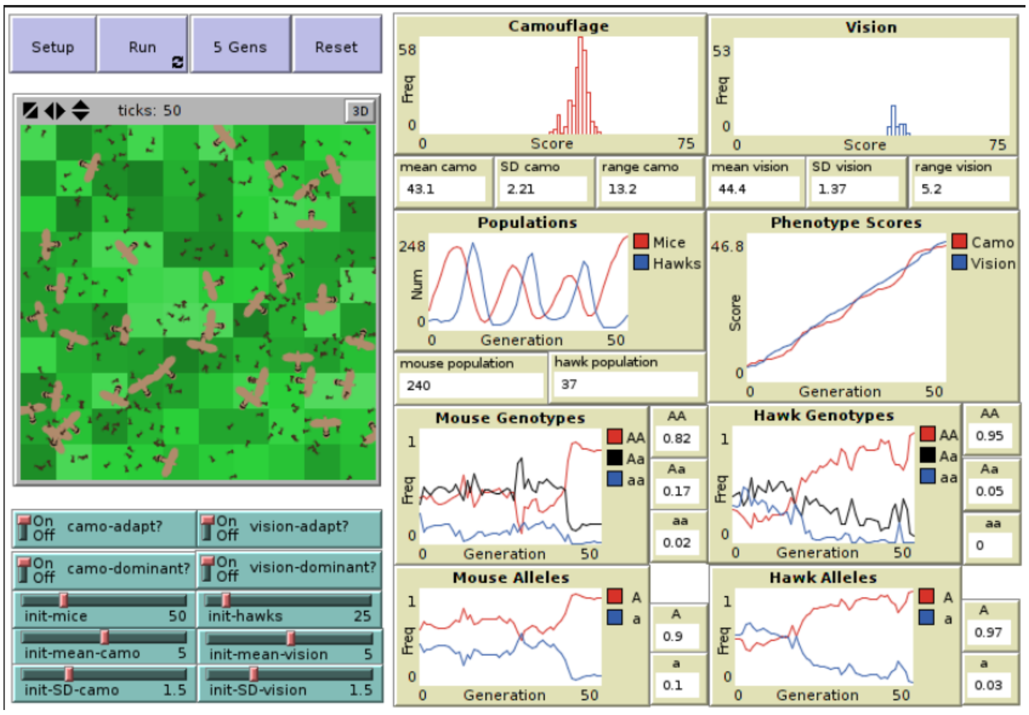
Ease of repair: Segmentation, i.e. multi-agent approach enables optimal processing, improves the ease of recovery by leveraging cloud service discovery, negotiation and composition in dynamic environments. As an example, the authors in (Sim, 2012) suggest an agent-based search engine for cooperative problem solving based on similarity, compatibility and numerical reasoning. Nonetheless, automatic coordination of services composition in multi-party, distributed, dynamic and complex settings requires systematic methods that optimise automation, yet reducing errors. In this research, this means cloud providers can compose survivable services in a manner that accommodates consumers' changing functional and non-functional requirements, at runtime.

Choreographic approaches, on one hand, can facilitate adaption in view of changes (Calinescu *et al.*, 2017). These emerging service engineering approaches enable distributed service composition by specifying tasks, participants and message protocols. Given its centrality, cooperative approaches require feedback control to monitor and manage adaptation among composed services (Arcaini, Riccobene & Scandurra, 2017). As authors in (Weyns *et al.*, 2013) suggest, control loops typically enable the realization of self-adaptation. (Talia, 2012), integrating multi-agent and cloud technologies can unlock even higher performing, complex, autonomous and intelligent applications and scalable yet reliable infrastructures. Survivability should aim to assure the continuity of service or mission with resilience, i.e. at low cost (to both the CSP and consumer), within the schedule, affordable, etc. In the systems engineering perspective, this generally imply a level to which a system is able to continue to provide timely services even during an attack (Ellison *et al.*, 1997a)(Redman, Warren & Hutchinson, 2005)(Mekdeci *et al.*, 2011)(Adams, 2015). Several surveys including (Zheng & Sicker, 2013) (Ribeiro & Hansen, 2012) (Nakano, 2011) (Meisel, Pappas & Zhang, 2010) (Dressler & Akan, 2010), etc. share the consensus view that monitoring and adjusting resources, detection and reacting to changes, including the ability to implement measures against threats, support ubiquitous and yet complex and complete self- organised systems.

4.5 Pilot investigation of prey mechanisms using NetLogo simulator

To investigate the efficacy of the analogies above, proof of concept experiments focusing upon prey mechanisms in the predate-or-survive dynamic are implemented NetLogo (Wilensky, 1999). This approach has also been successfully followed to carry out a sufficient investigation using EcoSim (Lytinen & Railsback, 2010). NetLogo is chosen due to its simple design and in-depth documentation and ease use (Railsback, Lytinen & Jackson, 2006) (Chiacchio *et al.*, 2014).

The simulation environment is setup on a local machine running a standard Windows 10 operating system. These are implemented using NetLogo version 5.3. with 'in silico' experiments, meaning that models are adaptable using different parameters to mimic different environments. Using the buttons on the left-hand side of this graphic, it is possible to configure and adjust the experiment parameters whereas the righthand side simulates the output. Figure 42 is an example NetLogo GUI.



mean camo	43.1
SD camo	2.21
range camo	13.2
mean vision	44.4
SD vision	1.37
range vision	5.2
mouse population	240
hawk population	37
init-mice	50
init-hawks	25
init-mean-camo	5
init-mean-vision	5
init-SD-camo	1.5
init-SD-vision	1.5

	Mouse	Hawk
AA	0.82	0.95
Aa	0.17	0.05
aa	0.02	0
A	0.9	0.97
a	0.1	0.03

Figure 42. The NetLogo GUI

4.5.1 Experiment

In the following experiments, agents inhabit an environment; predator agents kill and eat prey to survive, otherwise, they die. If prey agents avoid predation, they are assumed to survive. This is an arbitrary assumption that is purely designed for the system under investigation in this study. The experiments provide a proof of concept that predator-prey analogies are valuable indicators to survivability, itself useful and central to solving security challenges in the cloud. For each experiment, a run is implemented with varied parameters aimed to evaluate their effect upon prey populations. Due to the wide scope of the predator-prey dynamic in ecology, the experimental data obtained and explained are limited only to a few survival mechanisms. These deliberate restrictions are however adequate for the problem under consideration, and evidence borne out of these experiments suggests some important contributions.

4.5.1.1 Effect of the environment

This experiment simulates prey who feed on grass and reproduce and predators that feed on prey and reproduce. An invading species is introduced, which at the evolutionary scale, affects the behaviours of existing predator (red -birds) and prey (purple - bugs) species. The percentage of grass (green) available for prey is varied on a scale of (0 - 100), while the initial population of prey to predators is also varied in different ratios, i.e. more prey to predators, more predators to prey, etc. In this experiment, prey has unlimited resources (100% of grass) and an equal reproductive capability to predators.

Table 11. Random parameters of predator-prey interaction

	Grass %	# prey	# predator	Prey repro	Predator repro
Run 1	100	159	60	None	None
Run 2	100	159	60	Max	Max
Run 3	50	159	60	Max	Max
Run 4	50	172	109	-	-

Random initial population ratio of prey-to-predators is chosen since survival and grown parameters are not affected by the intensity of predators (Mandiki *et al.*, 2007). Each data point in the plot represents the final population at the end of a simulation of 1000 ticks (timestamps).

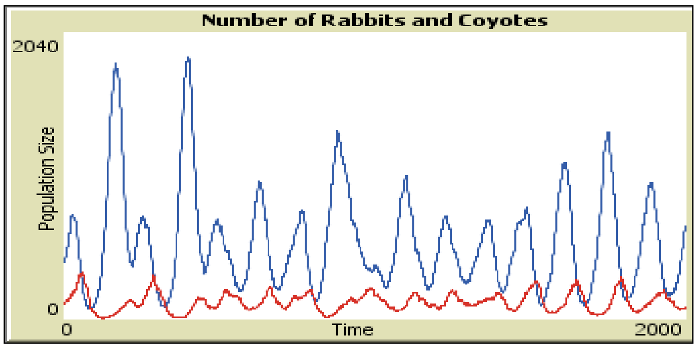


Figure 43. Simulation results of random predator-prey interactions in Table 11. Blue (rabbits) and red (coyotes) oscillation indicate the behaviour prey and predator populations, respectively, in a stable ecosystem

True to in LV's equation, predator and prey interactions indicate the survival or extinction of species. As demonstrated in Figure 43, predator and prey dynamics are oscillatory. However, less food (grass) availability for prey severely depletes the predator population in comparison to prey. As suggested by (Schoener, Spiller & Losos, 2001), predation or the lack of it, impacts upon the probability of extinction and threatens the overall survivability of that species.

4.5.1.2 Effect of poison

This experiment introduces poison, an anti-predator mechanism employed by frogs (*Dendrobatidae*) (Darst *et al.*, 2005) against predators. Survivability is evaluated (rabbits against predating cayotes) when a toxic poison is introduced in the environment. It is assumed that population changes are indicative of survivability: rising populations imply survival while declining populations imply the opposite Table 12 is a summary of random simulation parameters, including when poison is introduced as a survival strategy.

As suggested by Mandiki et al. (2007), initial values are randomly chosen since they have no effect on final survivability. This simulation tracks the interactions between prey (rabbit) and predator (coyote) and the population changes as they co-exist. As in experiment 1, both species are assumed to have adequate access to food and reproduce at ecologically reasonable rates. However, during the simulation, it is possible to introduce a poisonous toxin.

Table 12. Summary of predator-prey interaction with poison

	# of prey		# of predator		Poison %	Start Poison (t)	End Poison (t)
Run 1	300		100				
Run 2	200		130		5	500	1000
Run 3	150		250		10	500	1000
Run 4	400 Same as in 4		100 Same as in 4		2%	500	1550
Run 5	400		100			500	1550
	Gain 50	Repro 7%	Gain 10%	Repro 7%			

The introduction of poison motivates the question of how a generalised or targeted countermeasure impacts upon the survival or extinction of entities within this system. Three things are to be noted in this experiment: the effect of the poison on predators, the effect of the poison on prey and the dynamics before and after the introduction of poison. Figure 44 shows LV's oscillatory predator-prey dynamic. However, introducing poison (black line) at varying times during the simulation affects both species. Predator populations deplete, while on the contrary, prey populations seem to gradually increase and rapidly grow when the poison is removed. Varying the initial population sizes (more prey than predators or more predators than prey) has no significant impact on both populations.

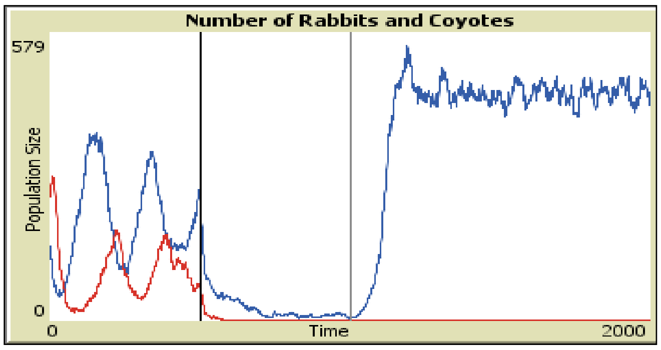


Figure 44. Predator and prey dynamics with poison added at t – 500 – 1000

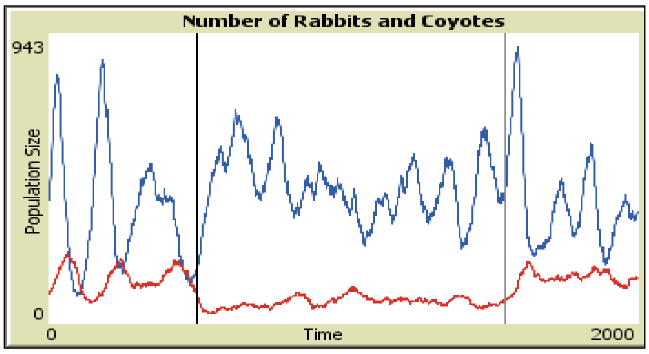


Figure 45. Predator and prey dynamics with poison added at t = 500 – 1550

Interestingly, when comparing Figure 44 and Figure 45, in the latter, predators seem to improve fitness against poison over a longer period of poising, in spite of lower initial populations. This seems to suggest that aggressive anti-predator mechanisms such as poison, while they work to improve prey fitness also force predator adaptation to tolerate the poison. The analogies developed earlier help deconstruct complex concepts in both nature and cloud, in a manner that is understandable to a non-computing reader. This makes it possible to develop explicit definitions of meaningful terms for the system they are used, for instance, terms such as "*deceptive alarm*", "*deceptive camouflage*", or "*deceptive masquerade*" have greater relevance as explicit descriptions of three different levels of deception in computing security terms, in contrast to the widely used general term. The introduction of secondary defensive action (poison in experiment 2) provides insight into cloud computing countermeasures. Nonetheless, aggressive countermeasures are contentious in computing environments due to possible associated legal liabilities (McGee, Sabett & Shah, 2013).

While the generalised approach for extracting a specific process from nature does not perfectly mirror the models and data in nature to cloud environments, applying TRIZ enables the transformation of natural survival mechanisms for deployment in cloud computing. The TRIZ approach proposed is systematic, making it extensible and amenable to future developments. In fact, this research takes inspiration from quantifiable prey processes and fitness parameters to manipulate dependencies and test their consequences in cloud environments. Thus, parameters in this context are not defined by explicit imperial data as those in natural systems, where data enables quantitative predictions. Instead, parameters are only intended for general-level use to highlight structural and mechanical concepts within prey communities.

Unlike nature, verifying security and survivability in real-time cloud environments aims to generate guarantees that these systems remain secure and retain their survivability capacities over time. Thus, the use of quantitative verification methods can be extended for application in prey-inspired systems. In this sense, verification would aim to ensure correctness, reliability, survivability, performance, etc. of dynamic cloud systems through stochastic methods and finite-state transition models. To enhance security, it is important that formal verification methods are integrated into reconfigurable self-adapting methods such as those in nature's prey animals. Highly secure and survivable cloud systems, in this case, can, therefore, guarantee continuous and timely delivery of cloud services, even under threat. QV methods generally analyses state transition models and DTMC and MDP models are generally described with transition probabilities. Thus, probabilistic model checking tools can be used to automatically verify systems. Based on the foregoing, the following concepts are suggested as significant to cloud solutions.

- Non-extinct prey animals are successful because they can manage contradicting demands of obtaining the resources necessary for survival and avoiding predation. Thus, roles and functions of entities within complex cloud environments can be designed with high autonomy and networked with extreme modularity. While cloud providers are capable of securing servers across public, private and hybrid clouds and give real-time detection for a wide range of security events and system states (Ahamed, Shahrestani & Ginige, 2013), monitoring arguably requires autonomic intelligent system and self-defence capabilities analogous to prey animals.

- Organisation and cooperativeness require feedback mechanisms to monitor and manage adaptation (Arcaini, Riccobene & Scandurra, 2017). (Weyns *et al.*, 2013) suggest control loops to typically enable the realisation of self-adaptation.

- Foraging roles follow a hierarchy of approaches to enable survival. For instance, avoiding predators implies adopting behaviours and mechanisms to detect predators first, staying in groups to reduce the likelihood of being killed or mobbing and other coordinated group methods.

Toxins or specialised morphologies can also be used against an attacking predator. Prey's predation avoidance behaviours and anti-predation mechanisms are central to its survival. Hierarchy is central to structure in relation to roles, functions and organisation, towards accomplishing a global goal.

4.6 Summary

This chapter presented a biology-to-cloud computing design method that satisfied H1 of the research hypothesis outlined in Chapter 1. Section 4.2 presents a high-level methodology for a prey-inspired design outlines two generalisable requirements for bio-inspired survivability solution: (1) a problem-driven approach, and (2) the 3-step process to transfer interdomain concepts. Section 4.4 introduces the TRIZ, a known methodology that facilitates innovative design by resolving contradictions and applies it for cloud computing. Section 4.6 presents a pilot NetLogo simulation that serves to evaluate the plausibility of proposed analogical solutions. The proposed solution form core components of the prey-inspired survivability concept designed in Chapter 5, i.e. prey-inspired survivability framework for cloud computing environments (Pi-CCSF).

Chapter 5 Prey-inspired cloud computing survivability framework (Pi-CCSF)

This chapter presents the prey-inspired cloud computing survivability framework (Pi-CCSF) built upon TRIZ-derived survivability specifications defined in Chapter 4 (Section 4.5.4). The Pi-CCSF is significant as it is proposed to support the extension of existing frameworks and model-based analysis of prey-inspired survivability requirements. This prey-inspired conception satisfies the hypothesis under test with respect to escalating survivability design principles and decision techniques presented.

This chapter is based on the work published in RP1, RP3, RP4 and RP6.

5.1 Introduction

As in prey animals, cloud computing requires survivable components to detect changes and evolving environment constraints, learn new behaviour patterns and update their action matrix. Hence, rigorous evaluation of environment constraints and behaviour patterns, selection of actions based upon learned adaptation, experience exchange and executing actions that achieve efficient adaptation is critical (Jiao & Sun, 2016). Traditional survivability methods relied upon a notion that compute-level interactions are predictable and therefore controllable. For instance, virtual embedding techniques are most efficient when desirable system input functions necessary for a desired system output are predictable. However, with UUURs, cloud system behaviours are difficult to observe, and the threats are difficult to detect and predict. It is clear in this regard, that virtual embedding as a technique therefore becomes limited.

5.1.1 Assumptions and Scope

While this thesis refers to a range of attacks, the detailed discussion of attack paths and models is outside the scope of this research. Instead, this thesis focuses upon a class of risks that are uncertain, latent, unobserved or unobservable (UUURs) as observed by (Ma & Krings, 2011)(Ma *et al.*, 2014)(Albanese, Jajodia & Venkatesan, 2018). Traditionally deterministic security methods are assumed to be inadequate mitigation, and often misapplied due to the unpredictability imposed by UUURs. Thus, Figure 46 is an illustration of the design assumptions under this consideration.

Briefly, some survivability objective is assumed to be affected by some UUUR and knowledge of UUURs helps define and identify survivability requirements. Meanwhile, some survivability requirement is assumed to address some UUUR, while also aiding to identify mechanisms towards a solution. A mechanism is assumed to assist in implementing some survivability requirement.

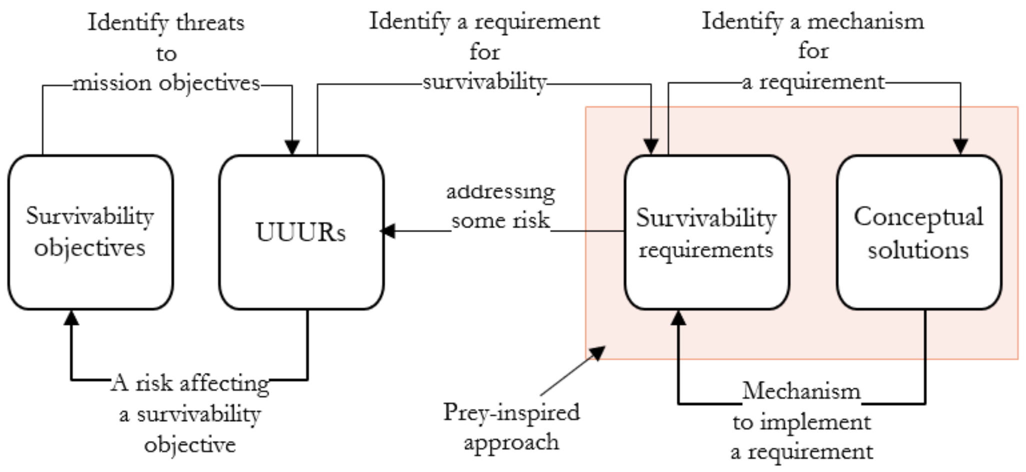

Identify threats to mission objectives
Identify a requirement for survivability
Identify a mechanism for a requirement
Survivability objectives
UUURs
addressing some risk
Survivability requirements
Conceptual solutions
A risk affecting a survivability objective
Prey-inspired approach
Mechanism to implement a requirement

Figure 46. Survivability design assumption

In addition, this research assumes the case that each initial deployment of survivable services, a CSP and CC have a negotiated SLA. SLAs stipulate security and survivability rules and contractual implications (Wagle *et al.*, 2016). This will identify among other things, the deployment of the prey-inspired cloud survivability solution. As shown in Figure 47 this SLA is also renegotiable. Due to the use of cloud computing in critical systems, the significance of SLAs in handling assurances for security and survivability of services in therefore assumed as increasingly critical to both the CSP and CC. Thus, the prey-inspired survivability design assumptions impact upon survivability SLA cycle.

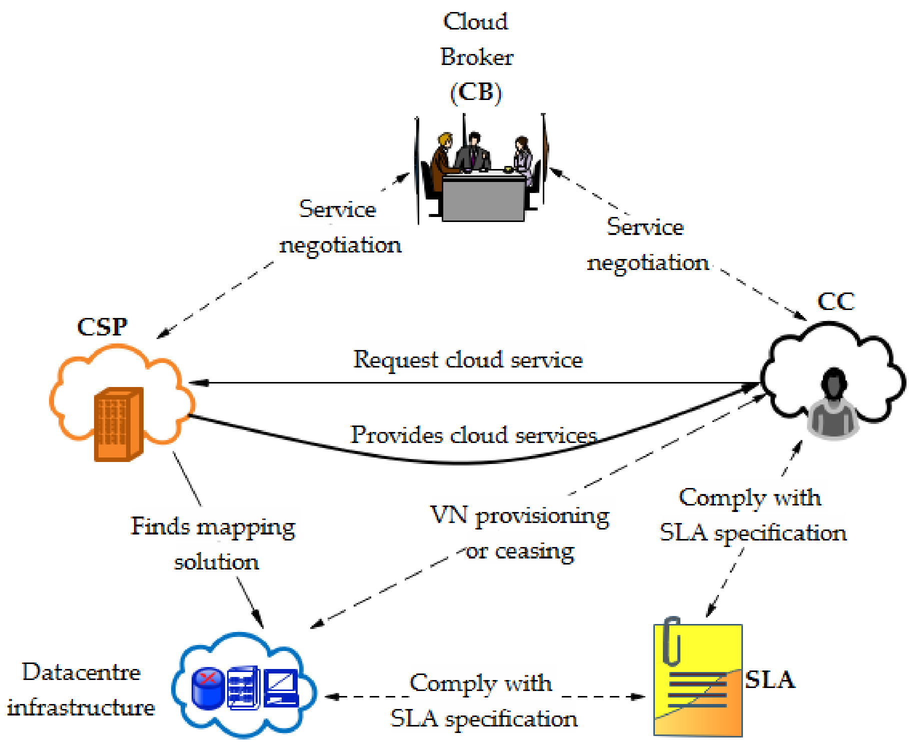


Figure 47. The research's SLA cycle assumption

Furthermore, this thesis assumes that IaaS, PaaS, and SaaS are the most common cloud service models, and therefore adequate to represent a cloud computing view. Figure 48**Error! Reference source not found.** shows the control responsibilities related to CC and CSP. As the user transitions across models (including from the traditional) the user's control and responsibilities decrease. Among these service models, the CC and CSP share to some degree, different levels of control and responsibility on applications, data, runtime, middleware, operating systems and the virtualisation layer components such as storage networks and servers.

Moving from the IaaS model to SaaS, the user's control over core cloud components decreases along with their responsibilities. Thus, from a service provision point of view, it is logical to implement both PaaS and SaaS atop the IaaS model. Due to the control and responsibilities considerations mentioned above, the prey-inspired cloud survivability notion is scoped around the IaaS service model as it removes the limitations of service deployment.

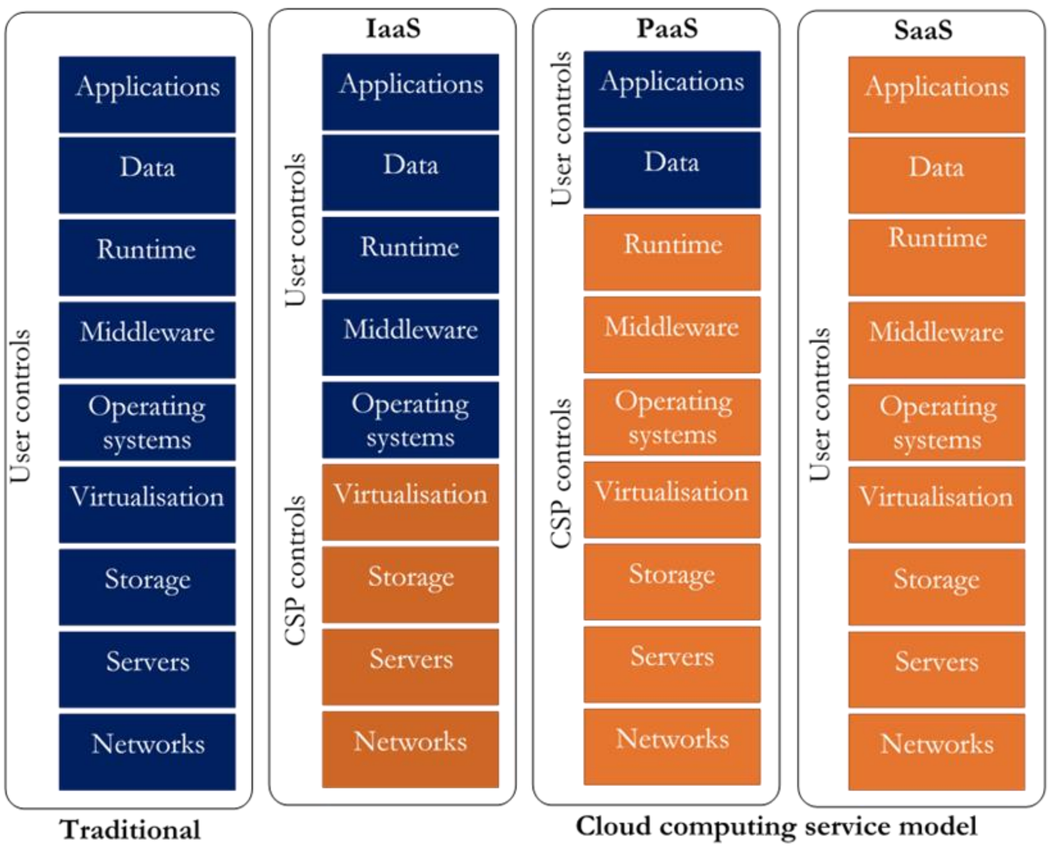


Figure 48. The cloud computing service model control research scope

The remainder of this chapter is as follows: Section 5.2 presents the design context for Pi-CCSF, briefly introducing the survivability mission, survivable system requirements and mission requirements. Section 5.2 details the security context around which Pi-CCSF is developed. Section 5.3 presents Pi-CCSF and details its main components and processes. Section 5.4 presents the survivability mechanisms and Section 5.5 presents Pi-CCSF's overall process-flow. Section 5.6 discusses Pi-CCSF in a conceptual application. Section 5.7 concludes the chapter and summarises the main findings.

5.2 Survivability-oriented design

This research researcher contends to the design challenges imposed by UUURs. However, the bio-inspired design theoretic perspectives present this thesis's researcher with relatively simple units to manage yet still provide emergent phenomenon. As noted by Shu et al. (2011), biologically "perceived" complexity to real complexity is a necessary paradigm shift that can be adopted from biology to manage complexity in cloud environments. Hence, this thesis's survivability design approach is placed in the engineering context to encompass systems security design as a component of service-oriented mission assurance (SOMS). To demonstrate this context, the relationship between cloud computing survivability design with other security systems-related domains is outlined below and illustrated in Figure 49. As shown below, survivability design draws from other related concepts such as resilience, durability, fault-tolerance, etc. More so, it also draws from other contextual requirements. In this research, this includes survivability requirements of cloud computing systems (e.g. a holistic view to cloud computing security presented in Chapter 3, or the contradiction analysis suggested by the TRIZ method and addressed by TRIZ's contradiction matrix (Creativity, 2019b).

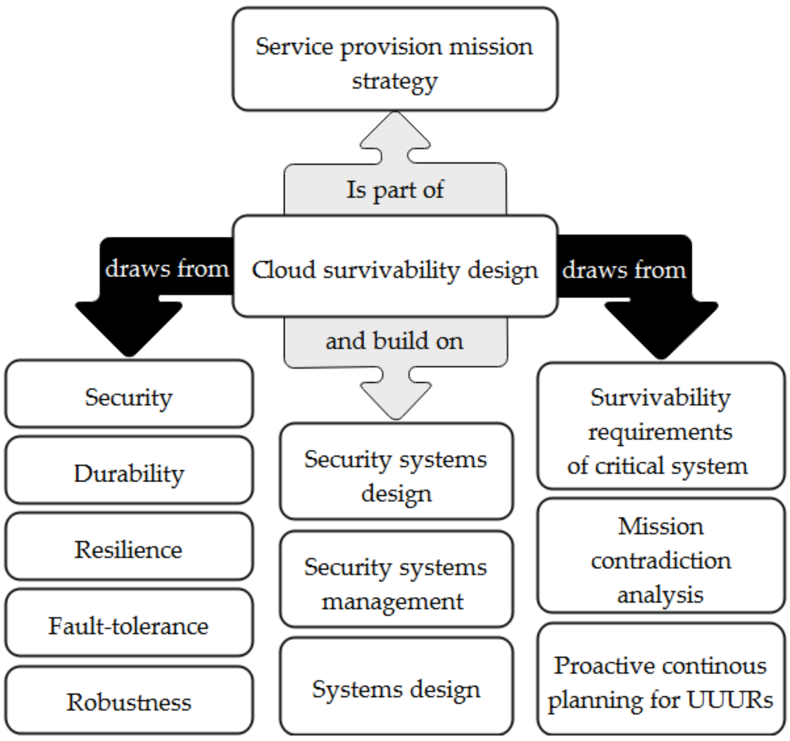

Service provision mission strategy
Is part of
draws from
Cloud survivability design
draws from
and build on
Security
Durability
Resilience
Fault-tolerance
Robustness
Security systems design
Security systems management
Systems design
Survivability requirements of critical system
Mission contradiction analysis
Proactive continous planning for UUURs

Figure 49. Cloud computing survivability design context

For IaaS cloud providers such as Amazon EC2 (Amazon Web Services, 2011) and IBM Business Cloud (Leone, 2015), VMs are central to the service commodity they provide. VM provisioning, with respect to specific resources and capacity, should, therefore, be survivable to ensure continued service. Hence, SLAs between CSP and CC should stipulate among others, the expected levels of services which implores CSPs to the evaluation of their infrastructure (Longo *et al.*, 2011). Whereas evaluation approaches among CSP specifically assess availability and reliability of cloud systems (Wagle *et al.*, 2016), (Nguyen, Kim & Park, 2016), and evaluation techniques focus upon the VM model, frameworks (Sterbenz *et al.*, 2010b), fault tree (FT), reliability block diagram (RBD), continuous Markov chain (CTMC) or stochastic models (Longo *et al.*, 2011), (Nguyen, Kim & Park, 2016), etc. it is important that survivability is emphasised. From a cost perspective, a service-oriented design mission ensures that cloud systems are designed with the capacity adapts to ongoing changes. As noted by NIST, it is critical that assessing and managing risk is iterative regardless of the level of exposure or the sophistication of the security incident (National Institute of Standards and Technology, 2018).

5.2.1 Survivability mission

Survivability is generally described as a mission-oriented process; to which a system can timely provide services after intrusion or compromise occurs (Wang *et al.*, 2012b). The authors in (Mehresh & Upadhyaya, 2012) note the 'mission' element of survivability as upon ensuring the continuity of a set of essential services, bearing in mind that precautionary countermeasures will fail. Adaptability, detection and ease of repair are unique concepts derived through TRIZ to facilitate the cloud survivability mission. Adaptability is evaluated as the capability to respond to situational changes, including changes in resources, requirements, as well as changes to timed events. This also includes response to real-time and dynamic events (self-adapting) in a timely manner. Timeliness is itself dependant on other attributes such as serviceability and cooperativeness. Adaptability is therefore fundamental for avoiding negative measurements of "timed" events and can demonstrate tolerance to attacks. Adaptability can be quantified based on the following: (a) on prior experience of the infrastructure's ability to mitigate attacks caused by, for instance, insiders or outsiders; and (b) proactiveness, which can be estimated based on the security risk management process. The adaptive property of survivability is therefore optimal when it satisfies 'a' and 'b' and represented as a linear function whereupon an increase in adaptability means the system can support survivability attributes, in turn enhancing the overall survivability of a system.

Detection and ease of repair are encapsulated in cooperability, which describes the ability of autonomous entities to collaborate based on a set of predefined survivability mechanisms in response to a security incident or a situational change. Theoretically, this concept means that: (a) VMs execute actions as countermeasures according to a learned criterion, analogous to cooperating autonomous preys (b) VMs execute collective countermeasure mechanisms, synonymous to mobbing behaviour in preys. Since knowledge about the state of the environment, a decision-making parameter which factors the attitude of a CC and/or CSP as stipulated in the survivability mission. This introduces the notion of a strategic model of choice under uncertainty. For the purposes of the current work, the following attitudes will be important to the survivability mission and the requirements that follow. They are also widely published in decision researches including (Yager, 1995)(Fenton & Wang, 2006)(Bracha & Brown, 2012) (Jefferson, Bortolotti & Kuzmanovic, 2017):

Optimistic: a survivability decision-making parameter key selects for each action, the best possible outcome, then selects the action that has the maximum best outcome. As postulated in Bracha and Brown (2012), this strategy relates to decisions when desired future outcomes are at stake, e.g. health, success, employment, etc. Intuitively, in the context of the current research, optimistic strategy implies aggressive approaches to attain survivability, which then is associated with the business or technical requirements of both the CC and CSP as stipulated in an SLA.

Pessimistic: a survivability decision-making parameter key selects for each action, the worst possible outcome, and then select the action that has the best worst outcome. As Yader (2003) suggests, a pessimistic strategy tends to conservative behaviours as the decision-maker is inclined to view outcomes of actions as unfavourable to them (Yager, 2003). Intuitively, in the context of the current research, this strategy implies passive survivability actions towards survivability.

Neutral: also referred to as the normative strategy, this is when a survivability decision-making parameter selects an average of all outcomes of action and then selects the actions with the best average.

Performability is the property of agents such that it meets its level of use, for instance, as described by measures such as selfish or cooperative, or by the quality of service (QoS) measures such as packet delivery delay (Sterbenz *et al.*, 2010a).

5.2.2 Survivability mission requirements

Survivability requirements significantly vary based upon the survivability target, the scope, objective, the risk or cost-benefit analysis of the cloud system. Survivability targets encompass contextual problem domains, scenarios, domain boundaries and some preconditions necessary for abstraction and modelling (Wang *et al.*, 2012a). For instance, critical infrastructures such as healthcare services in contrast to services such as Netflix. Clearly in the former, survivability requirement definition could include the cost and criticality of survivability failure whereas Netflix could include objectives and operation consideration.

In this thesis, requirements are defined according to how they are presented in the survivability mission, i.e. SLA. For instance, a new component, e.g. "survivability model", may be defined with attributes that model survivability sub-components, whereupon each model implements a function. As an example, the getCharacteristic (int/boolean key), defines the attitudes of a decision system when in choosing survival actions, e.g. *getCharacteristic (neutral)* defines a neutral survivability attribute, *getCharacteristic (optimistic)* defines an optimistic survivability attribute, *getCharacteristic (pessimistic)* defines a pessimistic survivability attribute, etc. Survivability mission parameter keys, i.e. neutral, optimistic, pessimistic, etc. act as unique identifiers corresponding to a specific survivability characteristic. In-depth details and application of the decision attitudes mentioned are presented in Section 6.4. Considering all survivability constraints, primarily the unpredictability introduced by UUURs, survivability decision-making is a critical requirement to guide survivability choices under uncertainty.

5.2.3 Survivable system requirements

This section introduces key requirements employed for the adaption concept of survivable cloud, environment, behaviours, actions and adaptation solutions. IaaS environment: The shared cloud environment for survivable agents, cloud services and resources which mediates how agents access services. Within the environment, there are general rules for accessing resources; these act as constraints at various levels of the environment. A cloud environment is defined according to the following tuple:

$$E = < R, N >$$

, where R, are resources accessible to agents (agents can perceive the state of resources in pursuit of their survivability goals) and N is being the rules or behaviours that impact of the resources.

VM resource: Resources, i.e. virtual machines (VMs) are the elements of a cloud environment which inform the building blocks of IaaS cloud. Hence, VMs are considered as passive objects with states and operations. In addition, attributes (*attr)* represent a set of attributes that define the state of a VM. Moreover, operations (*op*) are a set of operations defined over the attributes. VM resources are thus defined as:

$$r \in R = < attr, op >$$

where a set of attributes, *attr*, define the state of resources, and *op* is the set of operations over the attributes.

Environmental state: The state of IaaS cloud environment is determined by the state of VMs. Supposing that S is the set of possible states an IaaS environment may exist in and S_r represents the state of resource r ∈ R. The state of an IaaS environment s ∈ S, is, therefore, a conjunction of the states of all VMs, such that:

$$s \in S = \bigcup_{rR} s_r$$

, where $S_r=\{<attr_i , v_i>|attr_i \in r.\ attr,\ v_i \in dom\ (attr_i)\}$

In this case, survivability agents are capable of manipulating VMs through executing VM operations which impact upon the IaaS environment. Consequently, an IaaS environment can exert constraints on survivability agents thereby managing access agents have on VMs.

Norm of accessing a VM: A constraint (environment norm) stipulates instances where the environment permits or prohibits agents' manipulation of VMs (resources), such that:

$s \rightarrow Ags\ can\ do\ (r.op_i)\ to\ degree\ of\ pd$

, where $s \in S, r \in R\ and\ op_i \in op$. *Ags* is a set of agents constrained by the "norm", and *pd* is the permission degree for which $pd \in [0..1]$. In this case, if there are no constrained agents, i.e. *Ags* = *0*, the norm applies to all agents in the environment. For instance, while VM is provisioning services, agents are prohibited from executing aggressive actions, unless all passive actions have been exhausted. Moreover, some environments, e.g. SLA stipulated, aggressive actions may be totally prohibited altogether. Thus, a permission degree, *pd*, in this case, would serve to stipulate the extent to which actions around the "norm" can be executed in agents. Where *pd* = *0,* are prohibited from accessing VMs, i.e. through executing the . Similarly, where *pd* = *1*, agents are permitted to access VMs by invoking .

For adaptation to exist, prey agents must adjust their behaviours to changes in the environment to maintain or improve survivability. Thus, an agent should decide or reason what current actions can be performed and decide on how to select and implement the best actions. Nonetheless, the following are considered as constraints to such actions:

1. Survivability agents cannot access forbidden VMs (resources)
2. Actions may or may not be independent of each other, resulting in three different types of dependencies.
    a. a_2 as a definite result of a_1
    b. a_2 as a conditional dependent of a_1, i.e. that a_1 should happen before a_2 happens
    c. a_2 is exclusively dependant on a_1, i.e. a_1 should not occur for a_2 to happen, else a_1 destroys the conditions for a_2's execution.

Behaviour patterns: These are specified by a set of the fixed execution sequence of actions. This research is primarily concerned with 3 sequential actions. Supposing a prey agent to possess survivability capacity, ***Cap****,* its behaviour pattern, ***Bp***, is defined as follows: $Bp \in Cap \times Cap \times Cap$

6: Action matrix

$$\mathrm{AM} = \mathrm{A}\ (\{\mathrm{E}\} \cup \mathbf{Cap})$$

$$\begin{array}{c|cccccc} & \mathbf{E} & \mathrm{pre}_{a_1} & \mathrm{pre}_{a_2} & \cdots & \mathrm{pre}_{a_m} \\ \hline a_1 & \mathrm{EC}(a_1) & \mathrm{DR}(a_1, \mathrm{pre}_{a_1}) & \mathrm{DR}(a_1, \mathrm{pre}_{a_2}) & & \mathrm{DR}(a_1, \mathrm{pre}_{a_m}) \\ a_2 & \mathrm{EC}(a_2) & \mathrm{DR}(a_2, \mathrm{pre}_{a_1}) & \mathrm{DR}(a_2, \mathrm{pre}_{a_2}) & & \mathrm{DR}(a_2, \mathrm{pre}_{a_m}) \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ a_n & \mathrm{EC}(a_n) & \mathrm{DR}(a_n, \mathrm{pre}_{a_1}) & \mathrm{DR}(a_n, \mathrm{pre}_{a_2}) & & \mathrm{DR}(a_n, \mathrm{pre}_{a_m}) \end{array}$$

Rows define actions in A and the first column are the strength of environmental constraints. The remainder is identified by the dependency degree (***dp***) between preceding action and optional actions. Thus, EC(a_i), which is the action matrix of an arbitrary action relative to the environment constraints, i.e. AM[a_i][E] where $a_i \in$A, to identify the constraints strength of the environment on an arbitrary action, a_i, and EC(a_i) is the minimum of the currently effective permission degrees for a_i.

To identify dependency degrees between a preceding action, pre_a_i, and an optional action, a_i, $DR\left(a_i, pre_{a_j}\right)$, is defined as the action matrix of an arbitrary optional action in relation to an arbitrary preceding action such that $AM[a_i]\left[pre_{a_j}\right]$. Thus:

$$\mathrm{DR}\left(a_i, \mathrm{pre}_{a_j}\right) = \begin{cases} 0 & \text{if } a_i \text{ is exclusive to } \mathrm{pre}_{a_j} \\ 1 & \text{if } a_i \text{ is chained to } \mathrm{pre}_{a_j} \\ x(> 0.5) & \text{if } a_i \text{ is conditioned by } \mathrm{pre}_{a_j} \\ x(> 0.5) & \text{if}\left(\mathrm{pre}_{aj}, a_i \text{ is a recommended behaviour pattern}\right) \\ x(< 0.5) & \text{f}\left(\mathrm{pre}_{aj}, a_i \text{ is not a recommended behaviour pattern}\right) \\ 0.5 & \text{otherwise} \end{cases}$$

Adaptation solution: If agents build a set of action matrix, $AM = \{am_1, am_2 \ldots am_n$, to

achieve survivability goal, in which a series of actions have been selected and executed, i.e. $T = \{a_1, a_2 \ldots a_n$ in which $a_i \in am_i.A(i = 1..N)$. T is the adaption solution for achieving the goal at hand, whereupon T is said to be a good or bad solution dependent upon what solution T reaches, i.e. positive adaptation or negative adaptation, respectively.

5.3 Security systems design

To contextualise the designs in these chapters, this thesis distinguishes the following three categories of threat event as important:

- Critical threats, which directly impact survivability outcomes. For instance, the availability of resources which is critical to service provision. Similarly, the cost of executing a countermeasure over another is critical to the choice of action.
- Objective threats, which can be numerically assessed. For instance, the rate of compromise in cloud computing as analogous to the death rate in nature.
- Subjective threats, which are informed by expert perceptions and judgements and deemed to be qualitatively reliable.

While the threat concept above is not a one-size-fits-all approach to threat classification, it is an important reminder that cloud environments and indeed security events will suffer from unique risks, threats, vulnerabilities, risk tolerances, etc. Moreover, further in-depth analysis and discussion of specific threats are outside the scope of this research. Thus, for purposes of this research, the following assumptions are made:

- Cloud environments change synonymous to nature; agents die, are born, hibernate, extinct, etc.
- Agents are cognitive or passive with predefined behaviours, or behaviours adapted through evolution. However, for real implementations, adaptive autonomous agent as most suitable.
- Interactions among autonomous agents are generally diverse due to the diversity of agents. Most interesting are the new unknown capacities that exist out of such interactions, whether among social agents themselves, among competing agents or among agents and the environment.
- Self-organisation is key for the collective objectives of cooperating agents. The organisation thus can extend to how roles and hierarchies are defined and structured, whether pre-defined or emerge from inter-agent interactions. For the

purposes of this research, consider the role of the human agent (cloud administrator or adversary) as one who can control cloud environments; intervene with, control or update the agent database (correctively or maliciously).

Based on the foregoing, a perceived malicious and sophisticated agent capable of assessing risk and exploiting vulnerabilities of a system as observed by (Mehresh & Upadhyaya, 2012). An attacker can covertly perform reconnaissance and gain reasonable access into a compromised system undetected. If detected, the attacker has an advanced contingency plan for further actions, including aggressive and catastrophic destruction of data. Two attributes of Pi-CSF ought to be emphasised from the design perspective. Foremost, the survivability mission, which in this thesis is closely coupled with both the CSP and CC's missions, is defined to be adaptable to an evolving SLA. The mission thus directly impacts upon the survivability requirements. Another attribute pertains to specific dynamic constraints that impact upon a survivable cloud system. Hence, this attribute is defined to incorporate feedback, which informs the usability of an implemented architecture and its adaptability to evolving survivability requirements and tread-offs.

5.4 Prey-inspired cloud computing survivability framework (Pi-CCSF)

In the traditional context, survivability requirements are predictable or static goals which do not change over time. Pi-CCSF is therefore implemented in IaaS cloud environments to provide user-level access to influence adaptable service level agreement with the CSP. The modular design enables such components as the survivability service definition to be adjustable according to negotiated service levels or evolving survivability requirements. For instance, an arbitrary organisation requiring predictable and unpredictable capacity to cope with a seasonal and sporadic demands, a survivability SLA can be defined which stipulates how Pi-CCSF is able to manage and accommodate undefined or evolving survivability requirements during peak times. Besides this flexibility, it is possible to manage interoperability issues in the user-space, where any failures are simply isolated and confined within that user-space. Pi-CCSF is illustrated and Figure 50 and described below.

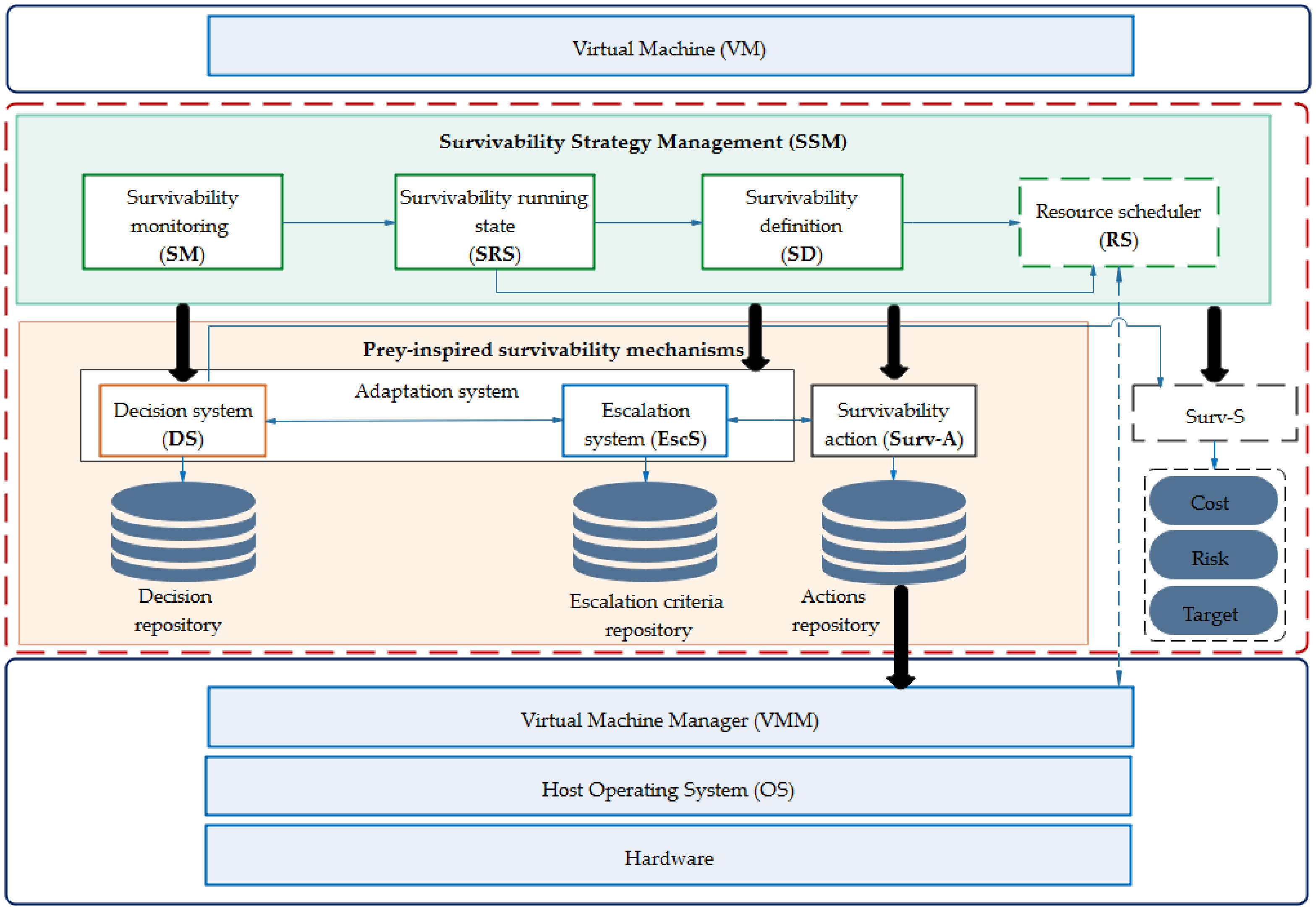

Virtual Machine (VM)
Survivability Strategy Management (SSM)
Survivability monitoring (SM)
Survivability running state (SRS)
Survivability definition (SD)
Resource scheduler (RS)
Prey-inspired survivability mechanisms
Adaptation system
Decision system (DS)
Escalation system (EscS)
Survivability action (Surv-A)
Surv-S
Cost
Risk
Target
Decision repository
Escalation criteria repository
Actions repository
Virtual Machine Manager (VMM)
Host Operating System (OS)
Hardware

Figure 50. Prey-inspired cloud computing survivability framework (Pi-CCSF).

By implementing Pi-CCSF, an SLA is leveraged by gathering knowledge of the state of user services, i.e. virtual machine; running state, performance, expected survivability, running survivability configuration and the resources required. Moreover, a Survivability strategy manager (SSM) provides an interface with the prey-inspired survivability mechanisms which can be configured according to a cloud user's survivability requirements. For instance, aggressive action-based escalation for critical systems. SSM remains operational throughout the lifecycle of service deployment to update SLA and adapt to changing survivability requirements at run-time. Two essential components of Pi-CCSF are:

- High-level survivability management layer in which the Survivability Monitoring (*SM*), Survivability Running State (*SRS*), Survivability Definition (*SD*) and the Service Scheduling (*SS*) form the key components of the survivability strategy management (*SSM*).
- Low-level survivability processes layer in which the adaptation system (decision system (*DS*), escalation system (*EscS*) and survivability actions (*Surv-A*)) are key components of the low-level prey inspired survivability mechanisms.

Low-level processes act towards ensuring survivability through implementing mitigations for prevention and resistance, fault-tolerance and recovery while enabling adaptation through intelligent decision-making and implementing escalating actions. The adaptation concept is an essential process to cope with cloud computing's dynamic changes, while escalation introduces a novel approach to enhance survivability by dynamically selecting a range of counteractions according to a prevailing security event. The decision-making technique employed in Pi-CCSF is covered in-depth in Chapter 6.

5.4.1 Survivability strategy management (SSM)

To enable survivability management at the user-space, SSM facilitates a users' direct involvement in defining, maintaining and updating survivability requirements. This is a major addition to traditionally CSP-oriented management in which the user's involvement is limited to the functional aspect of a system's survivability (Ellison *et al.*, 1997b). Within an IaaS cloud, the SSM presents an SLA-based strategy management system to facilitate service definition, monitoring, resource scheduling and tracking of a reusable running service state. Figure 51 illustrates the high-level processes involved in managing the survivability strategy in cloud computing environments.

While resource scheduling is not a new concept, its specific implementation details and challenges are outside the scope of this research. Nonetheless, several works in the literature suggest evolutionary computing (EC) algorithms for real-time scheduling, adaptive dynamic scheduling, large-scale scheduling, multi-objective scheduling and distributed and parallel scheduling (Zhan *et al.*, 2015). For the supposes of this Chapter, SSM's high-level operation will be discussed as two-fold. On one hand, it is aimed to augment the user's survivability objectives, i.e. quality of service (QoS) through augmenting processes such as resource scheduling, service definition, monitoring, user cost, etc. On the other hand, SSM enables the CSP to maximize survivable service provision by incorporating CC specific requirements. These activities and services that are common across cloud platforms to enable the managing (developing and updating) of essential survivability services. Through an SLA-linked profile, a CC can align and prioritise the survivability requirements and activities according to the following:

Step 1: SSM receives a trigger; in this research this is imposed by a UUUR event. If the event is unknown, SSM captures the event signature to update existing survivability processes, otherwise continues to step 2.

Step 2: If an event is known, SSM invokes SRS and maps the event to the current survivability running state. If an event meets an expected UUURs profile, SSM proceeds to step 5, otherwise goes to step 3.

Step 3: If the event does not meet an expected UUUR profile, SM maps the event to the current SLA to assess compliance. If compliance with expected survivability is not met, it is necessary to update existing survivability processes, otherwise goes to step 4.

Step 4: If expected survivability compliance is met, survivability definition enables predictive adjustments to survivability parameters and future states, i.e. with updated resource requirements and schedule, otherwise goes to step 5.

Step 5: If SD fails to yield, SSM passes current survivability state information, including UUUR data to the low-level prey-inspired survivability mechanisms.

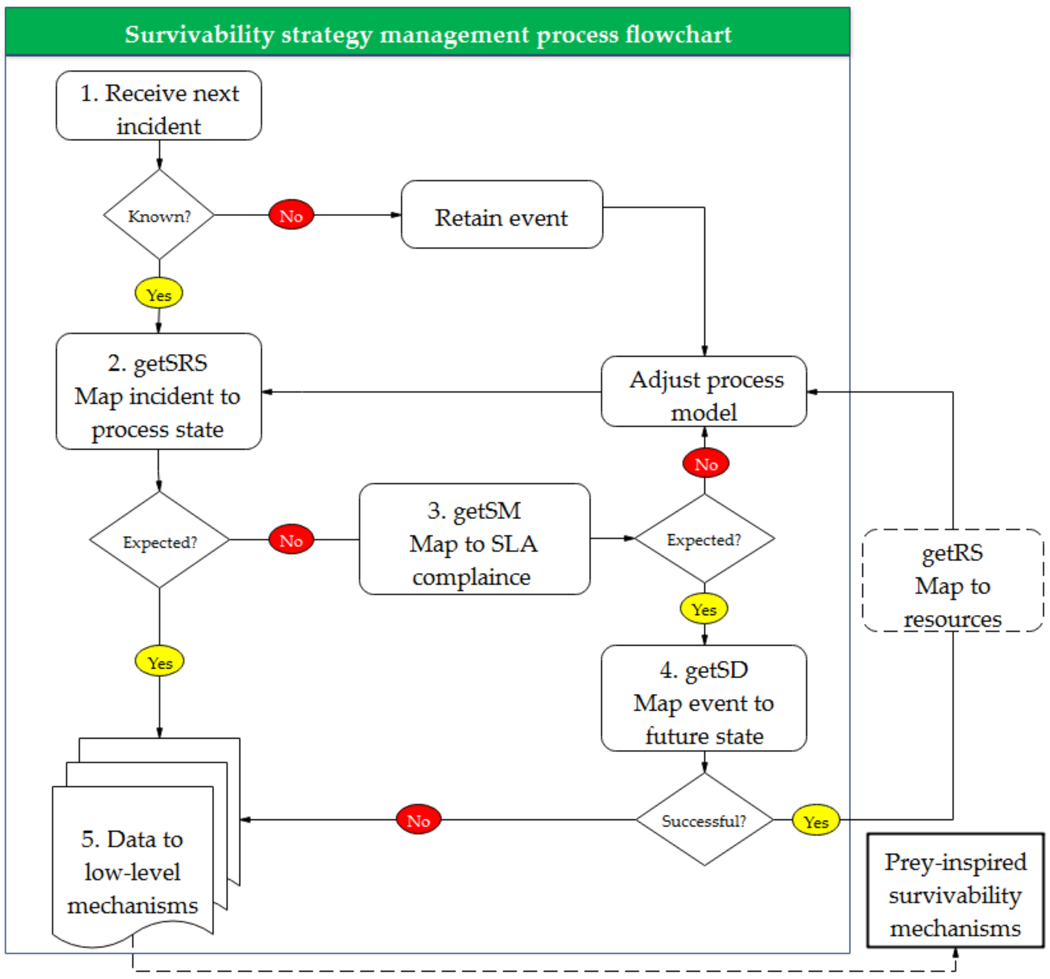

Survivability strategy management process flowchart
1. Receive next incident
Known?
No
Retain event
Yes
2. getSRS Map incident to process state
Adjust process model
No
Expected?
No
3. getSM Map to SLA complaince
Expected?
Yes
getRS Map to resources
Yes
4. getSD Map event to future state
5. Data to low-level mechanisms
No
Successful?
Yes
Prey-inspired survivability mechanisms

Figure 51. Survivability strategy management (SSM) process flowchart

5.4.2 Survivability monitoring (SM)

When SSM is captures deployed, monitoring the resource and UUUR-to-resources data. Monitoring encompasses learning and evaluating UUUR-to-resources inventory and the output survivability around the following survivability objectives: (i) enable efficient aggregation of survivability data, (ii) survivability risk and cost analysis and (ii) survivability target analysis. Monitoring is, therefore, a functional character of learning and thus, in practice implement functions such as machine learning algorithms to detect and predict uncertain changes in a cloud system.

Three parallel processes; survivability data aggregation, risk/cost analysis and survivability target analysis, are computed for two main purposes. Foremost, to evaluate the current survivability SLA requirements with respect to an ensuing incident, or to evaluate a survivability target and define survivability objectives for an existing or negotiated SLA. To address similar uncertainty imposed by UUURs, some works have shown that model-based evolutionary algorithms (EAs) present efficient learning abilities with low computational expenses (Cheng *et al.*, 2018). A CSP can manage and adapt a CC's changing survivability requirements and activities.

Figure 52 illustrates the foregoing survivability monitoring processes, which will be itemised below.

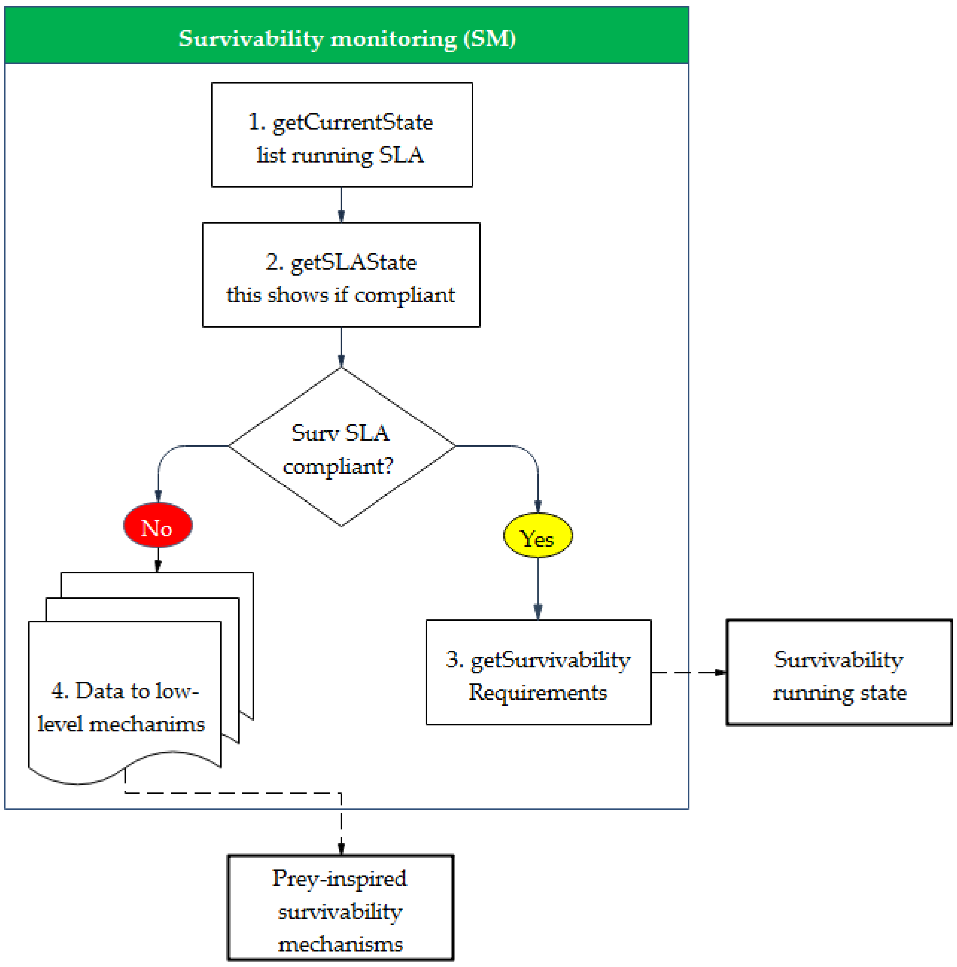


Figure 52. Survivability monitor (SM) process flowchart.

Step 1: After deployment, SM receives the currently running survivability SLA and evaluates compliance

Step 2: Monitors if the cloud environment is compliant.

Step 3: If compliance requirements are met, survivability requirements are retained and updated in the SRS, otherwise continues to step 4.

Step 4: If compliance requirements are not met, SSM passes current survivability state information, including UUUR data to the low-level prey-inspired survivability mechanisms.

5.4.3 Survivability running state (SRS)

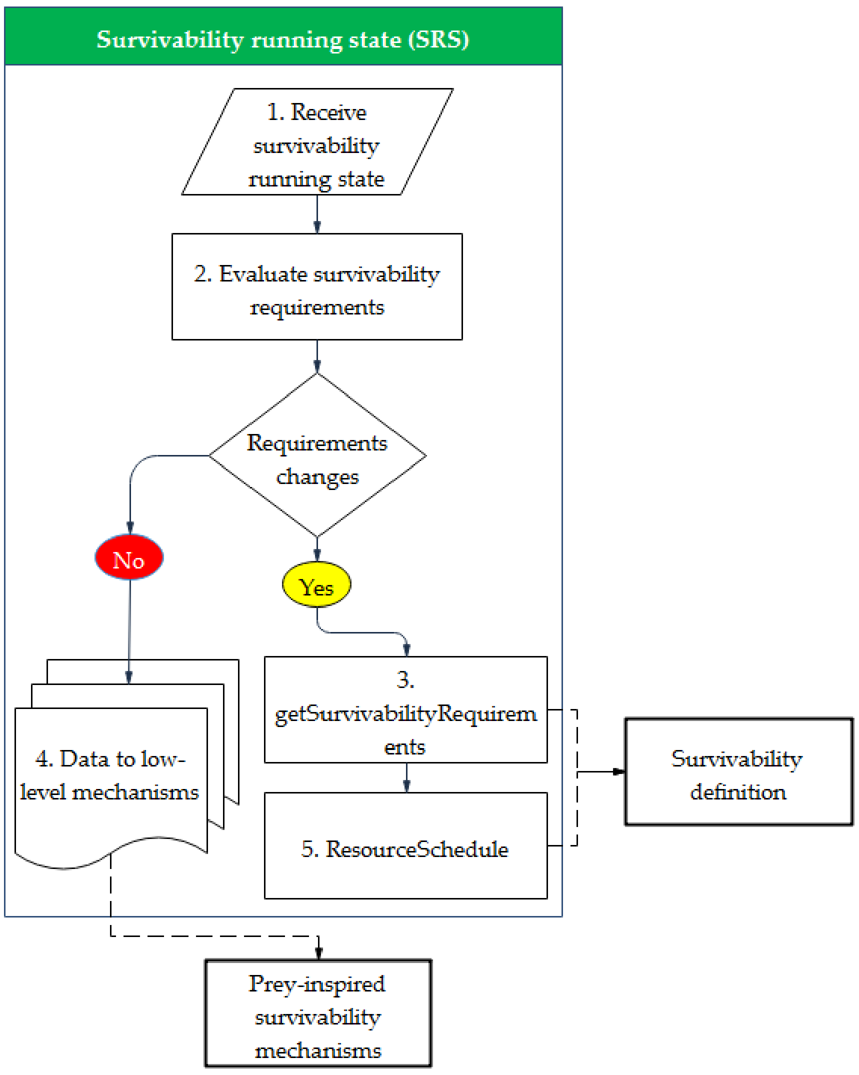
Survivability running state (SRS) captures the current survivability QoS configuration and updates its state according to SM. SRS activities hence specify the survivability parameters such as the survivability threshold, survivability target and/cost, expected survivability, etc. By default, these are typically specified in an SLA according to the user's specifications. Changing SRS can be done automatically to the running configuration based upon detected input from the SM. Alternatively, changes to the SRS can be done directly when SSM is deployed and an SLA is negotiated or renegotiated. An example is Amazon's AWS ConFigurerules which tracks the survivability service state to enable dynamic and flexible launch, use and termination of processes (Jeff, 2015).

Functionally, it should be noted that a survivability running state is only computed with known survivability requirements, which can be updated in run-time. Based upon the survivability running state and survivability objectives, low-level adaptation mechanisms; decision-making and escalation, are executed according to a specific use-case. For instance, where a survivability target is known. The details of applying survivability targets for decision-making under varied use-cases is discussed in section 6.4. Figure 53 illustrates the foregoing survivability running state processes, which will be itemised below.

Step 1: SRS receives running state data or survivability requirements from SM.

Step 2: Survivability requirements are evaluated against the current survivability SLA. If SRS yields known/expected requirement changes, it continues to step 3

Step 3: SRS gathers survivability requirements that satisfy an expected running configuration, updates the survivability definition (SD) processes and proceeds to step 5, otherwise continues to step 4.



Survivability running state (SRS)
1. Receive survivability running state
2. Evaluate survivability requirements
Requirements changes
No
Yes
4. Data to low-level mechanisms
3. getSurvivabilityRequirements
5. ResourceSchedule
Survivability definition
Prey-inspired survivability mechanisms

Figure 53. Survivability running state (SRS) process flowchart.

Step 4: If SRS yields unknown/unexpected requirement changes, SSM passes current survivability state information, including UUUR data to the low-level prey-inspired survivability mechanisms.

Step 5: Enables the scheduling of required resources to meet survivability requirements in step 4 and update SD processes.

5.4.4 Survivability definition (SD)

The ability to define survivability requirements (for both the cloud user and CSP) ensure that survivability is established according to deliberately defined survival QoS metrics and to desired probabilities. This concept has been described by (Yallouz, Rottenstreich & Orda, 2014) and (Yallouz & Orda, 2017) as tuneable survivability, which specifies a quantifiable measure of survivability. Figure 54 is an illustration of the survivability definition process flow. It highlights the processes involved in combining survivability knowledge and survivability QoS guarantees in existing or new SLAs to produce tuneable survivability. Hence Figure 54 illustrates survivability definition processes for data aggregation, including risk or cost analysis, assessing fuzzy survivability data, etc., as minimum requirements for end-to-end survivability QoS and SLA guarantees. For argument's sake, these represent structured activities and computations for information assisted survivability problem-solving. SD processes outputs include data for low-level mechanisms processing and resource scheduling support SD's objectives.

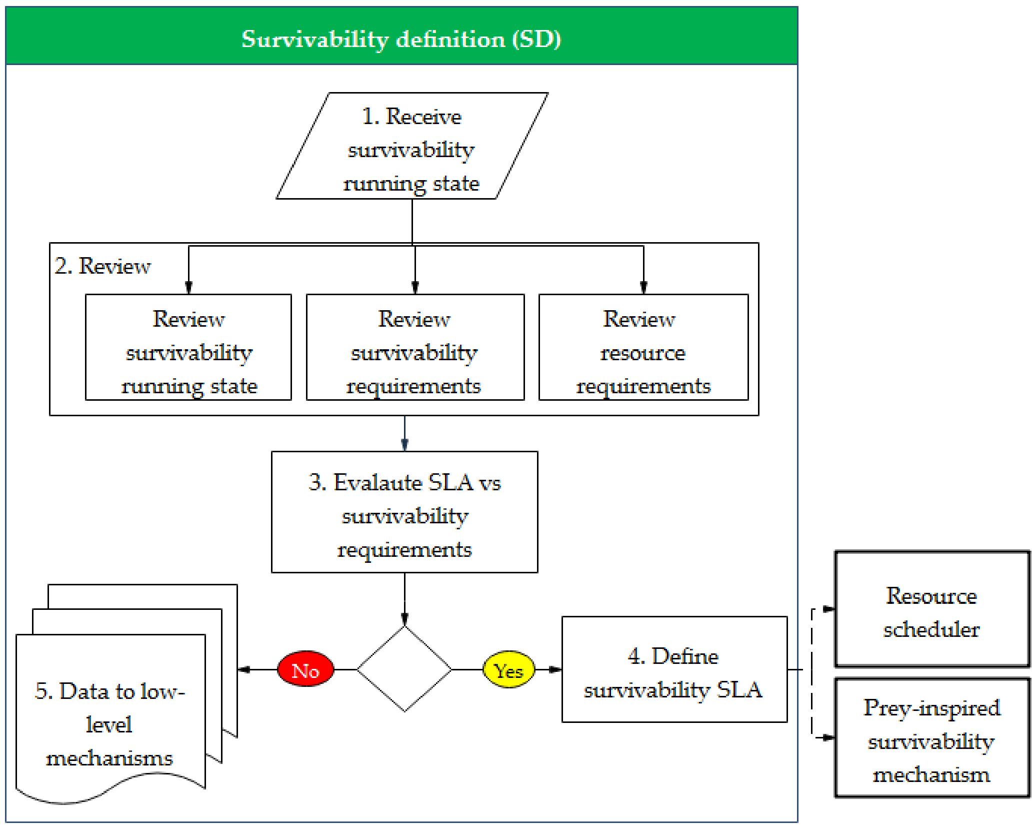

Survivability definition (SD)
1. Receive survivability running state
2. Review
Review survivability running state
Review survivability requirements
Review resource requirements
3. Evalaute SLA vs survivability requirements
No
Yes
4. Define survivability SLA
5. Data to low-level mechanisms
Resource scheduler
Prey-inspired survivability mechanism

Figure 54. Survivability definition (SD) process flowchart.

Step 1: SD receives SRS information including existing QoS and survivability requirements

Step 2: Aggregates SRS and user requirements along with resources to define the tuneable survivability requirements.

Step 3: To improve these processes, QoS requirements in an SLA are evaluated against the user's or system's service requirements, if these are achieved, the SD proceeds to step 4, otherwise, it goes to step 5.

Step 4: Define tuned survivability and invokes the resource scheduler Step 5: Otherwise if not achieved, SSM passes current survivability state information, including UUUR data to the low-level prey-inspired survivability mechanisms.

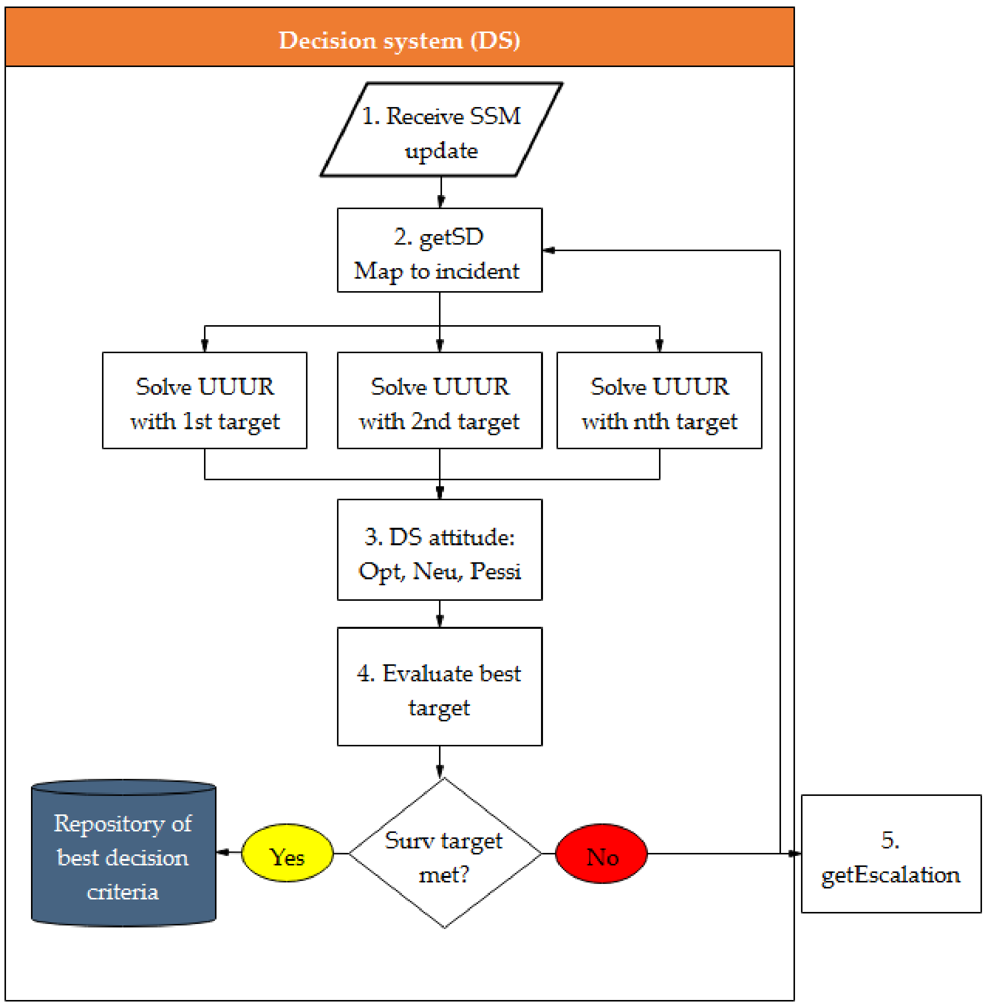
5.5 Prey-inspired survivability mechanisms

This low-level layer implements survivability mechanisms according to requirements obtained from SSM. Prey-inspired survivability mechanism's core components; adaptation system which consists of a decision-making sub-system and the escalation system, implements survivability actions which can be adapted and prioritised to meet a defined cloud service or system's survivability target. Since Pi-CCSF is implemented upon known cloud computing frameworks such as the CloudSim framework (Calheiros *et al.*, 2009), these low-level mechanisms are therefore organised around common cloud computing structures and standards to embed the prey-inspired survivability concepts. UUUR and system state data from SSM and the survivability method form the input and output to the workflow system, respectively. Within the workflow system, survivability data aggregation, risk and/or cost analysis, decision (fuzzy information assessment), etc., represent structured activities and computations for information assisted survivability problem-solving. Internal processes outputs such as the survivability requirements, survivability criteria and survivability objectives describe directed graphs connecting coupled survivability components.

While the mechanisms presented below are deliberately limited for demonstration purposes, the prey-inspired survivability mechanisms module encompasses a registry of other survivability nuances (including sub or sub-sub-parameters) that can be queried for synthesis or processing. CSPs can publish and update the registry with new survivability parameters, including those acquired through SLA negotiations. An example is where a service configuration with specific survivability or resource requirements is defined and retained for reuse on the fly. By combining the survivability configuration and registry-stored survivability parameters, low-level synthesis generates specifications for survivability coordination entities (SCEs) through transforming existing models. Functionally, SCEs are software entities that match a predefined survivability criterion, for instance, a predefined SLA, CC specification, architecture, etc. and manage, i.e. coordinate and/or enforce interactions among the participating entities. Alternatively, SCEs may also simply represent as an algorithm that solves mathematical functions towards a survivability goal, for instance, counterattack in prey presented in (Waltman, Braselton & Braselton, 2002) (Rozenfeld *et al.*, 2006). Moreover, SCEs may be executable procedure rules or other multi-agent models such as genetic algorithm (Shon, Kovah & Moon, 2006) that describe the inner workings of survivability agents.

5.5.1 Decision system

Although decision-making is traditionally not considered as a "first-class" concern (Cámara *et al.*, 2018) for managing computer networks' security and survivability, it is critical to handling unpredictability imposed by UUURs in complex environments. General control theory can be applied for decision strategies to minimising the cost function (or maximizing reward function) in evolving dynamic systems (Kreidl & Frazier, 2004a). In the presence of UUURs, unpredictability is such that it is operationally challenging to predict a system's state at any instant, which renders general control theory unsuitable due to the impossibility of capturing information on how the state of cloud systems are observed or how they evolve in time. The control concept describes a qualitative capacity of a measure to affect a true state (Kreidl & Frazier, 2004a), which implies that an uncontrollable state is one whose evolution a control cannot predictably affect. On the contrary, complete control suggests the capacity of control to affect the true state and subsequently all states that evolve thereafter. The observation element of UUURs, on the other hand, refers to how much information system detectors possess, in ration to a true state estimate. Thus, an unobservable state implies that there is limited information to estimate a true state, as opposed to total observability which implies the possession of complete information to estimate a true state. Communication is the processes that to some degree, enables the complete control and observation, a finite series of actions and a finite series of observation. Figure 55 illustrates the decision system processes, which are described below.



Decision system (DS)
1. Receive SSM update
2. getSD Map to incident
Solve UUUR with 1st target
Solve UUUR with 2nd target
Solve UUUR with nth target
3. DS attitude: Opt, Neu, Pessi
4. Evaluate best target
Repository of best decision criteria
Yes
Surv target met?
No
5. getEscalation

Figure 55. Decision system (DS) process flowchart.

Step 1: The decision system receives instructional data from the survivability strategy manager. Functionally, it assumed in this research that prey-inspired mechanisms are only invoked as required, to maintain other important survivability tread-offs such as cost.

Step 2: DS maps SSM-based data, particularly the incident in question, to available survival processes. In this research, this implies mapping data from SMM to a target-based technique; the appropriate mechanism to achieve a survivability target. The details for the target-based technique are provided in Chapter 8.

Step 3: Once an appropriate mechanism is mapped with the SSM data, the DS also factors into consideration, the "attitude" of the decision-making system. The notion of an "attitude" in this case emphasises the different priorities that capture the level of survivability a decision process must consider, as stipulated in an SLA. Section 5.1 briefly introduces pessimistic (Pessi), neutral (Neu) and optimistic (Opti) attitudes noted above.

Step 4: Based upon the preferred target-solution (Step 2) and the preferred attitude (Step 3), the DS evaluates the best decision method to meet the prescribed target. If an evaluated decision method meets an expected survivability target, this is implemented to address the current incident, otherwise, DS goes to step 5.

Step 5: If an evaluated decision method does not meet an expected survivability target, the DS invokes the escalation process.

5.5.2 Escalation system

As introduced earlier, one of Pi-CCSF's objectives is to present the user with additional control over defining low-levels mechanisms that can be implemented to meet their preferred survivability outcomes. Figure 56 illustrates the escalation process.

Step 1: The escalation system receives instructional data from the survivability strategy manager and invokes escalation

Step 2: An escalation criterion; defined according to the number and order of actions, is initialised with respect to step 1

Step 3: ES invokes the decision system and corresponding survivability targets and evaluates if current actions can address the current incident and achieve the expected survivability target. If yes, ES continues to "listen" for new updates from SSM, else proceeds to step 4.

Step 4: If the survivability target is not met, ES implements and executes variable escalations and evaluates if the target is met

Step 5: Based on step 4, the DS considers the context of the current incident and evaluates the effects of evolving escalation. If the target is not met, ES goes to step 4, otherwise the DS retains the best escalation criteria for the current context.

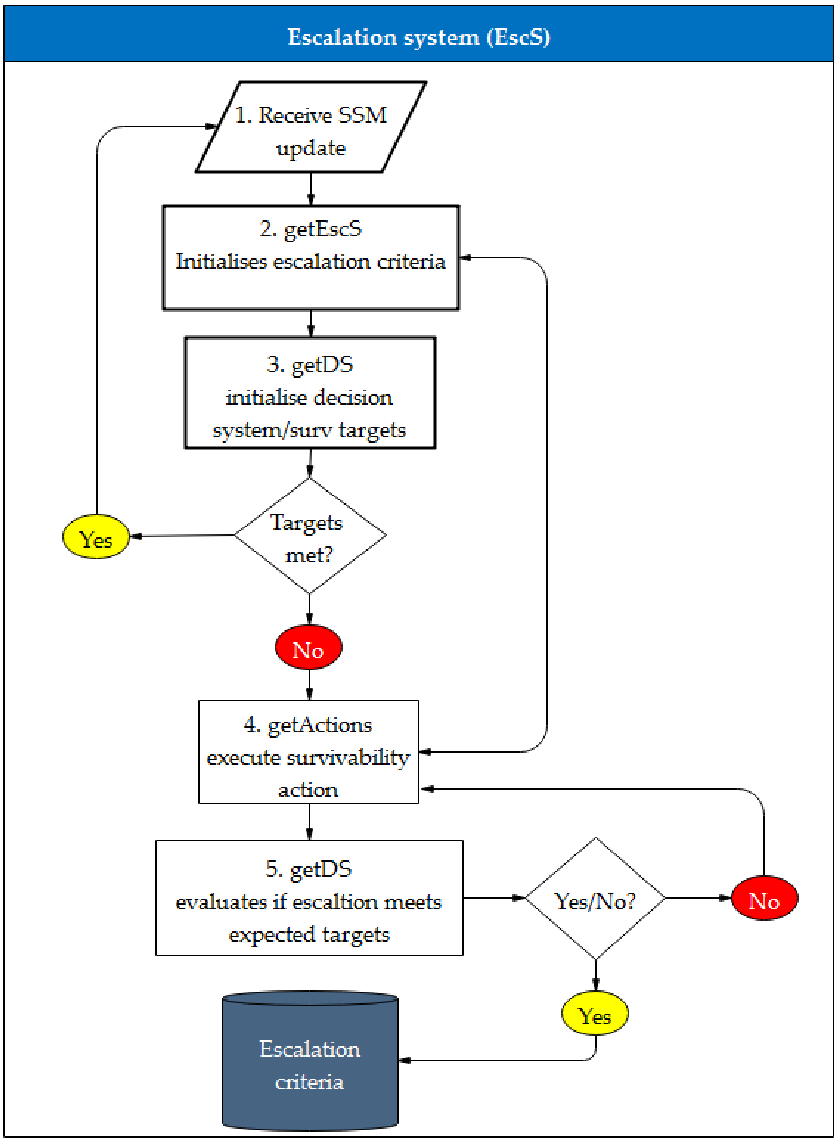

Escalation system (EscS)
1. Receive SSM update
2. getEscS Initialises escalation criteria
3. getDS initialise decision system/surv targets
Targets met?
Yes
No
4. getActions execute survivability action
5. getDS evaluates if escaltion meets expected targets
Yes/No?
No
Yes
Escalation criteria

Figure 56. Escalation system (EscS) process

5.6 Pi-CCSF's overall survivability process flow

When Pi-CCSF is deployed, SLA related information is received into the SSM via an interface such as Web API, enabling both the CSP and CC or user to access the SSM module. A context module builds the relevant information from the CSP administrator or CC's point of view. Hence, resources can be scheduled for specific actions as required by the CC or the CSP or an ongoing incident. Such information as the current survivability policy, SLA, the CC's profile and the level of sensitivity or value of data, the profile of service, the actions available and the repository for backup, if need be. Operationally, some Pi-CCSF processes are "private" to the CCs, providing them control to survivability processes. Figure 57 illustrates the overall process flow for Pi-CCSF. SSM data is passed into the prey-inspired survivability mechanisms, including the DS. Below is a detailed process flowchart for survivability management using Pi-CCSF. Each step is described below and where necessary, with examples provided where necessary.

Step 1 getSLA: When an unexpected event alert (UUUR), this step obtains an existing SLA for evaluation. VM provisioning, with request specific resources and capacity, should, therefore, be efficient, considering their utility usage. SLAs negotiated between CSP and CC stipulate among other things, the expected survivability of services, which inadvertently implores CSPs to perform survivability evaluations of their infrastructure (Longo et al., 2011), at least prior to service deployment. Some examples of evaluation techniques include models, frameworks (Sterbenz et al., 2010b), fault tree (FT), a reliability block diagram (RBD), CTMC or stochastic models (Longo et al., 2011), (Nguyen, Kim & Park, 2016), etc. Currently, most evaluation approaches among CSP specifically assess availability, reliability of cloud systems (Wagle et al., 2016), (Nguyen, Kim & Park, 2016).

Step 2 *Receive incident*: the incident is defined according to known or unknown signatures. For argument's sake, UUURs are defined in relation to their survivability QoS outcomes. In the IaaS model, CSP can assess and translate into guarantees, the availability, reliability of services based upon the incident. This is significant considering that some types of attacks, e.g. fraudulent resource consumption (FRC) attack results in unsuspecting cloud consumer incurring financial burden. (Hussain *et al.*, 2017).

Step 3 *getSRS:* If the event has expected outcomes, SRS maps the incident to the survivability process state for future behaviour prediction. Otherwise, if SRS does not meet expected outcomes, for the step 5. CHOReOSynt tool (Autili *et al.*, 2014) for instance, facilitates automated synthesis by enforcing reusable choreography goal to manage undesired outcomes. Since IaaS systems are in fact a composition of autonomous software services, a choreographic approach enables the modelling of autonomous software services by capturing pertinent interactions that maintain the global survivability form.

Step 4 *getSM*: Monitoring, learning and evaluating UUUR-to-resources inventory with the following survivability objectives: (i) to enable efficient aggregation of survivability data, (ii) to analyse survivability risk and cost and, (ii) to analyse survivability target. Monitoring is, therefore, a functional character of learning and thus, in practice implement functions such as machine learning algorithms to detect and predict uncertain changes in a cloud system. Approaches that facilitate cooperative problem-solving have been suggested to bolster cloud service composition (Sim, 2012). Given its centrality, cooperativeness requires a feedback mechanism to monitor and manage adaptation among composed services (Arcaini, Riccobene & Scandurra, 2017). (Weyns *et al.*, 2013) suggest control loops to typically enable the realisation of self-adaptation.

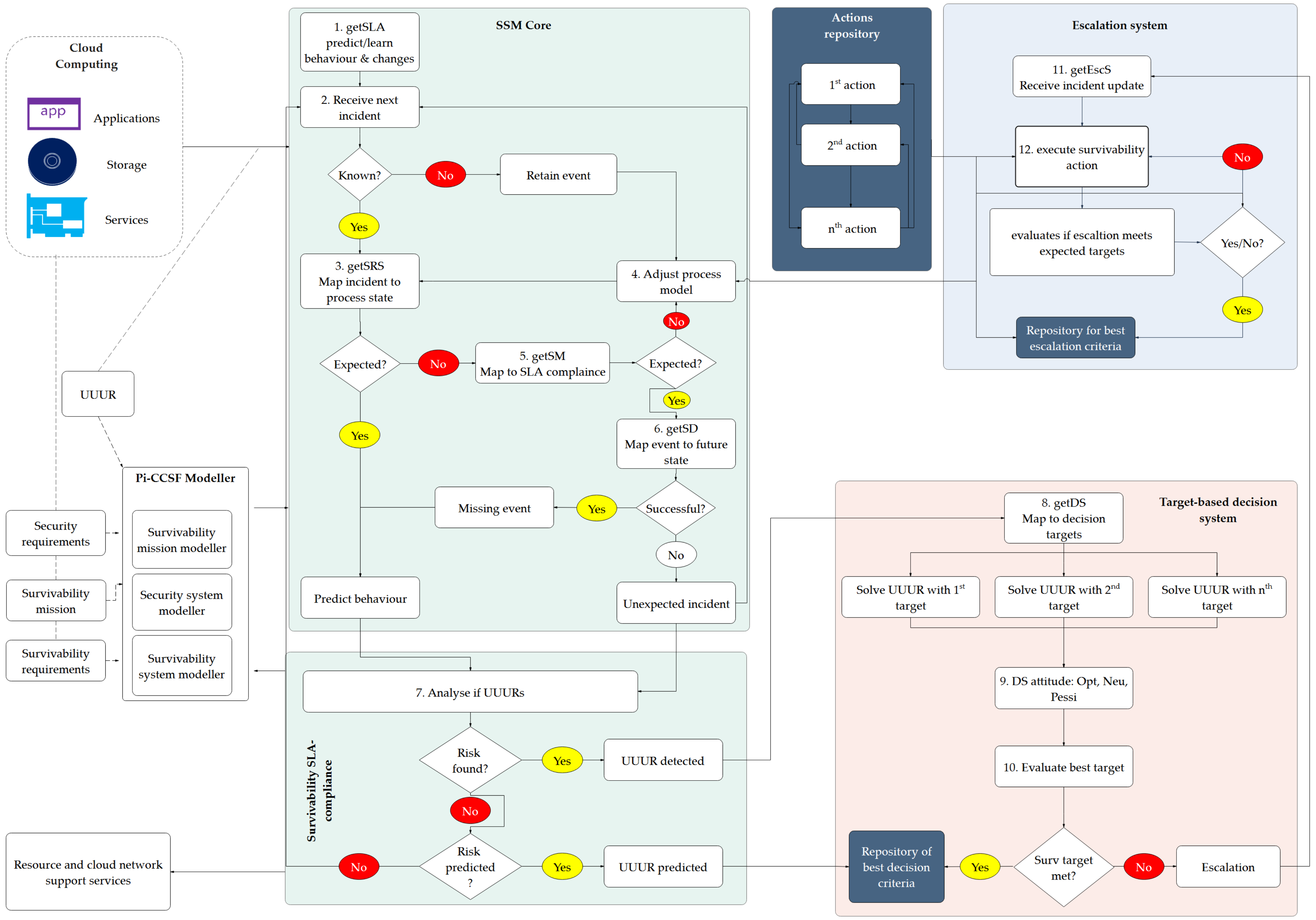
Step 5 g*etSD*: SD relies upon identified changes by SM at runtime. There is overwhelming evidence, for instance (Calinescu, Johnson & Rafiq, 2013), (Ahmad, Belloir & Bruel, 2015), (Calinescu *et al.*, 2013) and (Horikoshi *et al.*, 2012), that demonstrates the capacity for monitoring changing probabilities of a running systems and updating probabilistic models of these systems. In the current context, once SM processes are complete, obtaining results from SM processes are used by the DS to align or re-align compliance with SLA requirements, including QoS issues around availability or unavailability, costs, reliability, survivability, satisfaction, etc. In addition, obtaining SM processes results support the synthesis of a survivability reconfiguration plan. The significance of SD processes is to ensure that at runtime, survivability requirements persist despite changes.

Step *6 Analyse UUUR*: Survival analysis provides various tools to quantify the implications of UUURs. These include risk factors in affecting the overall survivability of a cloud system.

Step *7 getDS*: given a set of changes detected by the monitoring module, DS processes decide the best strategy for maintaining survivability. This includes selecting suitable actions to enable survivability evolution as prescribed by adaptation rules, available countermeasures and the results of the best result strategy. In a multi-agent system, for instance, DS also manages agents' communication to facilitate efficient cooperation around established survivability specifications and goals. Once this is achieved, the DS invokes EscS by providing it with new survivability specifications, otherwise, the decision process waits until the agent reaches a stable cooperative state.

Step *8 getDSAttitude*: a decision-making process that introduces the notion of a strategic model of choice under uncertainty (attitude), an important to a survivability mission and its requirements. Optimistic attitudes imply that a DS selects from a range of best possible outcome, an action with the maximum best outcome. As postulated in Bracha and Brown (2012), this strategy relates to decisions when desired future outcomes are at stake. For instance, a decision for using counterattack as a last resort to protect critical infrastructure (Grant, 2017). Pessimistic decision-making selects for each action, the worst possible outcome, and then select the action that has the best worst outcome. As Yader (2003) suggests, a pessimistic strategy tends to conservative behaviours as the decision-maker is inclined to view outcomes of actions as unfavourable to them (Yager, 2003). Neutral or normative attitudes are where a DS selects the best average from an average of all action outcomes.

STEP 9 getEscS: Managing interactions among low-level mechanisms or components is assumed around an automata-based approach; automaton's temporal discreteness and predictable changes according to predefined rules. The automata approach is used across many applications, such as to evaluate and select node queries on XML trees (Koch, 2003) and for understanding social dynamics (Hegselmann & Flache, 1998). More recently, the authors in (Calinescu *et al.*, 2017) propose an automata-approach to model the service interactions of self-aware systems in which survivability policies including recovery and escalating actions are dynamically configured and enforced. The escalation system accepts precisely defined computational rules from one process or system and selects the appropriate action. Therefore, it may be necessary perhaps, to compute a complete graph of a cloud computing system based upon SSM and DS data along with escalation parameters. Once computed, the possible state of the cloud system is critical to evaluate if an escalation criterion, for instance, meets survivability goals or not.



Cloud Computing
app
Applications
Storage
Services
UUUR
Pi-CCSF Modeller
Survivability mission modeller
Security system modeller
Survivability system modeller
Security requirements
Survivability mission
Survivability requirements
Resource and cloud network support services
SSM Core
1. getSLA predict/learn behaviour & changes
2. Receive next incident
Known?
No
Retain event
Yes
3. getSRS Map incident to process state
4. Adjust process model
No
Expected?
No
5. getSM Map to SLA complaince
Expected?
Yes
6. getSD Map event to future state
Yes
Missing event
Yes
Successful?
No
Predict behaviour
Unexpected incident
Survivability SLA-compliance
7. Analyse if UUURs
Risk found?
Yes
UUUR detected
No
No
Risk predicted ?
Yes
UUUR predicted
Actions repository
1st action
2nd action
nth action
Escalation system
11. getEscS Receive incident update
12. execute survivability action
No
evaluates if escaltion meets expected targets
Yes/No?
Yes
Repository for best escalation criteria
Target-based decision system
8. getDS Map to decision targets
Solve UUUR with 1st target
Solve UUUR with 2nd target
Solve UUUR with nth target
9. DS attitude: Opt, Neu, Pessi
10. Evaluate best target
Repository of best decision criteria
Yes
Surv target met?
No
Escalation

Figure 57. Overall process flowchart for survivability management using Pi-CCSFSTEP

STEP 10 getActions: at runtime, the SEE maintains the current survivability specifications to realize the survivability plan from the decision process. For this purpose, an instance of the algorithm, e.g. collective action algorithm (MURAT, 2015) buffers incoming communication. In addition, the actions module interacts with the SEE to reconfigure the architecture, e.g. hide VMs, deploy deception VMs, deploy poisonous VMs, as well as (re)establishing new dependencies. Finally, once adaptation terminates and survivability is agreed and re(established), pending requests are handled.

5.7 Discussion

In simple terms, Pi-CCSF should facilitate efficient survivability management and control from the holistic perspective of the cloud. While the application of cloud computing technologies generally enhances concepts related to survivability that is traditionally easily defined (e.g. resilience), survivability in the cloud, particularly considering UUURs, is challenging. Thus, Pi-CCSF pertains to how control and management of survivability can be handled at every level of the cloud, across multiple cloud assets, applications, single data centres or world-wide area networks (WANs). From a high-level view, Pi-CCSF builds upon the traditional survivability architecture by encompassing a software layer over the virtual network embedding components which maps virtual network resources to a physical substrate network.

To support survivability, conceptual design analysis is proposed to assess the Pi-CCSF's survivability mechanisms. Hence, the workflow outlined above can be extended into a tool for deploying the proposed framework. Although it is specifically designed for vendor risk management approach, CERT's V-RATE (Ellison *et al.*, 2002) serves a similar purpose. In this research, the stated tool serves to highlight key cloud components that must be assessed and addressed to meet a survivability solution. Table 13 is an example taxonomy tool for addressing the challenge of detecting UUURs. As suggested by the TRIZ parameters, enhancing efficient detection entails preserving the complexity of cloud computing. In this example, resolving this contradiction, i.e. improving detection while preserving the complexity of cloud computing, is achieved through incorporating the concepts of dynamics, preliminary actions and substitution. These concepts are defined under TRIZ's parameters (Domb *et al.*, 2011b) and summarised in section 4.4.1.2

To provide context, the following is a practical example. *Suppose an organisation or CC requires that their survivability SLA enables from time to time, survivability to be invoked by STEP 10. Improving the CC's survivability objectives points to changing the design attributes of the actions or algorithms that enhance adaptiveness; processes or the environment and ensuring that preliminary actions are timeously implemented.*

For each survivability objective under assessment, a Pi-CCSF component or approach that meets a suggested countermeasure is assessed based upon a category it can be used in, and one of three broad security categories proposed by the holistic taxonomy in section 3.4. In Table 13, the "category of use" identifies an area relevant to Pi-CCSF components and the general cloud survivability problem. The "expected impact area" identifies a cloud security category (according to the proposed taxonomy, i.e. The output of each survivability solution is envisioned to grow as more security "assessment and example", "category of use" and "expected impact areas" are introduced tied to real-world empirical data.

Table 13. Detection: Improve detection difficulty and preserve complexity

Assessment area	**Category of use**	**Expected impact area**
Dynamics: design the attributes (environment or process) to change to be optimal, make process adaptive		
Survivability management	STEP 1	CAT 0, CAT 1
Preliminary actions: Perform, before it is needed (either fully or partially). Design to act without losing time		
Mature survivability manager pre-designed for an update on the fly	STEP 1 STEP 2 STEP 3	CAT 0
Substitution: Introduce sensory, not mechanical, change from static to dynamic, unstructured to structured		
Adaptable feedback control loop. Running survivability service.	STEP 1 STEP 5 STEP 9	CAT 0 and CAT 1: algorithms and policy

Conceptually, the implementation of the detection concept summarised in Table 13 can be represented as Figure 58 below. The prey-inspired detection component controls the implementation of the detection system. This detection component is executed from instantiation, in part, directly or delegated in specific instances. Thus, the prey-inspired detection component is invoked by the prey-inspired survivability modeller, identifies the components to processes subtasks and transfers tasks with relevant configurations to the relevant components. It is key to note that "configurations" are extended from TRIZ's suggested solutions. For instance, for the "preliminary actions" subtask, the configuration entails "performing actions before they are needed (either fully or partially) and without losing time".

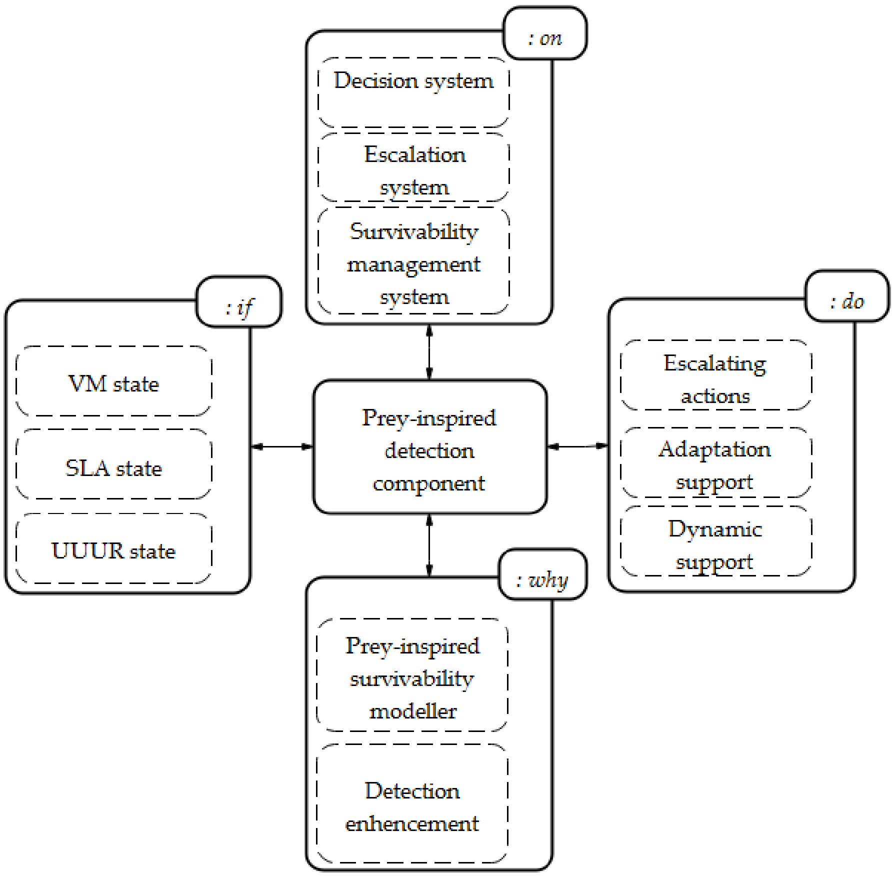


Figure 58. The conceptual components for detection

In Pi-CCSF presented earlier, the survivability strategy manager (SSM) processes the *"on"* aspect of the conceptual system, whereas the survivability monitor (SM) and UUUR analysis (Step 7) is abstractly presented and ought to be defined for precise processing. Alternatively, the *on* such as decision and escalation can be processed directly or triggered. The *if* aspects of the conceptual system are processed by different components. In the current example, these indicate different domain interests, for instance, virtual machine state changes, SLA changes and UUUR state. In fact, the *if* components provide dynamic awareness of the survivability state of the cloud environment. The *do* aspect of the conceptual system is triggered according to how the: *if* aspects are evaluated. For instance, the *do* components can be triggered to implement mitigation action and define the survivability criteria based upon an adaptation policy or dynamic parameter management. A prey-inspired survivability modeler is responsible for maintaining the survivability strategy while the detection component is useful in the *why* aspects of the conceptual system. In order to evaluate the conceptual approach above in practice, it will be necessary to implement all survivability solutions and gather several industry scenarios to provide requirements and guidelines. While this will help refine the conceptual methods proposed, this form of evaluation is outside the scope of this research.

5.8 Summary

In summary, the Pi-CCSF presented in this chapter aims at presenting prey-inspired survivability management and prey-inspired survivability mechanism at the conceptual and phase design phase. Specifically, the SSM conceptualised in section 5.3.1 is expressed around H1 of the research hypothesis whereas prey-inspired survivability mechanisms are expressed around H2 of the hypothesis outlined in the introductions to this research. In the work presented in section 5.2 , the design context for Pi-CCSF is presented to introduce the survivability mission and mission requirements, as well the system requirements towards this context of a survivable system. The work in section 5.2 briefly highlights the security design context upon which the survivability is conceptualised. The framework serves to address the issue of survivability design for cloud computing building upon a specific objective to support the development of prey-inspired solutions. Thus, the general objectives to enhance survivability; detection and monitoring, intelligent making decisions, escalation and escalating actions, integrated into the complete process flow.

The discussions in section 5.6 and the practical example presented give an indication of further noteworthy application directions which Pi-CCSF can be useful.

Chapter 6 Prey-inspired target-based decision-making technique (Pi-TBDM) for unpredictable survivability targets in cloud computing environments

This chapter is dedicated to the DS component of Pi-CCSF introduced in Section 5.4.1. It presents a prey-inspired target-based decision-making technique (Pi-TBDM) to manage survivability decision processes under unpredictability imposed by UUURs. This includes managing and controlling decisions on when to execute escalating survivability actions, the order of actions and survivability preferences. Hence, the DS component and TBDM are important to escalation and thus addresses H2 of the research hypothesis outlined in Section 1.2.

6.1 Introduction

Nature self-manages to meet set system objectives through intelligent decision-making. The survey of natural preys demonstrates that successful animals possess specific forms of anti-predation systems which enable continuous control and survival despite predation. Moreover, is also identifies complex monitoring as useful attributes in natural environments (Meisel, Pappas & Zhang, 2010). In earlier work that contributes to this thesis, (Mthunzi & Benkhelifa, 2017), reviewed a number of bio-inspired algorithms, with the predator-prey system identified as useful for high adaptation and intelligent survivability decision-making (Mthunzi & Benkhelifa, 2017).

Control theoretic approaches, mathematical models and numerical algorithms have been developed and applied to address decision problem. The main problem with these solutions is that most of them are limited when considering the analysis of unpredictable and complex relationships among different levels of the survivability criteria within a hierarchical cloud system. However, the decision, for instance, to select the best escalation method, both the number and order of actions, for a UUUR event requires a decision-making system that can perform that exact analysis. MDP is widely used for decision-making in dynamic environments (Nagarajan, n.d.). Partially Observable MDP for instance, achieve decision points by extending the observability element of MDP, i.e. that partial information can be inferred to provide probabilistic information about future states (Kreidl & Frazier, 2004b). This requires estimating core states and observing transition probabilities, which is clearly error-prone (Saghafian, 2018).

Realistically, optimal decision making requires monitoring, particularly where all information is not known with certainty. Logically, it is not a good strategy to base decision-making on a system whose observations may produce ambiguous or imprecise information. Fackler and Haight (2014) suggest characterisation approach within the monitoring system, which assigns a variable degree of truth that an observed state is indeed the true state (Fackler & Haight, 2014b), as a robust alternative. To give context, assume a cloud infrastructure runs a service with an expected survivability target/value, which can be achieved by several survivability policies. A survivability parameter ensures that a survivability solution is achieved through executing parameters that achieve an expected value for each service. A monitoring system addresses the traditional challenge of detection via a cyclic process of learning, observation and decision-making. In addition, as it is also necessary to determine how to approximate actions that are implied by survivability data.

Monitoring as a survivability objective encompasses decisions at high-level, to determine how at a larger scale, survivability is approximated. A feedback mechanism obtains sensed or observed information, evaluates its implications against previous observations and actions, which informs decision-making and new response selection. Two mathematical models may perhaps constitute this element. A stochastic model which characterises the multi-state evolution of survivability, and a multi-state observation model which rationalises the statistical correlation between observations. Multi-state observation (Lertpalangsunti *et al.*, 1999) is assumed as suitable since it is nearly impossibility to observe "true states" in UUUR events. Since this thesis does not focus on a specific attack model, risks imposed by UUURs are assumed as multi-path (Singh, Joshi & Singhal, 2013), according to the multiple ways cloud services can be compromised. A calculated survivability evaluation such as risk serves as a trigger for counteraction (e.g. recovery) upon a service (VM) under threat. The risk imposed on a cloud service is evaluated in a network of services, a tenant cluster or simply VMs. The decision system (DS) proposed earlier and detailed in section 5.4.1 is a control approach that is robust for cloud systems under UUUR. Under UUURs, uncertainty is such that an action results in unpredictable outcomes, which result in unpredictable implication.

This chapter considers the survivability decision-making problem of in view of UUUR events through a process of random variable interpretation of escalation actions (alternatives). In addition, it considers decision-making to entail defining decision functions as a probability of each decided action meeting an unpredictable outcome/target. Since survivability in this research is viewed as a fuzzy variable, i.e. within a range of uncertain variables, it is important to note this view induces a possibility distribution over the domain of survivability variables. When considering the presence of UUURs in various cloud computing scenarios and applications domains, survivability decision-making can thus be considered as a multiple-criteria decision making (MCDM) problem. The remainder of this chapter is as follows: Section 6.2 introduces the fuzzy concept and its application to cloud computing survivability decisions. Section 6.3 details the target-based decision-making technique for cloud computing survivability. Section 6.4 presents numerical examples to assess the applicability of the technique proposed above. Section 6.5 concludes the chapter with a summary of findings.

6.2 Fuzzy logic and its application to cloud survivability decision problem

The acceptance that survivability decisions are taken with non-complete detection information and imperfect execution indicates survivability as a fuzzy multi-indicator concept with many contradictions. On one end, unpredictability renders efficient anticipation, decision and execution of enough countermeasures almost impossible. On the other, the cost of services and service assurance becomes complicated. Limited probabilistic information thus requires formal methods to reflect a range of continuous states that cloud systems may resemble. And yet, availability and performance are important parameters for survivability and demand for unhindered provisioning of services is central to the attractiveness of the cloud.

Fuzzy theory (Zadeh, 1978)(Zadeh, 2013) is extensively applied in a range of areas to model complex behaviours due to its ability to represent casual relationships between concepts, and the analysis of a system after convergence. For a formal definition, the interested reader is referred to works by (Gras *et al.*, 2009)(Yesil, Urbas & Demirsoy, 2014), who aptly demonstrate the suitability of fuzzy in analysing complex and dynamical systems where mathematical modelling is complicated. For instance, it is possible to model prey agents' actions based on perceiving or sensing an attack and making an appropriate decision. Nonetheless, applying fuzzy evaluation methods in critical systems survivability is not constrained by the emphasis on character weighting and extreme value action. As suggested by (Cheng, Chen & Chen, 2008)(Sharif & Irani, 2006), because fuzzy weight values often carry useful information (individually) and yet little to do with relationships among objects being assessed, critical information is lost and thus the scientific rationale for weighting needs substantiating.

In this chapter, Feldman's notations style, i.e. capital letters indicate a random variable, while a lowercase to that character indicates a value of that variable (Feldman, 2002) is adopted. For instance, consider Y as a discrete random variable such that $y \in$ Y in which Y is a finite set of M possible values of Y. Introducing the probability of Y at a point in time as $p(Y = y)$ or p(y).

Let fuzzy events A and B be fuzzy sets on S and T whose membership function is described as $\mu_A \; and \; \mu_B \mu_\mathrm{A} \mathrm{and} \mu_\mathrm{B}$, a decision problem with fuzzy events is therefore defined as follows:

$$4 - tuple \; (F, C, \varepsilon, u)$$

, where $\mathrm{F} = \; \mathrm{f}_1, \mathrm{f}_2, \ldots \mathrm{f}_\mathrm{r}$ defines a set of fuzzy states and fuzzy events to a probabilistic space $\mathrm{S} = \; \mathrm{s}_1, \mathrm{s}_2, \ldots \mathrm{s}_\mathrm{n}$. $\mathrm{C} = \; \mathrm{c}_1, \mathrm{c}_2, \ldots \mathrm{c}_\mathrm{p}$ defines a set of fuzzy actions which are fuzzy events on an escalation space $D = d_1, d_2, \dots d_e$. $u(.)$ is a utility function such that $u(.): F \times C \rightarrow [0,1]$.

Assuming fuzzy states F to be orthogonal, (the simplest sense of orthogon to imply complete independence from each other), an expected utility of fuzzy action $C_i$can be defined as follows:

$$(C_i) = \sum_{I} U(C_i, F_j)P(F_j)$$

It is quite clear above that fuzzy action C_0 maximises $U(C_i)$, and thus, defines the optimal fuzzy decision, i.e.

$$U(C_0) \equiv \max_i U(C_i)$$

Suppose a message space X's conditional probability on state S is known, i.e. $p(x_j|s_i)$, the probability that a hypothesised message is true for a state, i.e. $p(x_j|s_i)$, the expected utility of a countermeasure C_i given a message x_j and the conditional probability of F_k, i.e. $p(F_k|x_j)$ is $U(C_{xj}|x_j)=\sum_k (C_I,F_k)p(F_k|x_j)$. Intuitively, an optimal decision C_{xj} is defined as:

$$U(C_{xj}|x_j) \equiv \max_i (C_{xj}|x_j)$$

e is defined as the probabilistic information w.r.t a random variable. Given probabilistic information, e, w.r.t random variable $\tilde{x}$, an expected utility of possessing information is defined as

$$U(C_{\tilde{x}}|x_j) = \sum_{I} U(C_{xj}|\, x_j)f(x_j)$$

Information entropy, i.e. worth of possessing information e is according to the following

$$V(e) = U(c_{\tilde{X}}|\tilde{X}) - U(c_o)$$

Furthermore, a perfect probabilistic information, e_∞ is that information which gives the true state of s_i. Thus, an expected utility for possession of perfect probabilistic information, e_∞ is

$$U(C_{\tilde{s}}|\tilde{S}) = \sum_k U(C_{s_k}|s_k)p(s_k)$$

, where $U(C_{s_k}|s_k) = \max_i U(C_I|s_k)$ and $U(C_I|s_k) = \sum_j u(C_i|F_J)uF_J(s_k)$. the value of information e_∞ can thus also be simplified as $V(e_\infty) = U(C_{\tilde{s}}|\tilde{S}) - U(C_o)$.

Let us consider fuzzy information with fuzzy messages $\{M_1 \ldots, M_p\}$ which describe events on X and satisfy the orthogonal condition. Fuzzy information E is defined as observing messages from $\{M_1 \ldots M_p\}$. Intuitively, the expected utility of countermeasure action C_i given M_k is define by,

$$U(C_i|M_k) = \sum_j u(C_i, F_j)P(F_j|M_k)$$

Clearly, as shown above, it is possible to attain an optimal decision point when both the probability of possessing fuzzy perfect information and the utility are maximal. However, this approach focuses upon optimising decision-making based upon probabilistic information over random variable. This is achieved by first defining the probabilistic value of information and converting this to a perfect probabilistic value based upon an expected utility for possessing that information. Moreover, there is an assumption that fuzzy states meet the orthogonal condition, i.e. states are independent. The current author suggests this decision-making approach be most applicable in traditional survivability decision-making methods where emphasis is upon the clearly defined value of information alone, e.g. business cost or risk, and less so on the state of a system. Nonetheless, as has been shown throughout this research, it is quite often common in practice that changes in the state of a cloud environment are highly dynamic and unpredictable, which impacts the survivability requirements and demands. Thus, a survivability decision-making problem under uncertainty would consider what value to place on unpredictable tenant services and cluster sizes. Suppose cloud services exist in at least four states; $S_1 = V$ and $S_2 = A$, $S_1 = C$ and $S_1 = S$, representing services in the vulnerable state, attacked state, compromised state and survivable state, respectively. It is possible to assess a survivability target for the vulnerable, attacked, compromised and survivable states, T_V, respectively to address this problem.

Let us suppose there are two systems, and each can only assess the probability of the other according to given probabilities. An optimising principle suggests that an agent should decide on an action maximises the probability of a random action meeting the optimal target. Although simple and appealing, with UUURs, uncertainty about the systems themselves, the information, targets, etc. means that the resultant model remains operationally limited. Bordley and LiCalzi, (2000) considering a random outcome whereupon the decision model prescribes the choice of an action that maximises the probability of meeting an uncertain target, as long as the target is stochastically independent of the random outcomes to be evaluated (Bordley & LiCalzi, 2000). Interestingly, Bordley and LiCalzi, (2000) decision procedure leads to a utility-based decision making which, as above, is operationally incomplete under UUUR, i.e. it defeats the survivability mission.

6.3 Target-based decision-making (TBDM)

Formal decision techniques have historically been applied in the computing domain to address the challenge of complex problems. For instance, decision tree techniques are applied in intrusion detection algorithms for probabilistic classification of split datasets. However, applying decision tree techniques in unpredictable environments is limited due to detection performance which is sensitive to mismatch training and test data (Jing, Bi & Deng, 2016). Decision tree techniques-based Markov models have been proposed to alleviate the detection performance shortcomings of standard decision tree techniques. However, the practical implementation of large decision models where significantly large decision points is embedded is a computational challenge. Moreover, decision tree techniques are most suited for traditional object-oriented, component-based computing environments where the survivability concept is fundamentally security-centric.

In Pi-CCSF, the DS will consider various components and attributes simultaneously to make decisions about survivability and their outcomes. This includes resources, cost, the current and possible future survivability state, available actions, system state data from the SSM, SLA state, domain-specific aspects, etc. It may also be the case that these attributes contradict, with each imposing a unique performance on the overall survivability property. Thus, with unpredictability, survivability decision selection is unpredictable, multiple-commensurable and contradicting properties requires an effective decision-making technique.

Several techniques have been developed to deconstruct the relationships between uncertain and undefined systems and the dependencies among the system components or attributes. Basic structural modelling for instance, which deals with modelling concepts and attributes' logic and mathematical constructs (Warfield & Staley, 1996), is widely used across many domains. The DEMATEL technique specifically, is applied in engineering design by (Liu *et al.*, 2014) to facilitate decision-making for engineering designers faced with the dilemma of materials selection. In their work, these authors use the ANP approach as a weighting system to evaluate the influence of dimension and criterion on material structure, and VIKOR method to rank each alternative based on performance difference. Hence, using the method by (Liu *et al.*, 2014), a decision point can be reached by engineering designers, of the materials to improve a product.

While the current chapter does not replicate the application of the technique by (Liu *et al.*, 2014), it will adapt the processes to suit the survivability decision-making problem under consideration. Making survivability decisions under uncertainty could be traditionally managed using probability distribution on the space of predictable states. However, as has been demonstrated throughout this thesis, traditional approaches are not adequate in cloud environments where survivability states are themselves unpredictable, and applying traditional approaches directly is challenging. To mitigate this challenge, the target-based approach is adopted for assessing the survivability target of a cloud system, transform the survivability outcome (payoff) into a probability of meeting the survivability target, and for each survivability action (alternative) $A_{i,}$ and state $S_{j,}$ define the probability of payoff value $c_{ij.}$ The decision processes below are summarised here to place into context the decision technique under consideration. Figure 59 presents the overall graphical illustration of the foregoing.

Step 1: The decision system should construct a relationship map based upon a survivability threat, the survivability mission and the known survivability state of the cloud environment. The map also maintains other important survivability tread-offs such as cost.

Step 2: Based upon the relationship map in step 1, the DS computes a criterion to determine influential weights for a survivability criterion that achieves the survivability target.

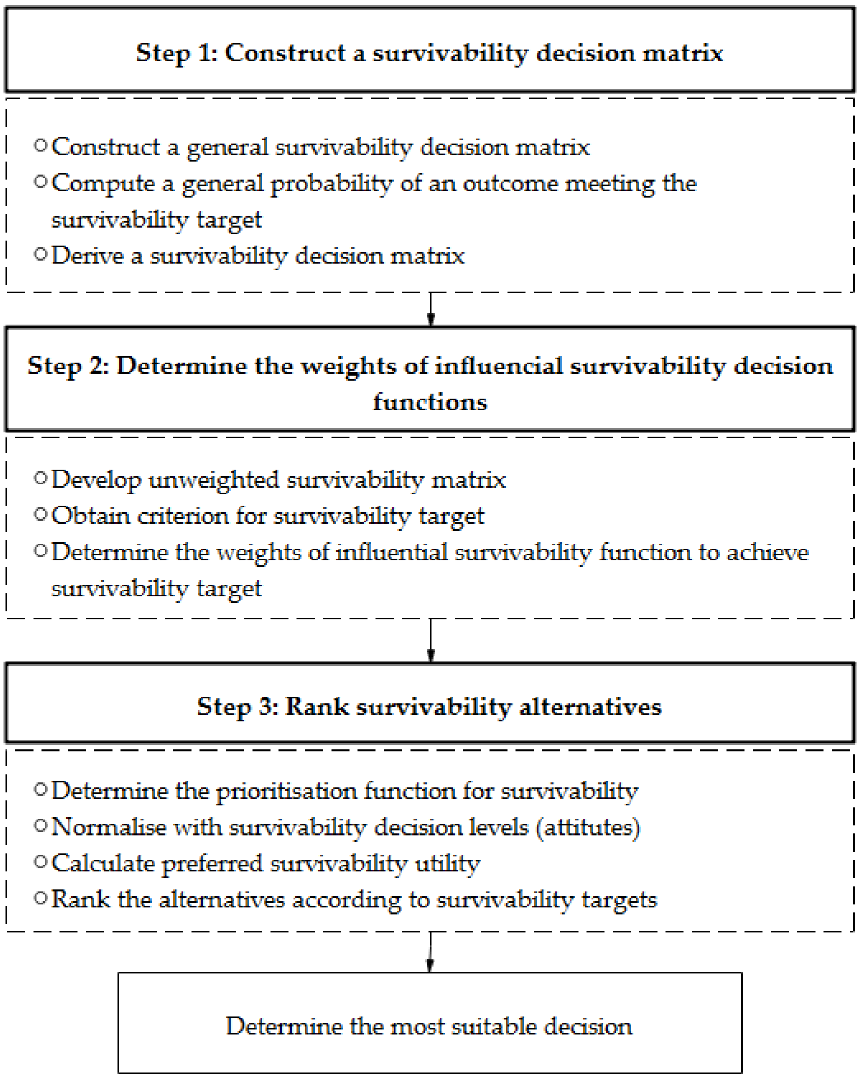

Step 1: Construct a survivability decision matrix
Construct a general survivability decision matrix
Compute a general probability of an outcome meeting the survivability target
Derive a survivability decision matrix
Step 2: Determine the weights of influencial survivability decision functions
Develop unweighted survivability matrix
Obtain criterion for survivability target
Determine the weights of influential survivability function to achieve survivability target
Step 3: Rank survivability alternatives
Determine the prioritisation function for survivability
Normalise with survivability decision levels (attitutes)
Calculate preferred survivability utility
Rank the alternatives according to survivability targets
Determine the most suitable decision

Figure 59. Illustration of Pi-CCSF's DS process

Step 3: Weighting also computes the "attitudes", i.e. the different priorities that capture the preferences for the level of survivability a decision process must consider, as stipulated in an SLA.

Step 4: Based upon the influential weight and preferred target-solution, the DS ranks the alternative survivability decision based upon a prescribed target.

Step 5: The DS determines the best survivability decision. This survivability decision system process in Figure 59.

6.3.1 Construct a survivability decision matrix

To construct a survivability decision matrix, the DS foremost computes the general survivability matrix based upon survivability factors, for instance, as CC or CSP indicates in an SLA. Such factors may point to a scale in a fuzzy set of possible survivability states described in natural language terms. Suppose A_i is a set of survival actions (escalating) which are executed according to a survivability decision-making criterion. Hence, *i* are finite actions (random) such that i = 1, ..., n. S_j represents the state of a cloud system, including intermediary states in which cloud systems may exist. Hence, the state of the system translates into the posture of cloud computing environment under UUURs.

Additionally, it represents such natural language descriptions as attacked, attacked but not compromised, vulnerable, vulnerable but low-medium-high risk, compromised but recoverable, compromised but not recoverable, and so on. Thus, *j* is a set of finite states such that j = 1..., m. From the preceding, it is possible to construct a general matrix with each action on a state producing an outcome, C_n, as shown in Table 12. As will be the case throughout this section, a fuzzy membership function and a possibility distribution should be viewed and will be used interchangeably.

Table 14. A general survivability decision matrix

Action	**State of survivability**						
	S_1	S_2	S_3	S_3	S_3		S_n
A_1	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}		C_{1n}
A_2	C_{21}	C_{22}	C_{23}	C_{24}	C_{25}		C_{2n}
A_3	C_{31}	C_{32}	C_{33}	C_{34}	C_{35}		C_{3n}
A_6	C_{61}	C_{62}	C_{63}	C_{64}	C_{65}		C_{6n}

Formally, (Yager, 2004)'s conversion method is applied to convert a general possibility distribution Table 14 into a probability distribution for the decision problem under consideration. Yager's method is as briefly as follows:

$$P_F(x) = \frac{F_X}{\sum_X F(x)} \qquad (1)$$

Suppose that for each survivability action A_i and state S_j, where T = Tj, if the probability of meeting survivability outcomes, c_{ij} depends upon an unpredictable state of the cloud environment, a general probability of an outcome or payoff meeting the survivability target, A_i and S_j is therefore defined as:

$$p_{ij} = p(c_{ij} \geq T) \qquad 2$$

Table 13 below is therefore derived from the general survivability matrix presented in Table 14.

Table 15. Decision matrix derived from the probability of meeting a target

Action	**State of survivability**						
	S_1	S_2	S_3	S_3	S_3		S_n
A_1	p_{11}	p_{12}	p_{13}	p_{14}	p_{15}		p_{1n}
A_2	p_{21}	p_{22}	p_{23}	p_{24}	p_{25}		p_{2n}
A_3	p_{31}	p_{32}	p_{33}	p_{34}	p_{35}		p_{3n}
A_6	p_{61}	p_{62}	p_{63}	p_{64}	p_{65}		p_{6n}

6.3.2 Determine weights for main survivability decision functions

Due to unpredictability, survivability decision-making outcomes are mostly heterogeneous, i.e. it cannot be known with certainty if executing an action yields a survivability outcome or payoff that is a crisp number, and interval value or fuzzy quantity. Thus, traditional decision-making methods cannot be applied directly. Nonetheless, where meeting survivability outcomes, c_{ij} is a crisp number,

$$p_{ij} = \frac{\int_{-\infty}^{c_{ij}} T(x)dx}{\int_{-\infty}^{+\infty} T(x)dx}$$

If c_{ij} is an interval value or c_{ij} is a random variable with a uniform distribution such that $c_{ij} = [a, b]$.

Otherwise, where c_{ij} is a fuzzy value F, the probability of meeting the target is defined as follows:

$$F_{ij}(x) = \frac{F_{ij}(x)}{\int_{-\infty}^{+\infty} F_{ij}(x)dx} \qquad 3$$

Since c_{ij}, is a random distribution variable with the probability distribution F_{ij} and T is the random variable associated with probability distribution P_T, i.e. the associated probability of target T, as given by $\mathrm{P_T(x)}=\frac{\mathrm{T(x)}}{\int_{-\infty}^{+\infty}\mathrm{T(x)dx}}$, the probability of meeting a survivability target is therefore defined as

$$p_{ij} = \int_{-\infty}^{+\infty} P_T(x)P\big(c_{ij} \geq x\big)dx$$

$$p_{ij} \ = \int_{-\infty}^{+\infty} P_T(x)[\int_{x}^{\infty} PF_{ij}(y)dy]\, dx$$

$$p_{ij} \ = \int_{-\infty}^{+\infty}\int_{x}^{\infty} P_{F_{ij}}(y)P_T(x)dydx \qquad 4$$

If c_{ij} and T are independent, the above method transforms the general survivability decision matrix into the derived matrix shown in Table 15. The advantage is that, unlike with the inhomogeneous decision matrix, the derived probability of survivability outcome p_{ij} uniformly comprises the probability of a survivability outcome cij meeting the survivability target T.

6.3.3 Rank survivability decision alternatives

Kao and Liu (2002) in their work proposed a fractional programming approach for decision-making prioritisation based upon a membership function derived from fuzzy sets (fuzzy weighted average) and the extension principle (Kao & Liu, 2002). Dong and Wong's (1987) method gives discreet and exact solutions in which repeated derivation of a membership value extents the exactness of a membership function (Dong & Wong, 1987). Dong and Wong's work is adapted by Liou and Wang whose method improves the efficiency of the foregoing by reducing the number of permutations (Liou & Wang, 1992). Along these lines, (Guh *et al.*, 1996)' method is, therefore, more efficient as it further reduced the number of permutations to give an efficient sequencing criterion. Using a value function proposed in (Huynh, Ryoke & Ho, 2007), it is possible to rank decisions and execute them according to the prioritisation function:

$$v(A_I) = \sum_{J=1}^{m} P_{ij} P_j \qquad 5$$

For brevity's sake, this value function is adopted by the current author as an expected probability of meeting a survivability target.

6.4 Numeric example

This section considers the following application case scenarios to demonstrate the applicability of the foregoing TBDM technique for cloud computing's survivability decision system (decision-making under UUUR). While the survivability application example presented in section 6.4.1 is arbitrary, the example in section 6.4.2 is adopted from official sources and attributed accordingly.

6.4.1 Semantic web platform

Suppose an arbitrary entity, CloudSea, is a web and semantic annotation platform whose varied client-base has wide-ranging service requirements. Along with providing existing clients with a reliable and secure platform, CloudSea is accessible to other sporadic clients. Hence, core to service provision is its ability to ensure that survivability targets of both traditional and sporadic client are met. Thus, survivability management and control decisions are much reliant upon an uncertain and evolving workload. CloudSea is therefore confronted with the issue of how to manage decisions and accommodate uncertain and undefined survivability and evolving scalability requirements.

Traditionally, the CloudSea infrastructure managed scalability via automated instantiation of services based upon historically analysed statistical data. A range of actions towards this effect (referred to as alternatives) are available and defined as follows:

- *A_1*: a survivability action which increases the CloudSea capacity by 25%,
- *A_2*: in addition to the A1 action, this action increases CloudSea's capacity by 50%.
- *A_3*: a survivability action which when implemented increases CloudSea's capacity by 100%.
- *A_4*: an action which when executed maintains the status quo.

An unknown factor determines CloudSea's survivability capacity. Historical information enables CloudSea to estimate at least three (states) which correspond to service survivability" values": high, medium and low. Associated prior probabilities for each state are known and computed as $P_1 = 0.3$, $P_2 = 0.5$ and $P_3 = 0.2$, respectively. A survivability value defines a previously developed survivability matrix to aid in the decision-making $\widetilde{U}_{ij}$. Survivability is parametrically shown by trapezoidal and triangular fuzzy numbers as shown below.

$$\widetilde{\boldsymbol{U}}_{\boldsymbol{ij}} = \widetilde{\boldsymbol{U}}(\boldsymbol{A}_i, \boldsymbol{S}_j)$$

Table 16. Fuzzy Survivability target matrix

Action	State of survivability		
	S_1: 0.3	S_2: 0.5	S_3: 02
A_1	(80; 90; 99; 105)	(75; 85; 90; 100)	(50; 60; 70)
A_2	(135; 145; 150; 165)	(120; 130; 140)	(-40; -30; -20)
A_3	(170; 190; 210; 230)	(100; 110; 125)	(-90; -80; -70; -60)
A4	70	70	70

Expected fuzzy survivability of each action *A_i (I = 1, 2 ..., 4)* is computed according to (Zadeh, 1978) as below:

$$\tilde{E}(A_i) = \bigwedge_{j=1}^{(3)} \left(p_j \otimes \widetilde{U}_{ij}\right) \qquad 6$$

, Where Λ and ⊗ indicate extended addition and multiplication, respectively. Based upon the survivability target matrix in Table 16, applying equation 6 yields expected survivability values in Table 17 below.

Table 17. Expected fuzzy survivability

Action	Expected fuzzy survivability	Centroid
A_1	(71.5; 81.5; 87; 97)	(84.25)
A_2	(92.5; 102.5; 104; 115.5)	(103.7)

A_3	(83; 96; 104; 119.5)	(100.76)
A4	70	70

Visualising the membership function of expected survivability is presented using Fuzzy Inference System Professional (FisPro); a collaborative framework for fuzzy systems modelling (Guillaume, Charnomordic & Lablée, 2002). By simple inspection of the membership function shown in Figure 60, it is quite clear that A_3 and A_2 are better alternatives in contrast to A_4 and A_1.

Thus, for CloudSea, the most appealing decision is one which increases CloudSea's capacity by 100% (A_3) followed by actions which increase its capacity by 50%. Likewise, decisions to do nothing and increase capacity by 25% are less favourable.

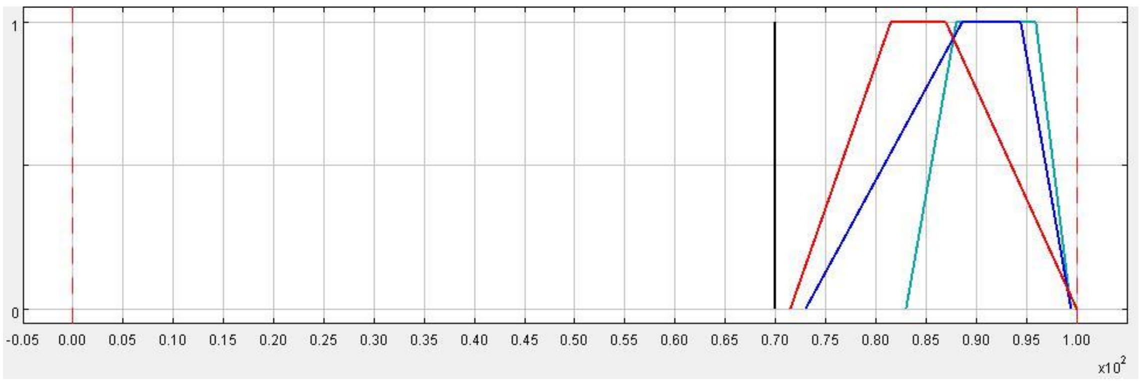


Figure 60. Membership function of expected survivability: A_1 is shown in green, A_2 in red, A_3 in blue and A_4 is black.

Nonetheless, it is not clear which among the above alternatives is better than the other, i.e. A_3 or A_2 and A_4 and A_1. (Chang, Yeh & Chang, 2013)'s opinion is that there is no generally accepted the best method to achieve consistent ranking outcomes. (Zanakis *et al.*, 1998) concurs, and in fact suggests that different methods produce different results even when applied to the same problem. Here, if using the centroid of fuzzy numbers as the ranking criterion, the ranking order is A_2; A_3; A_1; A_4.

6.4.2 Netflix and Hulu

Netflix (Havens, 2019) and Hulu(Hulu, 2016) are two of the major video streaming service providers. While Netflix owns its extensive infrastructure, Hulu does not, yet it provides competitively large-scale video streaming services (Adhikari *et al.*, 2015). Hence, Hulu can support additional platforms, e.g., set-top boxes and mobile devices and offers HD video quality. Let us suppose that due to adaptive streaming, its user-base has varied prioritisation as to what content to access, at what speed, and consistency, hence different parameter choices and behaviour choices over the same or dissimilar service. Essentially, Hulu is faced with the task of managing decision choices considering uncertain priorities and user choices on services. For brevity's sake, Hulu's survivability mission is defined according to actions. A_1, A_2, A_3 and A_4 in the previous example. Similarly, Hulu's survivability capacity is determined by an unknown factor, and historical information enables an estimation of at least three (states) which correspond to service survivability" values". These are critical, neutral and general, whose associated prior probabilities for each state are known and computed as $P_1 = 0.3$, $P_2 = 0.5$ and $P_3 = 0.2$, respectively. A previously developed survivability matrix summarised in Table 16, assists decision-making according to survivability value $\widetilde{U}_{ij}$.

From an infrastructure management point of view, Hulu's survivability requirements and survivability decision strategy are important for varied use and user choices. As observed earlier, the survivability decision-making character parameter key specificities three main decision strategies, which will be considered in the case of Hulu: an optimistic strategy for critical services streaming time-sensitive content, pessimistic strategy for critical services streaming risk or cost-sensitive content and a neutral strategy for critical and non-critical services. Suppose a domain for survivability values is defined as D = [-90, 230], and optimist target is defined based according to the optimistic requirements of the running survivability strategy, where

$$\text{SurvStrat}_{opt}(c) = \frac{c + 90}{230}$$

, Then invoking this strategy yields a derived decision matrix summarised in Table 18. Similarly, invoking the neutral, $\text{SurvStrat}^{\text{neut}}$, and pessimistic strategy, $\text{SurvStrat}^{\text{pess}}$ yields the derived neutral decision matrix and pessimistic decision matrix summarised in Table 19 and Table 20, respectively.

Table 18. Derived optimistic decision matrix

Action	State of survivability

	S_1	**S_2**	**S_3**
A_1	0.3346	0.3079	0.2199
A_2	0.5584	0.4728	0.0353
A_3	0.8229	0.3974	0.0026
A4	0.25	0.25	0.25

Table 19. Derived neutral decision matrix

Action	**State of survivability**		
	S_1	**S_2**	**S_3**
A_1	0.5781	0.5547	0.4688
A_2	0.747	0.6875	0.1875
A_3	0.9063	0.6302	0.0469
A4	0.5	0.5	0.5

Table 20. Derived pessimistic decision matrix

Action	**State of survivability**		
	S_1	**S_2**	**S_3**
A_1	0.8216	0.8014	0.7176
A_2	0.9356	0.9022	0.3397
A_3	0.9896	0.8630	0.0911
A4	0.75	0.75	0.75

As shown in the previous example, applying the value function in Equation 10, Table 21 summarises the value function for each decision-making strategy as well as the ranking order.

Table 21. Ordering based on different targets

Action	**State of survivability**				**Ranking order**
	A_1	A_2	A_3	A_4	
Opt	0.2983	0.411	0.4461	0.25	*A3* _ *A2* _ *A1* _ *A4*
Neu	0.7907	0.7997	0.7466	0.75	*A2* _ *A1* _ *A4* _ *A3*
Pessi	0.5445	0.6053	0.5964	0.50	*A2* _ *A3* _ *A1* _ *A4*

The following observations can be made from the results above. The ranking order of actions associated with the neutral target is the same as that obtained by using the fuzzy expected survivability with the centroid-based ranking criterion, where the risk neutrality is assumed. A neutral target induces a linear utility function which is also equivalent to risk neutrality behaviour, as opposed to optimistic which is equivalent to a risk-seeking behaviour. Thus, in the optimistic strategy, the decision system takes aggressive actions to achieve max survivability despite high risks, whereas in the former, the decision systems aim to achieve survivability and hence risk averse.

The survivability value attained with the different actions depends upon an unknown variable, i.e. unpredictable. Due to the amount of information the decision system estimates three states of nature corresponding to 'high', 'average' and 'low' survivability with associated prior probabilities of *p*1 = 0.3, *p*2 = 0.5, and *p*3 = 0.2, respectively. Moreover, assume that the prior matrix of fuzzy profits ˜*Ui j* is given in Table 8, where fuzzy targets are represented parametrically by triangular and trapezoidal fuzzy numbers. Then, the expected fuzzy target of each alternative *Ai* (*i* = 1*, . . . ,* 4) can be calculated as ˜*E (Ai)* = 3 _*j*=1 *(p j* ⊗ ˜*Ui j)* (22) where ⊕ and ⊗ stand for the extended addition and multiplication, respectively, and risk neutrality is assumed. Using Zadeh's extension principle for Equation 10 then results in the expected fuzzy profits of alternatives.

The issue under consideration is how best to encode the natural concept of prey survival into precise measures about decision-making (adaptation and escalating actions) around multiple evolving states. Conceivably, such efforts may not be perfect and in some cases, may completely misrepresent information if a context changes. There are, however, proven mitigations add complexity to an already complex situation. While dynamic programming conceptualises numerical algorithms with the optimal solution, this approach is computationally prohibitive in practical problems (Kreidl & Frazier, 2004a). Fuzzy theory qualitatively deals with ambiguity, particularly concerning human reasoning (León *et al.*, 2010). The mathematical framework proposed in this chapter attempts to add practical value to cloud computing's survivability challenge. It identifies and defines an appropriate balance between uncertainty and countermeasure actuation.

6.5 Summary

There is merit to the claim that TBDM is a significant basis for decision analysis under unpredictability. It seems an appealing that the TBDM technique is a generalisable approach for formulating decision functions that can be automated to adapt to contextual cues. Such cues include decision attitudes and preferences of varied and perhaps unpredictable decision problems. However, as this element is outside the scope of the current work, it will require further investigation. This chapter presented a fuzzy-based technique to leverage decision-making towards survivability targets. The target-based decision technique addresses the research hypothesis by considering the fuzziness of information when faced with UUURs.

The approach brings fuzzy survivability information, i.e. survivability state which directly impacts the survivability actions (escalation), closer to survivability decision processes. Figure 55 illustrates this by considering survivability preferences and survivability attitudes.

- The investigation suggests that it is possible to institute variable survivability actions (escalation) based upon the known survivability preferences and attitudes.
- Moreover, the target-based approach ensures that the decision model focuses upon achieving survivability outcomes under UUURs. This is a promising method to address the problem of survivability decisions under UUURs.
- The numerical examples show that the decision technique introduces the possibility to generalisable survivability SLAs according to variable requirements and priorities. For instance, survivability decision requirements for a critical system as different for a neutral system. With further work, dynamic automation for instance and evolutionary computing, the decision system, including the actions to recover can be contrived to adapt dynamically and intelligent to suit changing survivability requirements.

Chapter 7 Evaluating Pi-CCSF using Pi-CCSF simulator

This chapter presents the Pi-CCSF simulator, developed and implemented to evaluate Pi-CCSF's theoretical conception of escalating actions proposed in 5.4.2. The aim is to understand the implications of escalation (prey-inspired survivability mechanisms) in the context of practical application in a real-world cloud environment. To achieve this, the experiments in this chapter assess the impact of prey-inspired escalating survivability actions on VMs in various states of compromise. Analysing the implication of escalating actions on the vitality of VMs and the overall survivability will help identify the best conditions for survivability. In addition, hypothesis testing will help to prove if indeed the escalation concept has no effect on survivability.

7.1 Introduction

It is common practice in research to evaluate systems using CTMC, simulation or system prototype. A common influence on the choice of an evaluation approach is the level of development of the system under evaluation. For instance, CTMC evaluation is suitable where a system architecture exists. However, a system prototype brings experimentation closer to real-world systems, albeit with limitations of scalability. Simulation is suitable where a system architecture is developed. Thus, Pi-CCSF simulator is presented in this section as suitable to study the behaviour of the prey-inspired survivability mechanisms and the contributions proposed. While CloudSim provides near real results (Sakellari & Loukas, 2013)(Haas, 2014), its use for the current context is not suitable for several reasons. Foremost is the time limitation, learning the CloudSim framework and adapting it to current considerations is a timeous task. In addition, CloudSim is more suited for modelling Discrete-Time Event-driven models, which limits its suitability for the current experiment. On the contrary, the main application models supported by the Pi-CCSF a multiple-phase analysis of continuous Markov-chain models for the proposed prey-inspired escalation. Thus, using CloudSim in this instance will add useful but unnecessary complexity and render it challenging to scale these models. Restricted validation that allows for a scientific computing researcher to employ separate validation methods (Oberkampf & Roy, 2011) is employed. (Oberkampf & Roy, 2011)'s opinion is that the alternative approach; all-encompassing approach, is confusing when communicating results.

The approach in this research hence comprises one key component: A python implementation based on the Lotka-Volterra predator-prey model (Wangersky, 2003)(Parker & Kamenev, 2009) validates Pi-CCSF's theoretical escalation concept. LV's model is the base model upon which the current hypothesis of prey-inspired survivability property is built. Many pieces of research have shown that LV's properties and behaviour are valid indicators for survivability, and hence, the system model in this simulation are deemed valid for the Pi-CCSF simulation.

To evaluate Pi-CCSF, it is assumed that cloud environments by default are vulnerable, and if the vulnerability is unresolved, an attack is launched (attacked state) leading to compromise (compromised state). Pi-CCSF's objective is, therefore, the application of prey-like actions to alter the transition processes of cloud system states from various levels of compromise to the survivable state. These transient states have a steady-state probability, which implies that the probability of an action being aborted, the probability that action proceeds another and the probability of escalating actions, etc. do not change with time. Related approaches include (Nguyen, Kim & Park, 2016) steady-state analysis of cloud service availability, (Zheng, Okamura & Dohi, 2015)'s single-phase survivability analysis of VM-based architectures and (Changa *et al.*, 2016)'s multiple-phase survivability analysis of continuous Markov-chain model to capture a datacentre recovery process.

The remainder of this chapter is as follows: Section 7.2 presents the Pi-CCSF simulator; details the Pi-CCSF simulation environment and simulation code description. Section 7.3 outlines the simulation design. Section 7.4 presents the simulation results in the analysis. Section 7.5 concludes the chapter with a summary of key findings.

7.2 Pi-CCSF simulator

The simulator and experiments presented in the chapter are designed around Pi-CCSF. Specifically, the experimental approach aims to gather the details of VMs survivability dynamics under different scenarios. A python simulator hence is developed to investigate VMs' behaviour when prey-inspired survivability mechanisms are employed. The expectation is that simulation results give an indication of the extent to which prey-inspired actions alter the state of the cloud system, and precisely demonstrates the consequences of escalating actions to cloud systems or services in vulnerable, attacked or compromised states.

7.2.1 Environment

For a visual illustration of the application of the conceptual framework, Figure 61 is presented as an example of the cloud service application view of Pi-CCSF. Survivability areas 1 and 2 emulate the traditional survivability configuration; one is operational whereas the other acts as backup or standby, with redundant resources. From left, a gateway enables CSP and CC to access Pi-CCSF via a web front-end, e.g. an API. Based upon an existing SLA and the cloud infrastructure, different scheduling policies will have different effects in terms of task execution. Pi-CCSF's management layer (SSM) is publicly available, enabling both CPS and CC to interact directly and influence the survivability management strategy. However, low-level prey-inspired survivability mechanisms are private and only accessible to the CPS, to secure survivability mechanisms against malicious attack, manipulation or misapplication. Survivability data repository stores running configuration files and facilitates efficient configurations reuse at real-time. Suppose an incident occurs in Area 1, the system configuration reverts to the traditional setup, whereupon the CSP and CC operations are provided via the backup survivability Area 2

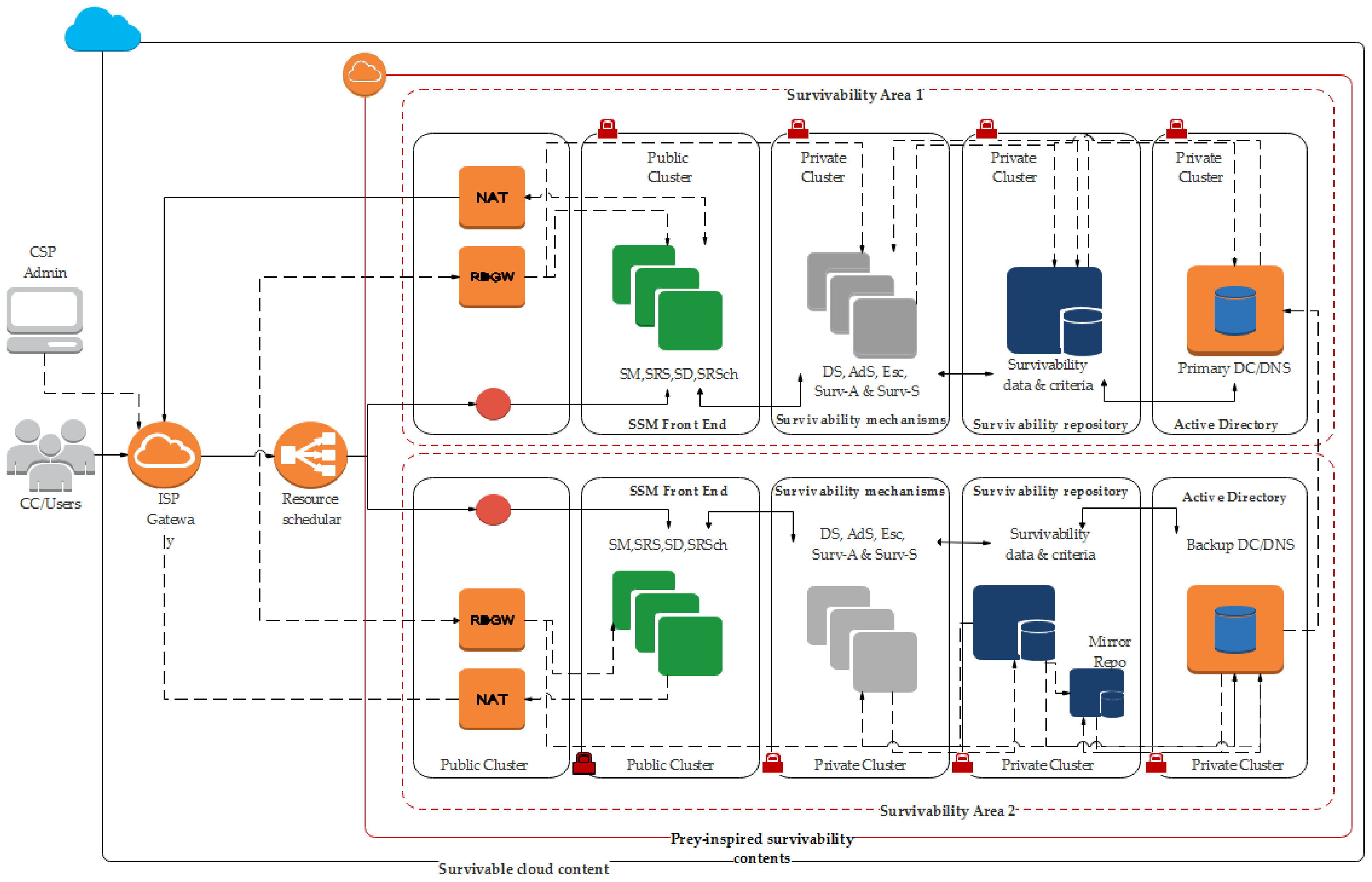
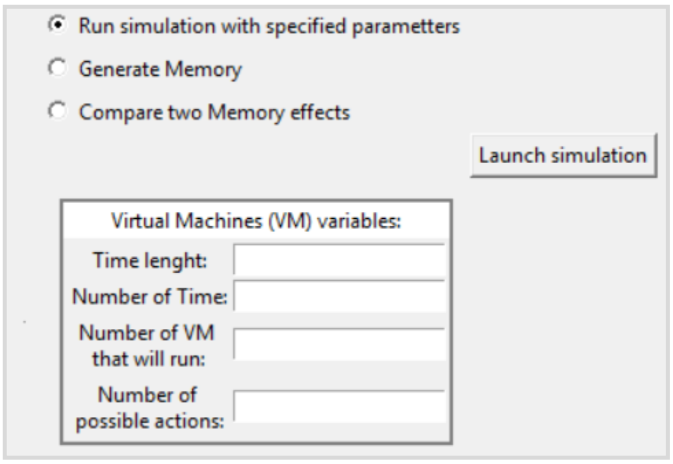


Figure 61. A cloud service application of Pi-CCSF around running survivability service area (Area 1) and a backup area (Area 2)

CCSF simulator is tested on Microsoft Windows 10 workstation and implemented in Python providing an intuitive GUI (Figure 62 and Figure 63). Since Pi-CCSF simulator uses Python code libraries, it can be integrated into real or near cloud environments with relative ease (Hwang & Li, 2010).

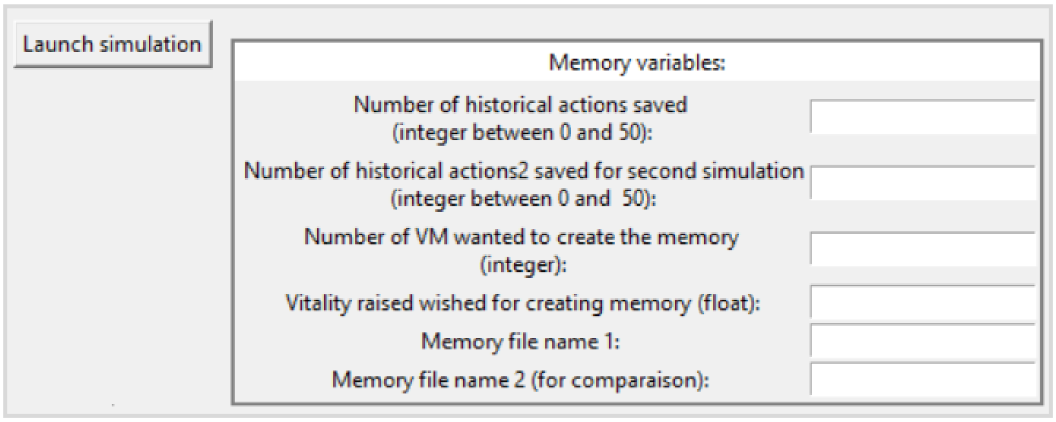


[✓ Run simulation with specified paramettters]
Generate Memory
Compare two Memory effects

Virtual Machines (VM) variables:

Time lenght:	
Number of Time:	
Number of VM that will run:	
Number of possible actions:	

Figure 62. Part of the graphical user interface (GUI) to set virtual machine variables. A user can launch a simulation based upon specified variables, memory generated variables or using the best variables in memory



Memory variables:

Number of historical actions saved (integer between 0 and 50):	
Number of historical actions2 saved for second simulation (integer between 0 and 50):	
Number of VM wanted to create the memory (integer):	
Vitality raised wished for creating memory (float):	
Memory file name 1:	
Memory file name 2 (for comparaison):	

Figure 63. Part of the graphical user interface (GUI) to set memory variables. A user can launch a simulation by specifying the amount of actions that can be stored in memory, the numbe of VMs and a threshold VM survivability vitality.

As shown in the following section, the Python scripting system forms the underlying simulation platform, providing a conduit between the GUI and the underlying programmatic prey-inspired survivability components.

Two simple programmatic functions are performed, setting up virtual machine and memory variables. These enable a researcher to simulate with specified parameters, run the simulation with generated memory and run a simulation to compares the implications of generated memory. Also, the researcher can automate the launch and termination of a simulation run. Pi-CCSF simulator is designed to provide the features to simulate prey-inspired survivability, specifically unique features that facilitate the development of escalation models. The quantitative analysis aims to evaluate VM vitality and survival rate (as indicated by population changes) in the presence of an unpredictable (random) attack (attacking VM). Pi-CCSF simulator also includes customisable models of memory generation, VM population dynamics, model parameter integration and an intuitive GUI.

The use of python to create Pi-CCSF simulation models implies that various component groups are defined with specific dependencies defined among them. Whereas simulation parameters are determined in advance, incremental learning algorithms (Xiao *et al.*, 2014; Yoon *et al.*, 2018) would be most suitable with further development to the simulator due to its adaptiveness and dynamicity in large-scale systems. The complete code is provided in Appendix B. It should be noted that in its current form, Pi-CCSF simulator does not fully integrate all Pi-CCSF's components, as integration can be quite complex. As such, the decision system (DS) is addressed separately in Chapter 6. While decision triggers and decision targets and attitudes, are assumed in the experiments, none of these requirements is met in simulation. Decisions associated with escalating actions are hardcoded in the python script to allow for analysis.

To create the default parameters, the following code example defines the timestamp, number of VMs, the number of actions, including historic (or learned) actions, preferred number of VMs to achieve survivability and the preferred vitality for survivability, etc. based upon properties from the prey inspiration. In addition to creating and modifying the default models parameters, Pi-CCSF simulator defines a range of common tasks. For instance, to define a "stimuli" or input which invokes the simulation of VMs with respect to the input and the current state. Hence, controller.py allows the researcher to evaluate the survivability models by running multiple simulations, changing parameters and storing and outputting simulation data. The sample code (See Appendix C) generates memory parameters and searches and stores historic actions. In addition, the code allows the researcher to explore the behaviours of VMs through the simulations cycle; define vulnerability parameters, the vitality of vulnerable VMs, the average survivability of VMs in general and record the simulation output to a local directory. In the code simulation (sample provided in Appendix C), the simulation performs a search task, to identify the number of vulnerable VMs and execute survivability actions if the number is below a threshold value. If current actions improve VM vitality, the search and execute task (escalation) is not necessary, i.e. raiseSpeed = False, and stores the actions for future use.

Given the novelty of the escalation concept for cloud computing survivability as conceptualised for Pi-CCSF, it would be necessary to develop the escalation model further. In its current form, this model simplicity facilitates easy definition and execution of the escalation process throughout Pi-CCSF's system components. In a real cloud environment, escalation is triggered by several of Pi-CCSF components including the SM, SSM and other related adaptation and dynamic support sub-tasks. However, in its current form, escalation model behaviours are deliberately designed to be triggered by simple rules, i.e. if several survivable VMs or VM vitality falls beneath a certain value, then execute escalation from a set of actions. Theoretically, the escalation process implies that an action vector maps actions to probabilities of an occurrence (compromise, exploiting a vulnerability and surviving). Thus, it defines the escalation sequences imposed by vulnerable, attacked and compromised VMs. Due to its current simplicity, the escalation model can be translated into mathematical models and further code and so on, as the simulator is developed. To mimic prey animals' efficient learning and future threat prediction and response (Systems, 2006), escalation outcomes are retained in memory. Actions are randomly computed to search for a survival solution and retain its value when the vitality of vulnerable VM decreases (it is assumed to indicate negative survivability), otherwise no solution is saved if the vulnerable VMs survive.

7.3 Simulation design

While knowledge of an adversary informs the type of countermeasure implemented, (Albanese, Jajodia & Venkatesan, 2018)'s opinion is that possessing such knowledge with certainty when dealing with UUURs. Pi-CCSF simulator does not consider any attack or adversary model but instead, the properties of adversaries are identified simply around the outcomes of UUURs (Ma & Krings, 2011)(Ma *et al.*, 2014). Hence, a worst-case scenario is assuming that VMs are by default "vulnerable", implies that they are to some degree under the control of an adversary. Hence, discovering the behaviour of the simulation requires an analysis of the dynamics of VMs' probable state transitions concerning prey-inspired escalation upon the vulnerable state. Figure 64 illustrates the Pi-CCSF simulator component and the interactions between them. SurvivabilityModel.py simulates the vitality of vulnerable VMs; equation 7 and rate of survival for attacked VMs equation; 8. Furthermore, the simulation aims to discover the distribution of survivable VMs equation; 9 over time, if executing survivability actions recovers VMs from the compromised state; equation 10. For the purposes of this simulation, this is assumed to be closed system, i.e. V_VM(t) + A_VM(t) + C_VM(t) + S_VM(t) ≡ 1. Known attacks (with established solutions) pose a threat with known and established solutions. The computing of VM dynamics is according to the probabilities summarised in Table 22.

Table 22. Summary of default parameters

Description	**Annotation**
Probability of exploiting VVM	β
Probability of AVM resisting attack	γ
Probability of compromising SVM	$\propto$
Survivability action	a
Rate of compromising SVM	1/a
Escalating action, a = {a1, a2, ... an}	Esc
Probability of CVM recovering to SVM	η, increase with escalation
Rate of juvenile VM	$\alpha = V_{VM}(t) + A_{VM}(t)$
Rate of parent VM	$1-\alpha = S_{VM}(t) + C_{VM}(t)$

Specifically, survivable and vulnerable VMs are presented analogous to parent and juvenile preys, respectively. "Juveniles" represents a system of two VM systems; Vulnerable (V_VM) and Attacked (A_VM), distinguished by their inability to resist attack., "Parents" represents a system of two VM systems; (Compromised (C_VM) and Survivable (S_VM), distinguished by their capacity to recover from compromise.

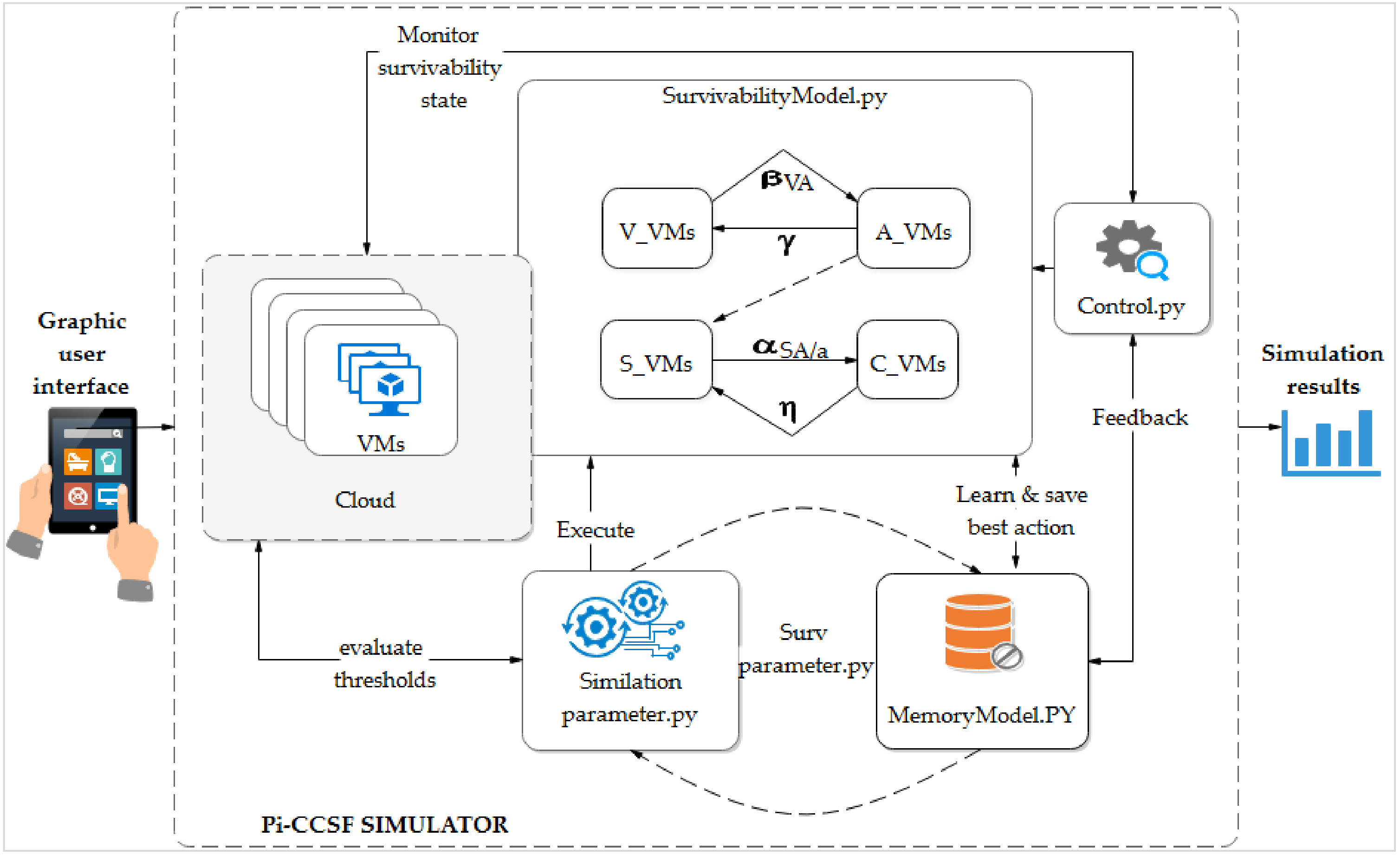


Figure 64. Pi-CCSF simulator components

To observe the distribution of vulnerable VMs over time concerning the relationship between the probability of a vulnerability being exploited (successfully or otherwise) and the probability of resting an attacker at a point in time during the simulation, the evaluation is according to the following:

$$\frac{\Delta V(t)}{\Delta t} = -\beta V_{VMS}(t) * A_{VMs(t)} + \gamma A_{VMs}(t) \qquad 7$$

, where VVMS (t) and AVMS (t) are the total number of vulnerable and attacked VMs at time t, ΔV(t) specifies the rate of a range of change of vulnerable VMs considering the given probabilities specific to each VM state.

To observe the distribution of attacked VMs over time, based upon the relationship between the probabilities of a vulnerability being exploited, the probability of failing to resist an attack and the total number of attacked VMs at a time, the evaluation is according to the following:

$$\frac{\Delta A(t)}{\Delta t} = \beta V_{VMs} * A_{VMs}(t) - \gamma A_{VMs}(t) \qquad 8$$

, Where ΔA(t) represents the rate of attacked VMs at a given time t, AVMs(t) and VVMs(t) specify the total number of attacked and vulnerable VMs, respectively, at any given time t. Intuitively, - γAVMs (t) is a negative constraint on the number of attacked VMs.

To observe the distribution of survivable VMs over time-based upon the relationship between the total number of survivable VMs and attacked VMs, with respect to the strength of an action performed, where the probability of compromise is a constraint on the total number of compromised VM and the rate of survival, the evaluation is according to the following:

$$\frac{\Delta S(t)}{\Delta t} = -\frac{\alpha S_{VMS}(t) * A_{VMs}(t)}{a} + \eta C_{VMs}(t) \qquad 9$$

, Where ΔS(t) is the rate of survival, i.e. survivability at a time t, the action(s) taken, impacts upon the survival rate, intuitively, inadequate action(s) harm survivability.

The evaluation to observe the distribution of compromised VMs over time-based upon the relationship between the total number of survivable VMs and attacked VMs at a time with respect to action(s) taken is according to the following:

$$\frac{\Delta C(t)}{\Delta t} = \frac{\alpha S_{VMs}(t) * A_{VMs}(t)}{a} - \eta C_{VMs}(t) \qquad 10$$

, Whereas ΔC(t) is the rate of compromised VMs at time t of the simulation, SVMs (t) and AVMs (t) specify the number of survivable and attacked VMs at that time t, of the simulation. Intuitively, $-\eta C_{VMs}(t)$ in the equation points to a negative constraint on the number of compromised VMs. Simulation parameters were chosen to reflect best the prey-parentage upon which Pi-CCSF is developed. A pilot experiment presented in section 4.5 which simulated prey survival dynamics against predators suggests the current experiments as plausible. For the experiment below, the following benchmarks are used: a known vulnerability probability, β, the probability of resisting an attack, probability of compromise and the probability of recovering, i.e. surviving; denoted by λ, ∝ and η, respectively.

7.3.1 Simulating base model with escalation

Figure 65 shows the Pi-CCSF's simulation dynamic. The simulation is initialised with default parameters after which the escalation model is deployed.

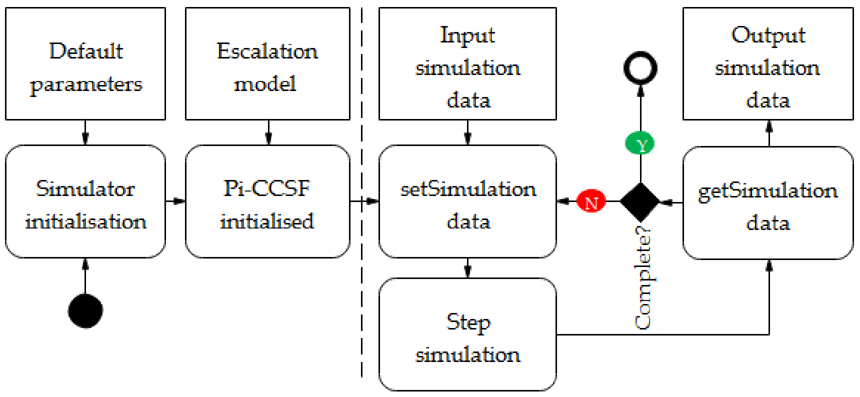


Figure 65. Graphical representation of the simulation dynamic

An escalation model provides escalation parameters such that at run-time, "set simulation data" and "get simulation data" methods are called until the end of the simulation.

Suppose CS is a set of requirements that a cloud application must achieve to deliver a cloud service. At instantiation, every aspect of CS, i.e. CS_i^{th} is met (provided all other service levels are met). Suppose Fcs_i is an evaluation function of CS, such that $Fcs_i(t)$: R → [0,1], where R is a set of all positive values. Fcs_i hence stipulates the degree to which the ith requirement from Fcs is satisfied at any given time, (t).

Suppose in functional analysis of a complete system, a subsuming function $gCS(t)$: R → [1,0], defines the degree to which the whole system requirements are met at any given time. Without UUUR events, assume gSC = 1 to imply service delivery at expected levels, whereas gSC = gCS ≠ 1 or gSC ≤ 1 implies the negative effects of UUUR. The deviation from gCS = 1, upward or downward is indicative of the UUUR and gCS = 0, being catastrophic. A restorative movement towards gCS = 1 is indicative of actions that are applied to reach acceptable service requirements, i.e. escalating actions.

Let us consider a set of systems states, i.e. survivable, vulnerable, attacked, compromised (and catastrophic) such that $0 \leq \text{gCS} \leq 1$. Let the maximum and minimum state of gCS, i.e. vulnerable but not attacked, attacked but not compromised, compromised but not catastrophic, define a cloud service states before complete catastrophe such as when services are completely unavailable. Suppose a dead threshold (DT) is an unacceptable service state, i.e. where gCS < DT. Nonetheless, in rare circumstances, DT can be induced from a service security administration perspective, e.g. where a VM is killed. Intuitively, a dead set is a set of undesirable dead states, i.e. 0 ≤ gCS ≤ DT, and the probability of exiting a dead set is insignificantly. Figure 66 illustrates this context of the dead set explained above. Hence, a system's survivability (SS) is generally defined as a function of two variables such that $SS(\overline{\text{gCS}}, \overline{t})\colon [0,1] \times\!\to [0,1$, where $\overline{\text{gCS}}$ is the mean of gCS and $\overline{t}$ is the meantime to reach the dead state from the vulnerable state. Hence, escalation is both conditional and jointly dependent upon other probabilities. In this example, escalation is considered in space modelling a series of prey-inspired survivability actions. An escalation space is defined as follows.

3-tuple (C, μ, δ), where C represents all available survivability actions such that $C = \{c_{1,} c_2 \ldots, c_n\}$.

$\mu(.): C \times cs \rightarrow [0,1]$ is a function to evaluate the effectiveness of each survivability action concerning a cloud service.

$\delta(.): C \rightarrow Z^*$ is a function to evaluate the cost of a survivability action, and $Z^* = \{1,2, \dots n\}$, i.e. a set of positive integers.

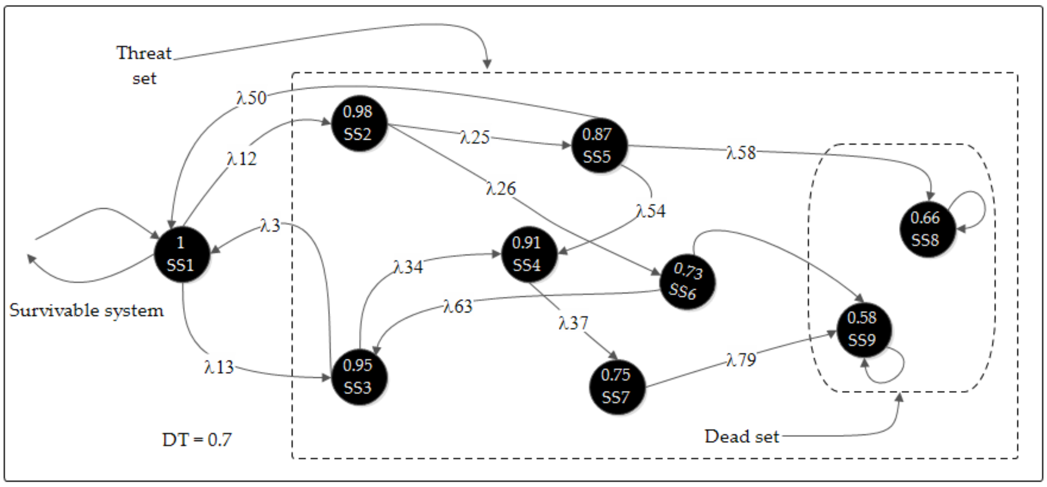


Figure 66. Illustration of variable systems states including the dead set

For simplicity, it is assumed that action selection varies depending upon the state of a cloud environment and across variable cloud services. Thus, an effectiveness evaluation is a function for both the survivability action and the cloud service. Hypothetically, it implies that survivability action X may be effective against a certain vulnerability in one cloud service, and yet insufficient to recover a compromised service. It is quite clear in this case that the dependency between escalation and service state decisions must be optimised to achieve survivability. Although optimisation and decision making are both outside the scope of this chapter, an optimal solution is reached when both escalation and service states are selected to maximise effectiveness against compromise and minimise risk.

7.3.2 Experiments

The experiments performed are executed over several runs and the results are summarised into four experiments according to the parameters in Table 23. For each experiment (Simulation Id), a number of VMs are considered with a predefined survivability threshold value to stipulate a point to invoke escalation. Theoretically, this emphasises a value upon which a survivability SLA is agreed as acceptable to service provision. For each experimental run, some survivability actions are retained to emulate memory. In a practical application, this process improves decision-making for real-time escalation (RTE). Since vulnerable VMs (V_VM) and attacking VMs (A_VM) both act as a set of 5 escalating actions, escalating actions lead to a desirable outcome, i.e. enhanced survivability, dependent upon the scenario in question. Survivability is evaluated in range 1 and 0; illustrated as the vitality of a VM.

Theoretically, 1 implies a high degree of survivability, while 0 vitality implies the opposite. If the vitality reaches 0, the VM is considered as dead. In real terms, a dead VM characterises an undesirable outcome, i.e. compromised, such as an event whose effect means that cloud services are not available to the CC. It is necessary to suppress NAN and infinity values, i.e. computations whose values cannot be expressed as a number to observe VMs dynamics over time. Hence, survivability exists within the solution space (-1, 0, 1). For each vulnerable VM, an attacking VM will exist in the simple model (survivability model.py in Appendix B). Due to computational constraints, each VM perform a limited number of different actions and the results are liberalised using an average of 100 groups of simulations. Each group of simulation will simulate vulnerable virtual machines (V_VMs) and attack or attacked (A_VMs). The simulation starts from random survivability value whereas the simulation time number of VM components is variable. To the best of this researcher's knowledge, exact resolution of survivability analysis with escalation as presented here does not exist in current literature. Thus, in principle, the analysis of results presented here is limited due to the absence of a comparative reference to evaluate the effectiveness of escalation model presented in this work.

The Pi-CCSF simulator implements a "shared memory" folder to simulate simple communication between vulnerable VMs (V_VM). In real cloud environments, inter-VM communication mechanisms facilitate a range cloud computing process, e.g. VM live migration (Ren *et al.*, 2012)(Author & Author, 2013)(Zhang *et al.*, 2013). Hence, for this investigation, a shared directory is assumed synonymous to XenSocket which implements communication between VMs based on shared memory (Jamal *et al.*, 2009) (Gebhardt & Tomlinson, 2010)(Mthunzi *et al.*, 2018). Similarly, Xway and IVC use the shared memory concept based upon the Xen hypervisor architecture. VM behaviours are compared based upon a constant initial vitality and the best-known actions; a random action performed without memory and a random action with memory. Random actions with memory imply that the best action is learnt and saved in memory if the best action exists and the "gain" is above 0, otherwise, a random action will be used. This best action behaviour is used as a baseline to provide a realistic comparison since both mechanisms are not optimised. Unless stated otherwise, it is assumed that all simulation intervals are exponentially distributed, and survivability actions are performed randomly according to a generic action matrix presented in section 5.2.3 .

Table 23. Input parameters for five simulation runs

Simulation id		Nbr of VMs	Threshold in %	Number of actions saved
Sim 1	1_1	1000	0	0
Sim 1	1_2	1000	0	0
Sim 2	1_2	50	50	1
Sim 2	1_3	50	75	1
Sim 2	1_4	50	90	1
Sim 2	2_2	50	50	3
Sim 2	2_3	50	75	3
Sim 2	2_4	50	90	3
Sim 2	3_2	50	50	5
Sim 2	3_3	50	75	5
Sim 2	3_4	50	90	5
Sim 3	1_2	200	50	1
Sim 3	1_3	200	75	1
Sim 3	1_4	200	90	1
Sim 3	2_2	200	50	3
Sim 3	2_3	200	75	3
Sim 3	2_4	200	90	3
Sim 3	3_2	200	50	5
Sim 3	3_3	200	75	5
Sim 3	3_4	200	90	5
Sim 4	1_2	500	50	1
Sim 4	1_3	500	75	1
Sim 4	1_4	500	90	1
Sim 4	2_2	500	50	3
Sim 4	2_3	500	75	3
Sim 4	2_4	500	90	3
Sim 4	3_2	500	50	5
Sim 4	3_3	500	75	5
Sim 4	3_4	500	90	5
Sim 5	1_2	1000	50	1
Sim 5	1_3	1000	75	1
Sim 5	1_4	1000	90	1
Sim 5	2_2	1000	50	3
Sim 5	2_3	1000	75	3
Sim 5	2_4	1000	90	3
Sim 5	3_2	1000	50	5
Sim 5	3_3	1000	75	5
Sim 5	3_4	1000	90	5

7.4 Results and analysis

Several experiments were performed, with varied escalating actions configurations tested around survivability. The goal is to study how survivability evolves with respect to the number of actions, sharing memory and the memory parameters such as the threshold value and the number of saved actions. As a control will be required, some simulation results will also be realised to allow a comparison. Each simulation performs ten runs to simulate a network composed of ten different groups. Each group has its shared memory, and each simulation supposes that five different actions are available. The simulations will run 50 clock times (ticks) including the initialisation as the results stabilise after each step. 50 ticks are chosen due to the computational limitations. These summaries correspond to the rounded results at 10-3 obtained for each group. Appendix C presents the simulation results for the best actions.

7.4.1 The vitality of random and best actions

To enable synthesis and to view in graphs, minimum and maximum vitality values are observed to cater to the huge variance in results. For each graph, the x-axis shows the simulation time in ticks while the y-axis shows the vitality value. The results in this section plot VMs' responses when random actions (RD) and the best action (BA).

Table 24 presents a sample summary for vitality evolution (Min-Max) of vulnerable VMs, whereas Figure 67 tracks the response of vulnerable VMs and shows that random action marginally outperforms the best action, and vitality is relatively unchanged.

Table 24. Sample data of vitality evolution for vulnerable VMs over 10 ticks

Min-Vitality V_VMs	0.329	0.357	0.348	0.340	0.333	0.328	0.325	0.323	0.322	0.321
Max Vitality V_VMs	0.346	0.377	0.369	0.360	0.357	0.355	0.353	0.351	0.350	0.349

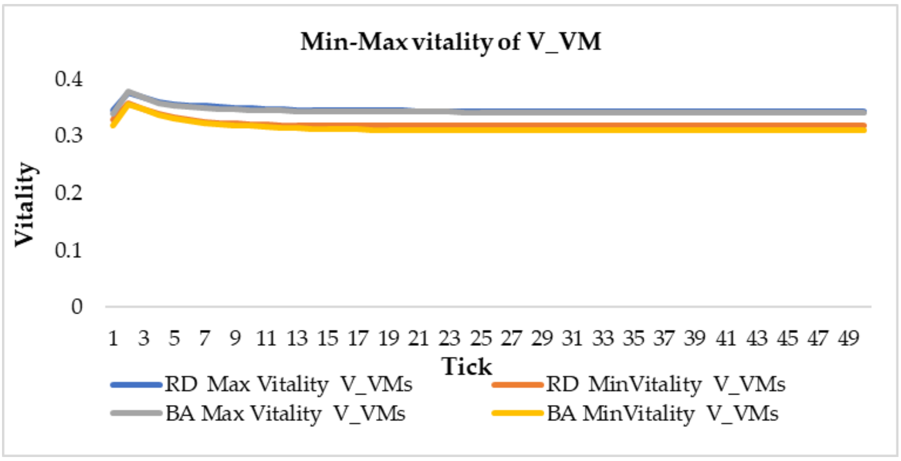


Figure 67. The vitality of vulnerable VMs.

Table 25 presents a sample summary for vitality evolution (Min-Max) of survival VMs and Figure 68 plots the behaviour of survivable VMs and shows a relative decline of vitality over the simulation period.

Table 25. Sample data of vitality evolution for survival VMs over 10 ticks

Max- Vitality S_VMs	0.345	0.336	0.326	0.321	0.310	0.298	0.287	0.278	0.270	0.264
Min-Vitality S_VMs	0.316	0.276	0.264	0.255	0.244	0.233	0.226	0.220	0.216	0.212

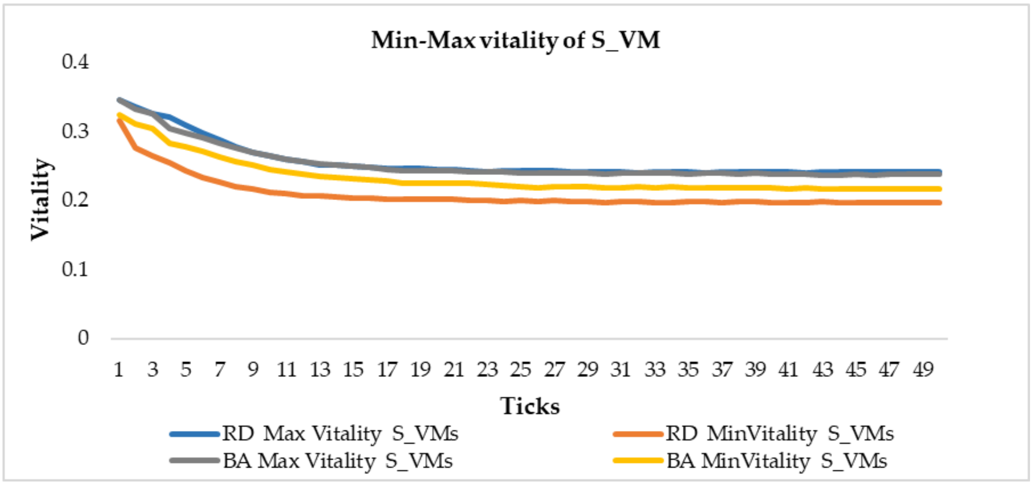


Figure 68. The vitality of survivable VMs.

RD and BA both generally have a marginally different impact, despite a deficit when considering the Min RD.

Table 26 presents a sample summary for vitality evolution (Min-Max) of attack VMs whereas Figure 63 plots the behaviours of attack VMs' response to both random and best actions. Foremost, both RD and BA have marginal differences although RD seems in parts to outperform BA. Moreover, A_VMs achieve the highest vitality (about 0.5) than any VMs over the simulation period.

Table 26. Sample data for vitality evolution for attack VMs over 10 ticks

Min-Vitality A_VMs	0.316	0.310	0.346	0.377	0.400	0.417	0.427	0.433	0.437	0.441
Max-Vitality A_VMs	0.345	0.343	0.389	0.423	0.442	0.455	0.465	0.472	0.476	0.479

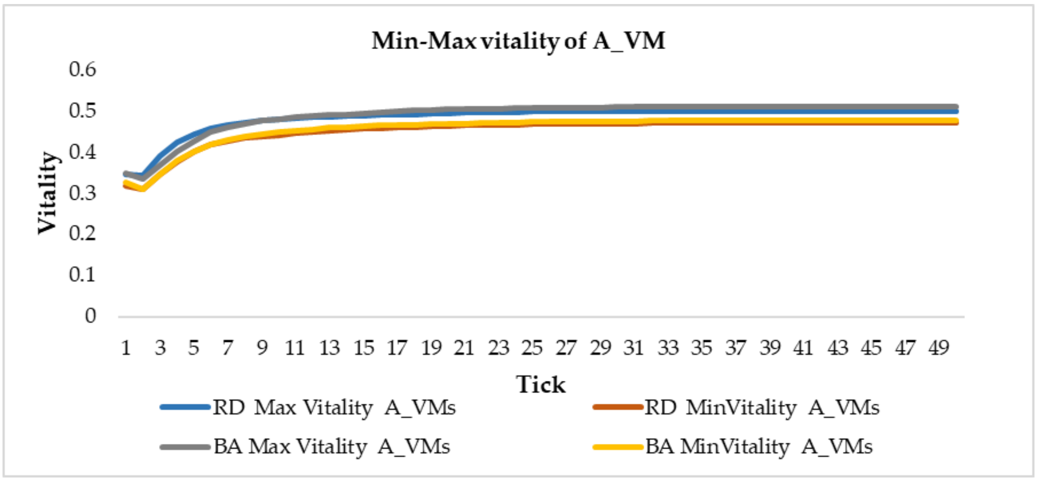


Figure 69. The plot of vitality of Attack VMs

Table 27 presents a sample summary for vitality evolution (Min-Max) of corrupt VMs and Figure 70 plots the response of corrupted (interchangeably referred to as compromised) VMs to RD and BA. This graphic shows a steady increase in vitality considering both actions. Moreover, the random action outperforms the best action.

Table 27. Sample vitality evolution for corrupted VMs over ten ticks

Min-Vitality C_VMs	0.000	0.010	0.017	0.024	0.033	0.045	0.067	0.091	0.116	0.135
Max Vitality C_VMs	0.000	0.049	0.061	0.075	0.098	0.129	0.159	0.183	0.201	0.214

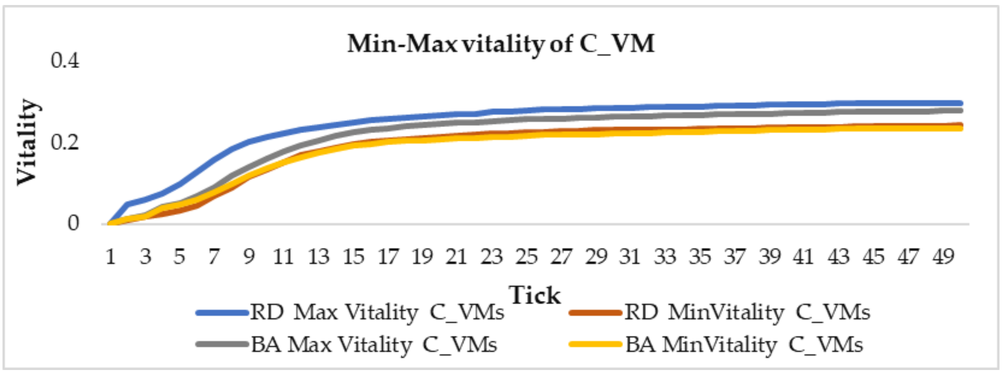


Figure 70. Sample data for vitality of Corrupted VMs

7.4.2 The vitality of actions and threshold values

This section presents the result of running 5 actions based upon similar simulation parameters in terms of probabilities of compromise, attack, survival, etc. The first escalation model is are random actions which are not preserved for future use. This contrasts with best actions which arise from actions evaluated and stored in memory based upon their outcomes. The tested actions have four distinct regions to plot survivability trajectory in a graph. All actions have the same initial random action (RD) vitality value. Each has a minimum and a maximum vitality value and a transition between one action and its threshold value and a consecutive action with its threshold value.

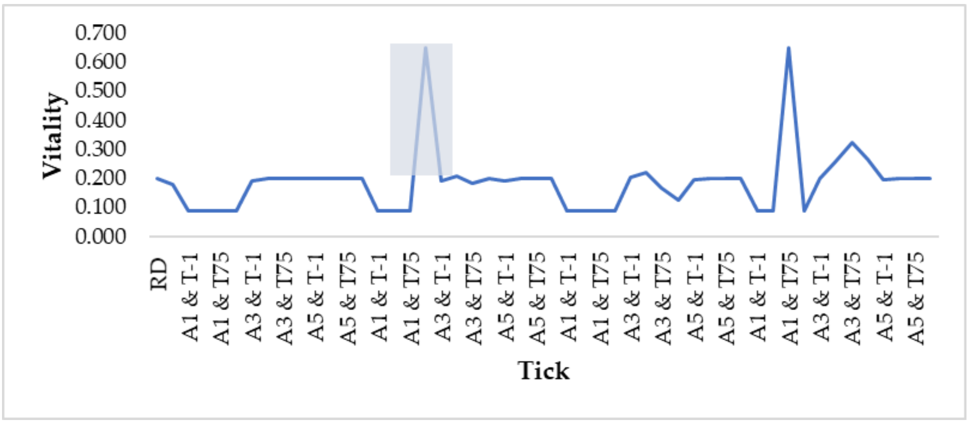


Figure 71. A plot showing the evolution of the vitality for action one (1).

Figure 71, Figure 72, Figure 73, Figure 74 and Figure 75 depict the vitality trajectories when actions A = {1, 3 and 5} are evaluated against a survivability threshold, T = 90. The x-axis in each depicts the action type; random (RD), the action (A*) and survivability threshold value (T*). The y-axis represents vitality, which is a value synonymous to the survivability in this section. In Figure 71, A1 and T75 both produce the highest vitality.

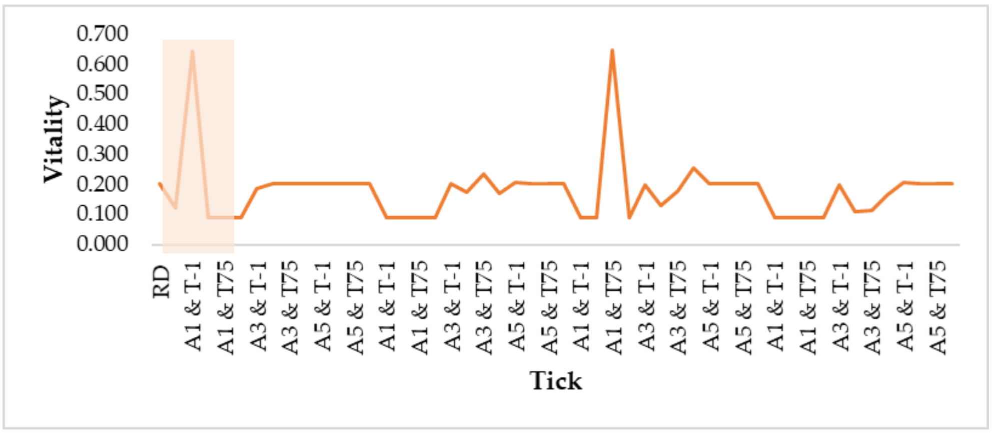


Figure 72. A plot showing the evolution of the vitality for action two (2).

In Figure 72 a prior A1 and T1 achieve a vitality above 60%. In Figure 73, A1 and T1, A1 and T75, and A5 and T1 achieve over 60% vitality. In Figure 74, over 60% vitality is achieved at A1 and T1 at two intervals during simulation whereas the same vitality is achieved across five different intervals in Figure 75; A1 and T75, A1 and T1 and A3 and T1. This depicts the overall escalation process averaging over 60% vitality.

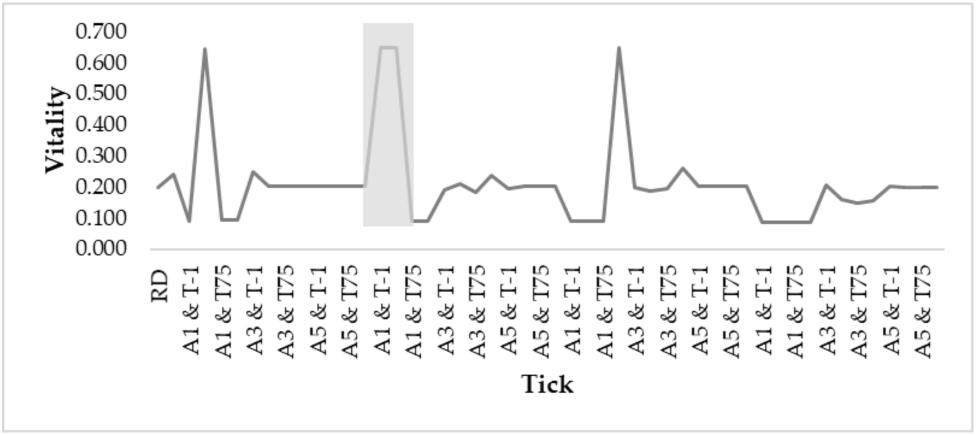


Figure 73. A plot showing the evolution of the vitality for action three (3).

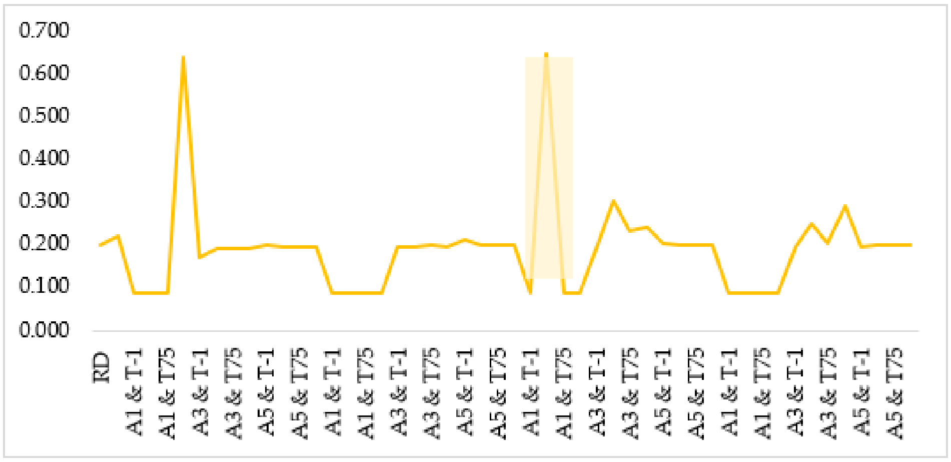


Figure 74. A plot showing the evolution of the vitality for action four (4).

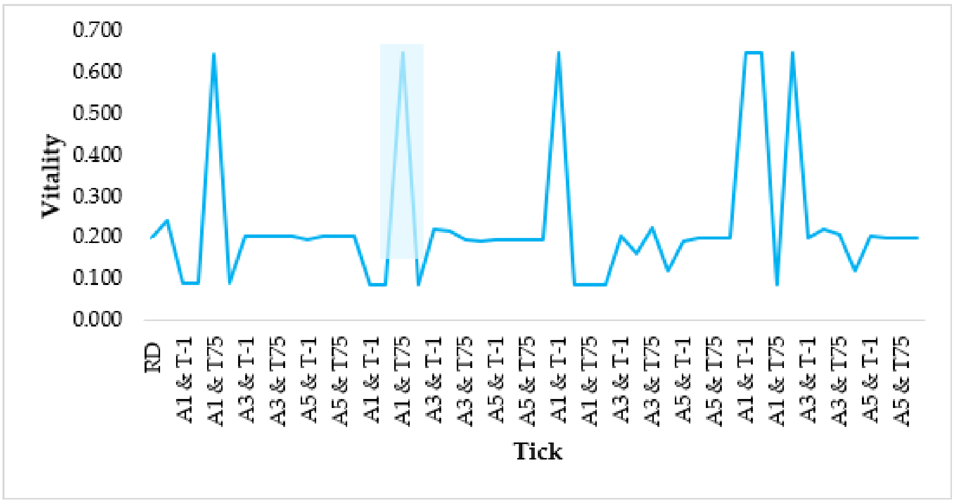


Figure 75. A plot showing the evolution of the vitality for action five (5).

Figure 76 illustrates an ideal escalation process encompassing five actions with their vitality outcomes according to corresponding threshold values. The highlighted areas (pink) in Figure 76 are significant as probable escalation decision points. Since survivability is stochastic phenomena (Ren *et al.*, 2007)(Oreifej *et al.*, 2018), path-dependent and unpredictable, vitality in any simulation run is not representative of decision-making point. Instead, these are points of reference or data sample points from a continuous distributions of probable survivability lifecycles.

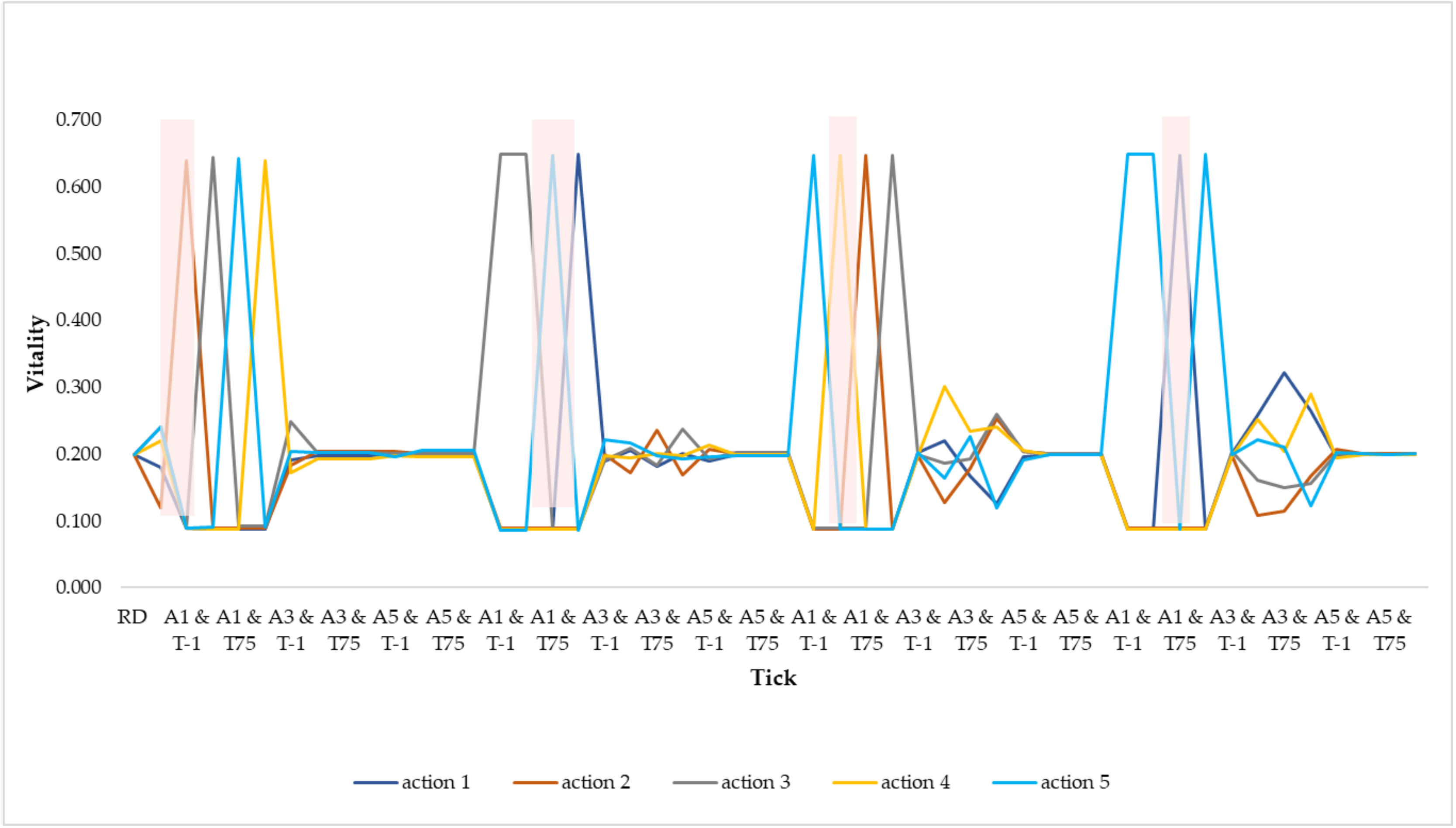


Figure 76. An overall plot showing the evolution of vitality for five actions and the overall survivability

This section presents simulation results for the general evolution of VM vitality over a simulation run. Moreover, it identifies unique patterns that leverage vitality among preferred VMs. Table 28 presents a summary sample for the overall vitality evolution presented in Figure 77 below.

Table 28. Sample summary of overal vitality evolution data over 10 ticks

Max- A_VMs	0.345	0.343	0.389	0.423	0.442	0.455	0.465	0.472	0.476	0.479
Max- C_VMs	0.000	0.049	0.061	0.075	0.098	0.129	0.159	0.183	0.201	0.214
Max- S_VMs	0.345	0.336	0.326	0.321	0.310	0.298	0.287	0.278	0.270	0.264
Max- V_VMs	0.346	0.377	0.369	0.360	0.357	0.355	0.353	0.351	0.350	0.349
Min- A_VMs	0.316	0.310	0.346	0.377	0.400	0.417	0.427	0.433	0.437	0.441
Min C_VMs	0.000	0.010	0.017	0.024	0.033	0.045	0.067	0.091	0.116	0.135
Min S_VMs	0.316	0.276	0.264	0.255	0.244	0.233	0.226	0.220	0.216	0.212
Min V_VMs	0.329	0.357	0.348	0.340	0.333	0.328	0.325	0.323	0.322	0.321

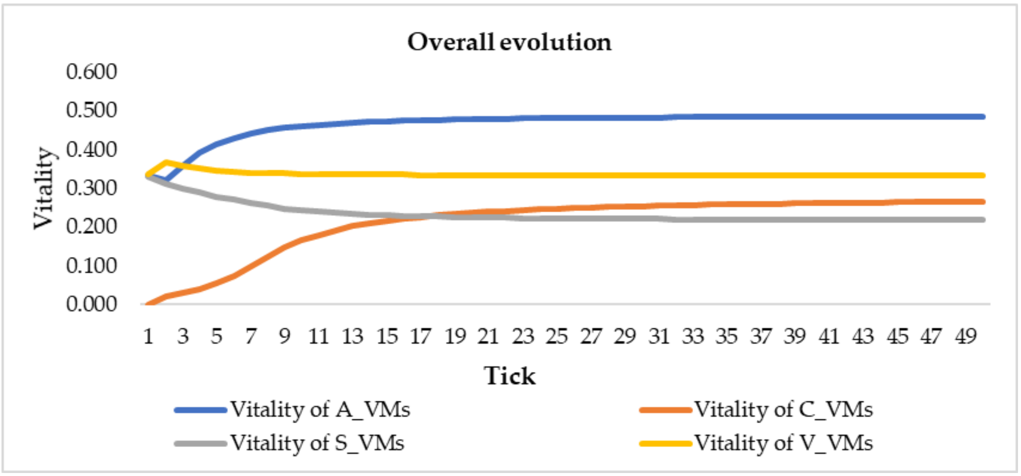


Figure 77. The plot of overall evolution of VMs' vitality over simulation

Figure 77 is a plot highlights the evolution of VMs' vitality over a simulation run. As is expected, attacking VMs have overall higher vitality due to the model design. Interestingly, however, compromised VMs show a relative vitality improvement. Vulnerable VMs have relatively unstressed evolution as vitality is largely stable, while survivable VMs indicate stressed evolution.

Figure 78 shows the vitality of VM 491 in different states of compromise; vulnerable, attack, corrupt and survive. The appendices section (Appendix C) provides the complete simulation data. Figure 79 shows the plot of the vitality of vulnerable VMs in simulation groups 7, 8 and 9. In both graphs, vulnerable VMs gain significant vitality over the simulation run. In Figure 78, VM 491 sharply increases and achieves maximal vitality from its vulnerable state as compared to the compromised state. In Figure 78, vulnerable VMs 488 and 489 with similar gains in vitality but achieving different levels, i.e. VM 489 achieve almost 50% of VM 488 vitality.

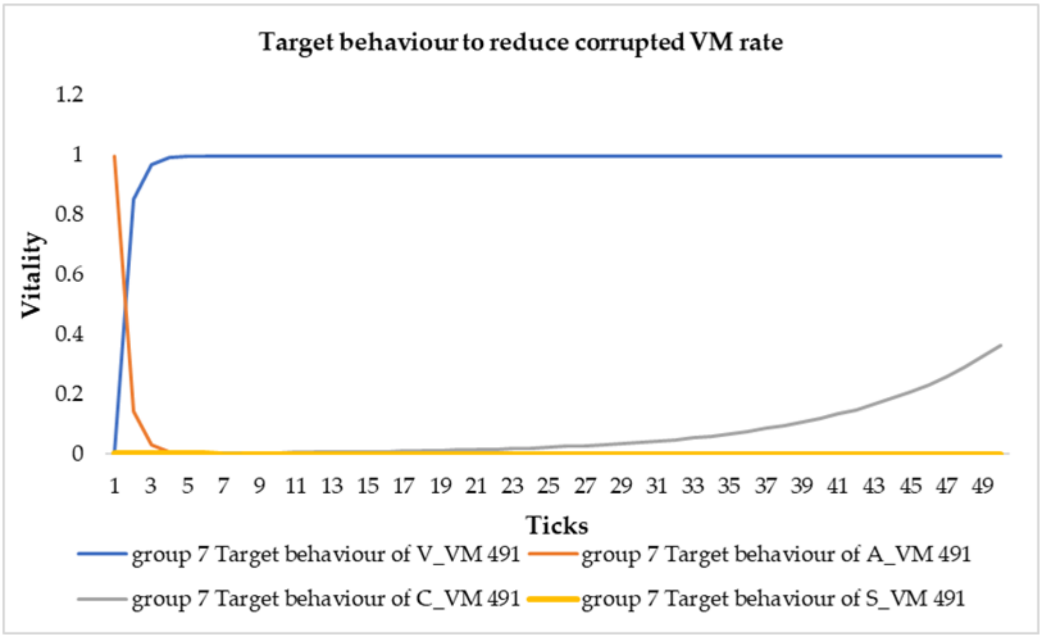


Figure 78. The plot of vitality of VM 491 in simulation group 7. This data shows positive vitality on corrupted VMs over a simulation run

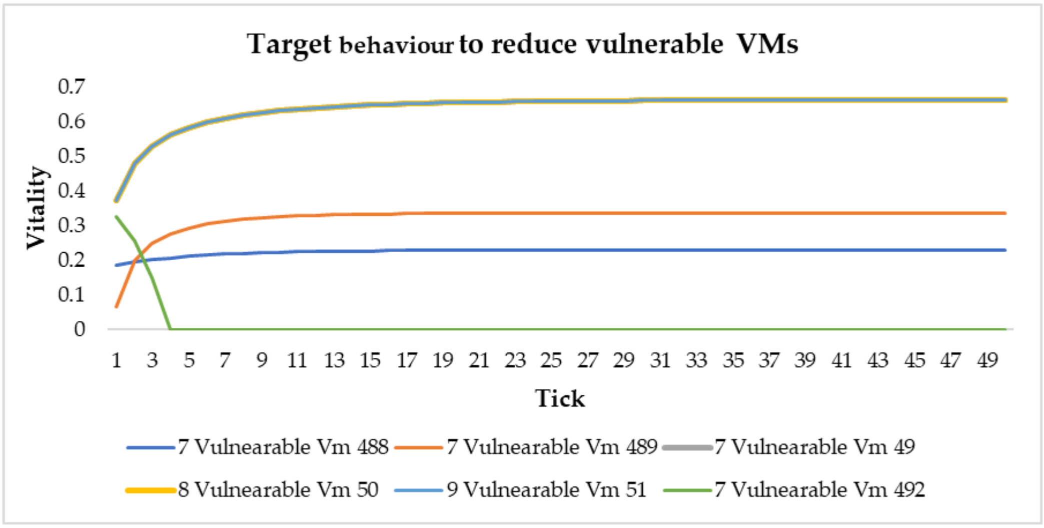
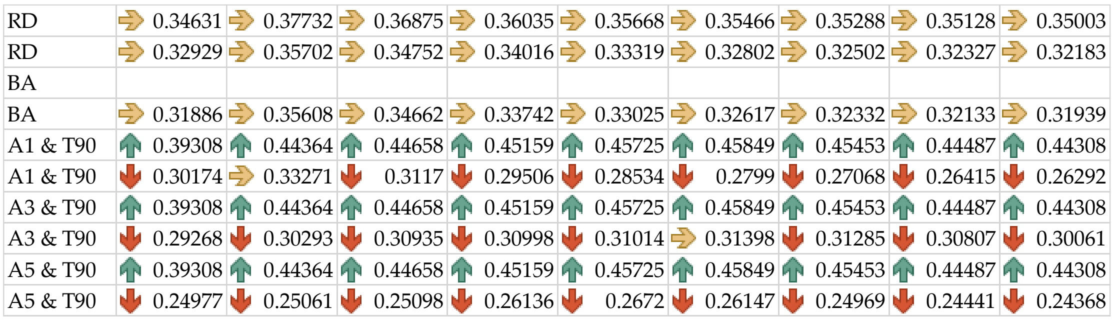


Figure 79. The plot of vitality of vulnerable VMs in groups 7, 8 and 9. This data shows positive vitality on vulnerable VMs over a simulation run

7.4.3 Analysis

Simulations in the thesis gives direction for future research. A similar implementation is scalable and useful for developing larger adaptive software systems than was previously possible (Kwiatkowska, Parker & Qu, 2011). Implementing LV's predator-prey model as the base model for the survivability solution of VMs ensures the reliability of the numerical formulations in this experiment. The main concept within Lotka and Vito Volterra's model (Luo, He & Li, 2004)(Rozenfeld *et al.*, 2006)(Campillo & Lobry, 2012)(da Silva Peixoto, de Barros & Bassanezi, 2008) is that population changes are a result of a combination of forces, including the rate of predation and the interaction between species. This model is adapted by (Pinol & Banzon, 2011) who introduces a Verhulst factor, i.e. survival probability as a capacity to recover from compromise. These results show that the system has good stability, at least over the simulation period. The small oscillations in survivability rate and the average vitality of survivable VMs over both, evolution over a simulation run and the simulation time, suggest that the system is stable. Python in many research works (Klugl & Bazzan, 2012)(Manapragada, Webb & Salehi, 2018), especially for analysing quantitative data, particularly to handling complex data. Its use in this experiment ensures that the simulation code is easily debugged to eliminate procedural errors in the simulation input and, or output.

Figure 80 suggests that the vitality of vulnerable VMs is stable when considering random actions (note that green arrows indicate an increase, red indicate a decline and orange arrows indicate no change). As shown in this sample analysis, it can also be observed that at the end of the simulation, more than half of the vulnerable VMs are survivable albeit with low vitality.



RD	⇨ 0.34631	⇨ 0.37732	⇨ 0.36875	⇨ 0.36035	⇨ 0.35668	⇨ 0.35466	⇨ 0.35288	⇨ 0.35128	⇨ 0.35003
RD	⇨ 0.32929	⇨ 0.35702	⇨ 0.34752	⇨ 0.34016	⇨ 0.33319	⇨ 0.32802	⇨ 0.32502	⇨ 0.32327	⇨ 0.32183
BA									
BA	⇨ 0.31886	⇨ 0.35608	⇨ 0.34662	⇨ 0.33742	⇨ 0.33025	⇨ 0.32617	⇨ 0.32332	⇨ 0.32133	⇨ 0.31939
A1 & T90	↑ 0.39308	↑ 0.44364	↑ 0.44658	↑ 0.45159	↑ 0.45725	↑ 0.45849	↑ 0.45453	↑ 0.44487	↑ 0.44308
A1 & T90	↓ 0.30174	⇨ 0.33271	↓ 0.3117	↓ 0.29506	↓ 0.28534	↓ 0.2799	↓ 0.27068	↓ 0.26415	↓ 0.26292
A3 & T90	↑ 0.39308	↑ 0.44364	↑ 0.44658	↑ 0.45159	↑ 0.45725	↑ 0.45849	↑ 0.45453	↑ 0.44487	↑ 0.44308
A3 & T90	↓ 0.29268	↓ 0.30293	↓ 0.30935	↓ 0.30998	↓ 0.31014	⇨ 0.31398	↓ 0.31285	↓ 0.30807	↓ 0.30061
A5 & T90	↑ 0.39308	↑ 0.44364	↑ 0.44658	↑ 0.45159	↑ 0.45725	↑ 0.45849	↑ 0.45453	↑ 0.44487	↑ 0.44308
A5 & T90	↓ 0.24977	↓ 0.25061	↓ 0.25098	↓ 0.26136	↓ 0.2672	↓ 0.26147	↓ 0.24969	↓ 0.24441	↓ 0.24368

Figure 80. Illustrating vitality changes for vulnerable VMs.

Since the initial simulation values are generated randomly, even if the best action is performed, the vulnerable VMs might not survive. This implies two possibilities. The results suggest that with only random actions performed over the time, the V_VMs are easily compromised and hence only a few of them avoid the attackers (less than 10%). This is an expected outcome as vulnerable VMs are not designed to attack in this experiment; Attack VMs are nonetheless in most of the simulation able to attack continuously. Alternatively, the results suggest that the majority of C_VMs may ultimately be compromised since exploited vulnerabilities result in compromise. Nonetheless, even with the 10% of vulnerable VMs with higher vitality, it can be observed that some behaviours correspond to an ideal system as hypothesised in section 1.2. In cloud environments, these behaviours illustrate two distinct cases; one where vulnerable VMs invoke a suicidal instruction (Hirai, 2017) (and shutdown of an attack) before being compromised (survival rate over corrupted rate). Alternatively, the case where vulnerable VMs are successfully recovered from compromise despite vitality going close to the dead set, i.e. close to 0.

Random behaviour seems better for the final average vitality. This outcome results from the simulation goal to enhance the survival rate and not the vitality. In the comparisons between the best actions and shared memory of 5 actions, the simulation moves closer to the best actions than the random one. Moreover, a random action (RD) can outperform the best action (BA). Logically, this is due to the existence of a randomised parameter that is influencing the interactions between virtual machines. A random factor in this sense represents an external element affecting the simulation. For instance, a second prey or introducing poison during simulation. Thus, as assumed, the best actions are overall outperforming the random actions to the rate of survivability.

The results obtained by performing the best actions are slightly close to the random one but are still slightly above. It is key to note that the current model favours attack VMs and therefore, where a vulnerable VMs is initialised with a random vitality, it might not be able to survive even if it corresponds to the best action possible. Another possibility for the inefficiency of the random action is the vitality that each VM is assumed to possess at the beginning of the experimentation, i.e. a random vitality [0;1]. For instance, where the vitality received approaches zero (0), e.g. 0.00001, the best action could be one that reduces the damage received as much as possible. It stands to reason that under of UUURs threats, and there is no best action to address the current threat, escalation implies that the best action available is an aggressive action such as "suicide". In this sense, this suggests that immediately executing an aggressive action such as suicide is the best action to avoid malware or a virus from spreading in a VM cluster or network and ensure survivability.

7.4.3.1 Hypothesis testing

The hypothesis under investigation is that prey-inspired survival actions against known and unknown predators, are applicable to enhance survivability in cloud environments. Such survivability is integrated into the management and control system and the user-space, to enable user-level input at run-time. Put more accurately, this implies that if an action increases the vitality of VMs in a shorter time difference than another action, the action improves survivability. Thus, hypothesis testing aims to prove that the escalation concept does not arrive at a null hypothesis, i.e. what is perhaps the default state of "the world". Notwithstanding existing debates, for and against the application of the null hypothesis for hypothesis testing, there is convergence to what elements inform its adequacy. Summarily, that a null hypothesis is possible, additionally, that experiment results are consistent with a null hypothesis, and finally, that experiment intended to find and effect. This section therefore evaluates how close the prey-inspired survivability hypothesis is to survivability in real cloud environment. Whereas a range of techniques and methods, for instance the KS test and specifically, the goodness of fit (Magalhães *et al.*, 2015) have a successful application, their strict use in this research is limited. Thus, for simplicity, evaluating the hypothesis error bars will be used.

While this method assesses how close the mean statistic is the true mean, error-bar plots have more value and tend to be more persuasive than statistical testing, however, this is not a consensus view among researchers. According to (Motulsky, 2002), error bars do not explicitly identify the statistical significance of data, and randomness complicates hypothesis testing. It is nonetheless a useful method to produce evidence that the best conclusion to a hypothesis has been reached (Munger, 2008). In order to conclude data being analysed, it is vital to assess whether bars overlap or do not overlap among different data groups. If error bars do not overlap, this concludes that data to be different from other data. Inversely, if errors bars overlap, it can be concluded that data within overlapping data groups is not different.

Figure 81 is an illustrative example of the preceding and shows the error bar plot of data on vitality of vulnerable, compromised, survivable and attack VMs. This data shows that for individual plots in Figure 67. Figure 81 data shows that attacking VMs have the highest vitality whereas corrupted VMs have the least vitality. Since data on compromised VMs does not overlap with other data, it can be conclusively noted that compromised VMs have low vitality and therefore are logically less survivable. However, the same cannot be concluded about attacking or vulnerable VMs since error bars show an overlap among data groups.

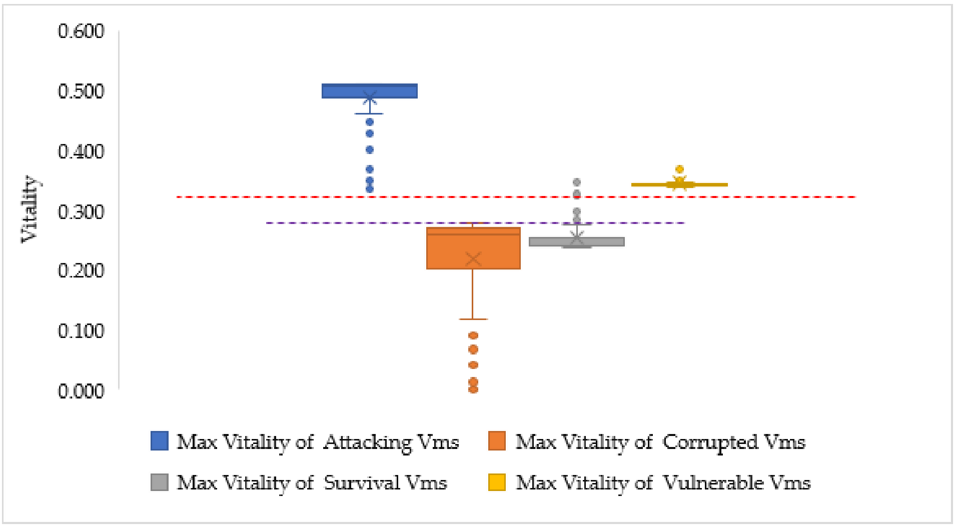


Figure 81. A within-group inference error bar plot: Maximum vitality of A_VMs, C_VMs, S_VMs and V_VMs

Thus, this data is said to be inconclusive. Since hypothesis testing in this section seeks to test H1, the following deductions are presented for simplification.

- Escalating actions that invoke a positive increase in vitality on vulnerable and compromised VMs enhance survivability
- A vitality approaching or equal to 1 implies increasing survivability of VMs other than A_VM.
- Actions that do not change vitality result in Null hypothesis.

Figure 82 shows data of VM 491 whose vitality behaviour results are analysed below it. As noted earlier, the data suggests that actions upon VM 491 improve survivability. Hence, to test the escalation hypothesis (H1), the error bar plot below is analysed, and conclusions are drawn below. It is conclusive that vulnerable VMs have reached the highest vitality since V_VM data does not overlap with any other data groups. It shows that 1 and 2 directly satisfy the alternative hypothesis (H1). Based upon the experimental assumptions presented earlier, i.e. vulnerable VMs do not attack but attack VMs do, the expected outcome is that A_VM has higher vitality than V_VMs. The foregoing conclusively satisfies items 1 and 2. Logically, it can also be inferred that this supports the hypothesis that prey-inspired actions can be applied as an unconventional method to enhance survivability in the cloud.

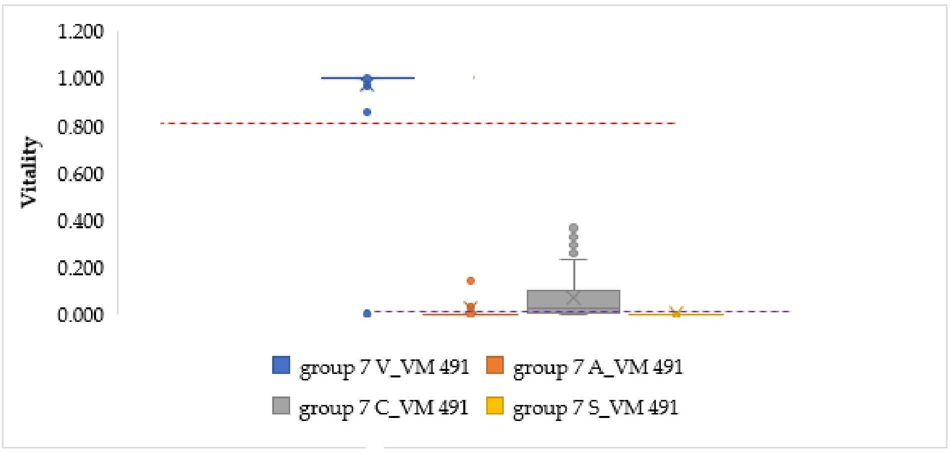


Figure 82. Within-group inference error bar plot: Vitality of VM 491 in the attack, compromised, survivable and vulnerable states

Figure 83 is an error bar plot for group 0 VM 67. By applying the overlapping and non-overlapping concept above, survivable VM data does not overlap with any other VM state data. Hence, it can be observed that items 2 and 3 of the H1 above are satisfied. As noted earlier in this analysis, compromised, survivable and vulnerable VMs are not designed to attack in this experiment but attack VMs are.

Based upon earlier assumptions that V_VMs, C_VMs and S_VMs can execute prey-inspired actions against attack, the experiments seek to evaluate how compromised, survivable and vulnerable VMs' respond to attack. S_VMs' vitality does not overlap with any other state. In addition, the vitality of S_VM 67 is over 0.6, which implies a significant statistical difference to the attack state at 0 vitality. The foregoing conclusively satisfies items 1 and 2 of the H1 above.

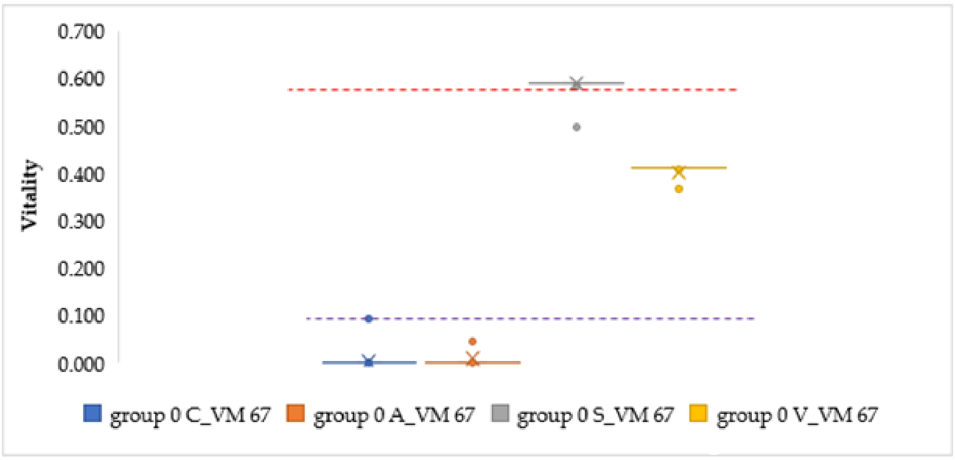


Figure 83. Within-group inference error bar plot: Vitality of VM 67 in the attack, compromised, survivable and vulnerable states.

Considering V_VM 67 with A_VM 67 and C_VM 67 (attack and compromised states), V_VM 67 does not overlap and vitality is significantly higher. Due to the initial experiment setup, the foregoing implies that vulnerable VMs have gained vitality due to the prey-inspired actions and hence satisfies conclusively satisfies 1 and 2 of the H1 above. Nonetheless, C_VM 67 and A_VM 67 vitality data overlap and therefore does not conclusively indicate a clear effect prey-inspired actions impose. These findings conclusively support the hypothesis (H1) that escalation mechanisms for avoiding predation can be applied to enhance the survivability of cloud computing systems. By analysing results and identifying target or preferable behaviours (in terms of survivability) of group 0; VM 67 and group 7; VM 491 suggest that it was possible to proactively and intelligently manage and control survivability by deciding the best actions to achieve targeted survivability outcomes. As Figure 84 seems to suggest, the task to enable survivability focuses upon the capabilities of vulnerable VM 491, and both vulnerable VM and survivable VM 67.

V_VM 491	⬇ 0.001	⬆ 0.853	⬆ 0.968	⬆ 0.991	⬆ 0.996	⬆ 0.997	⬆ 0.997	⬆ 0.997	⬆ 0.997
A_VM 491	⬆ 0.995	⬇ 0.144	⬇ 0.029	⬇ 0.006	⬇ 0.001	⬇ 0.000	⬇ 0.000	⬇ 0.000	⬇ 0.000
C_VM 491	⬇ 0.000	⬇ 0.002	⬇ 0.002	⬇ 0.002	⬇ 0.002	⬇ 0.003	⬇ 0.003	⬇ 0.003	⬇ 0.004
S_VM 491	⬇ 0.003	⬇ 0.002	⬇ 0.001	⬇ 0.001	⬇ 0.001	⬇ 0.001	⬇ 0.000	⬇ 0.000	⬇ 0.000
C_VM 67	⬇ 0.000	⬇ 0.093	⬇ 0.009	⬇ 0.000	⬇ 0.000	⬇ 0.000	⬇ 0.000	⬇ 0.000	⬇ 0.000
A_VM 67	➡ 0.384	⬇ 0.044	⬇ 0.002	⬇ 0.000	⬇ 0.000	⬇ 0.000	⬇ 0.000	⬇ 0.000	⬇ 0.000
S_VM 67	➡ 0.590	➡ 0.497	➡ 0.582	➡ 0.590	➡ 0.590	➡ 0.590	➡ 0.590	➡ 0.590	➡ 0.590
V_VM 67	⬇ 0.025	➡ 0.366	➡ 0.408	➡ 0.410	➡ 0.410	➡ 0.410	➡ 0.410	➡ 0.410	➡ 0.410

Figure 84. Illustration of vitality changes for VMs 491 and 67

The key question for achieving survivability through escalating VM actions revolves around the underlying decision support processes. As has been shown earlier, maintaining survivability requires timely decisions. Such decisions encompass survivability targets, survivability preferences and the best survivability actions that are necessary to support critical cloud services.

7.5 Summary

This chapter has investigated the behaviour of VMs in different states of compromise, i.e. vulnerable, attacked, compromised and demonstrated vitality dynamics when survivability actions are applied. These experiments should be considered what prey-inspired escalation is capable of, in relation to survivability as an additional complement of security, rather than as an alternative to security.

The experiment result indicates that escalating actions can improve the overall survivability of VMs. The results of group 0 VM 67 and group 7 VM 491 provide conclusive evidence to support the hypothesis that prey-inspired escalation can improve survivability in cloud computing. The results in this chapter support the hypothesis under test.

Chapter 8 Conclusion and recommendations

This chapter summarises the research works, including an outline of the research process and the research's contributions. It evaluates the research limitations and recommends the directions for future research work.

8.1 Research process

It was clear at the beginning of this research that within the cloud computing domain (academia in particular), there was no consensus concerning the maturity perceptions of cloud computing. On the one hand, extensive adoption by industry and international organisations suggested cloud computing had reached maturity with wide use and acceptability. On the other hand, its adoption was viewed with scepticisms among dissenters, foremost based upon cloud computing's relative novelty. In addition, the vast reportage of security issues was viewed as an indication of the immaturity. From a research perspective, this implied that several alternatively competing approaches could have been followed. For instance, explorative research in the former or descriptive research in the case of the latter.

Another observation was that the application of biological inspiration for cloud computing security was underwhelming. For this reason, bio-inspired research suggested itself as a possible approach based upon its potential for novel research. The predator-prey system was an under-researched area and a potential niche in bio-inspired cloud computing context. Hence, mechanisms in non-extinct prey seemed most suited to the research concept. Predation avoidance and anti-predation behaviours and mechanisms were therefore identified as central to the technical conceptualisations and developments of this thesis. This notably was a multi-faceted task, cutting across at least two domains, which both required comprehensive investigation and analysis. Subsequently, this had an impact on the overall progress as this part of the research was prolonged concerning the overall timeline. Achieving prey-inspired survivability was challenging on several fronts. Foremost, prey survivability ideally required learning among autonomous agents to mimic nature's animals. In a real nature-like scenario, agents learn, decide and act autonomously. Interactions are local, and decisions and actions are taken towards a global goal. Hence, in the process of researching the technical models and implementations, much time was spent on attempts to retain the characteristics of the natural prey system.

8.2 Thesis summary and contributions

This thesis aimed to develop a prey-inspired solution to address cloud computing's security challenges. The predator-prey system was chosen in view of non-extinct prey communities' survival against predators. A formal method for the interdomain bio-inspired design process was implemented, specifically by following a problem-driven holistic approach to gather, translate and transfer concepts from nature to cloud computing. Moreover, TRIZ was utilised to proffer specific solutions for cloud computing by addressing known contradictions in published literature. Pi-CCSF is then designed, verified and validated in a multi-step evaluation approach (Mehresh & Upadhyaya, 2015). Thus, the following high-level challenges are particularly addressed in this research. First, how to manage and control security in a manner that complements existing approaches yet leveraging the mission to provide services regardless of threat type continuously. Also, how to leverage prey survival mechanisms to achieve enhanced survivability in cloud computing environments.

Chapter 1 introduces the research and outlines the motivating themes around the challenges mentioned above. It presents the research hypothesis which defines the bio-inspired themes that develop in the thesis chapters. Furthermore, this chapter introduced the research aim and objectives, the research contributions and the research methodology followed to achieve them. This chapter's conclusions form the remarks which inform the directions of Chapter 2. Summarily, that inadequate security approaches contribute to the current cloud computing security landscape. Also, the unpredictability of UUURs particularly complicates the ability to detect and predict threats. Finally, that instead of security alone, survivability has promise as a suitable property to address the risks imposed by UUURs. There is potential in bio-inspired systems which have established survivability solutions.

Chapter 2 presents the literature review to establish the current state of cloud computing security. It conducts a comparative analysis of the traditional and cloud security view, against the backdrop of current security challenges. This analysis exposes gap areas with respect to how challenges are handled in existing countermeasures. In addition, this chapter presented a literature review of the survivability concept to establish a critical understanding of its traditional use compared to the cloud computing context. Finally, the literature review also presented the current state-of-the-art of the use of the bio-inspired approach in computing. This review also aimed to identify suitable mechanisms to address gap areas identified in the previous sections (Section 2.2. and Section 2.3). The significance of this section is as follows: It presented a comprehensive state-of-the-art around the thesis's motivating themes. It complements the research methodology by allowing subjective interpretations of subject areas while also allowing for objective and logical inference and measurability of verifiable facts. This chapter's work culminates into the contribution of Chapter 3. The contributions provided by these chapters are the following.

Chapter 3 proposes a holistic taxonomy for cloud computing security challenges based upon the findings in section 2.2**.** The holistic taxonomy forms the foundational basis for cross-domain design. A formal definition of a holistic taxonomy is proposed to emphasise comprehensive understanding of cloud security challenges from a source or origin perspective. Most significantly, the holistic taxonomy satisfies H1 of the research hypothesis and is utilised in the discussions in section 5.6.

Chapter 4 outlines a 3-step method to systematically transfer concepts from nature to cloud, and TRIZ-based approach for developing creative solutions to achieve survivability while addressing cloud computing challenges. This chapter addresses the research hypothesis and satisfies the applicability aspect of H1 (escalating survival behaviours and mechanisms). Bio-inspired design theory is seen from the biology to computing or engineering lenses to exploit design contexts including creative design, complex system design, sustainable design, etc.

Chapter 5 presents a conceptual prey-inspired cloud computing survivability framework (Pi-CCSF). Pi-CCSF is significant as it supports the extension of existing frameworks for prey-inspired survivability and model-based analysis of systematically deduced survivability requirements. Such satisfy the hypothesis for escalating survivability design principles and decision techniques presented (H2).

Chapter 6 is dedicated to the decision system (DS) component of Pi-CCSF and formulates a target-based decision-making technique (TBDM) for managing survivability decision-making. Instead of specifying decision-making in line with the traditional sense of survivability (as a constraint with an allowable level of service loss), TBDM specifies decision-making with respect to survivability targets. This is significant as decision-making can be adaptive to evolving survivability targets and controlled and prioritised on the fly via the SSM (Section 5.3.1) according to requirements.

Notwithstanding this significance, a finding borne out of this chapter is the need to integrate the DS within the Pi-CCSF simulator and evaluate the efficacy of TBDM. Nonetheless, it is also notable that the TBDM technique addresses the research hypothesis by bringing evolving and uncertain survivability information closer to survivability decision processes, survivability preferences and attitudes. Moreover, this technique is observed to ensure that even under unpredictability, the decision model is focused on achieving survivability outcomes, i.e. targets.

In Chapter 7, a python simulator (Pi-CCSF simulator) is developed and implemented to evaluate the applicability of the conceptual Pi-CCSF. This is a custom-built environment to understand Pi-CCSF's implications in a practical application of escalating survivability actions upon vitality of VMs in various states of compromise and overall survivability. The experiment results suggest that certain actions improve the vitality of vulnerable VMs and average overall survivability. Analysis of experimental results identify outright instances where corrupted, vulnerable and survivable VMs gain significant vitality, thereby informing desirable escalation configurations. By analysing these VMs, e.g. group 7's VM 491 and group 0's VM 69, evaluations suggest that Pi-CCSF offers the potential for enhancing survivability. In addition, the hypothesis testing on VM data conclusively supports the view that escalating actions can enhance survivability. However, there is a need for further development of the simulator, with complete integration of all components and more exhaustive experiments to reach a firm conclusion.

This thesis's main contribution is the prey-inspired survivability framework (Pi-CCSF) which is presented in Chapter 5 and Chapter 6 and evaluated in Chapter 7. Other contributing areas of this research include the holistic taxonomy presented in Chapter 3 and the application of the TRIZ method for prey-inspired cloud computing survivability presented in Chapter 4. Some of these contributions are published in peer-reviewed journals and conferences. The following broadly represent the main contributions of this work. Based upon a comprehensive review of the literature on cloud computing security and bio-inspired systems, cloud security taxonomies, survivability and bio-inspired approached are discussed. Each is analysed and propositions are posited to address identified gap areas. Thus, contributions provided by these chapters are the following:

1. A holistic taxonomy of cloud computing security challenges (Publication PR 2 is the contributing publication to this section),

    1. A theoretical model from mimicking predator-prey systems exhibited in nature (PR3, PR4 and PR6 are some contributing publications to this section).

Section 4.2, Section 4.3 and Section 4.4. resented the TRIZ-based method for the prey-inspired survivability design, the prey-inspired cloud computing survivability framework, the target-based decision-making technique to achieve survivability targets and a practical framework simulation environment to evaluate the Pi-CCSF, respectively.

2. A conceptual framework: Pi-CCSF is developed, discussed and evaluated. Analysis of experimental data conclusively suggests that escalation actions improve overall survivability. Hence, Pi-CCSF offers conclusive support to the hypothesis under investigation. (PR1, PR3 and PR 6 are some contributing publications to this section).

3. A target-based decision-making technique (TBDM); a component of Pi-CCSF, is presented to address decision-making processes for selecting survivability actions and targets, prioritising survivability actions during escalation and prioritising contextual survivability information relative to CC or CSP under unpredictable scenarios.

4. A simulation environment is developed and customised to evaluate Pi-CCSF's escalation concepts. The experimentations and hypothesis testing on experimental data proves some merit to the H1 and H2 of the hypothesis under investigation.

8.3 Limitations

While the aim of the research and the objectives set out have been fulfilled, this author contends with some limitations borne out of the research. These may be addressed in future work.

- The simulator is not a full implementation of the prey-inspired survivability concept in Pi-CCSF. Nonetheless, it provides the core functions of escalation proposed in Pi-CCSF and serves as a proof of concept to the prey-inspired approach for survivability that Pi-CCSF proposes.

- Pi-CCSF depends on the DS component to institute optimal survivability decisions. Although the DS component is not integrated into the simulator. Nonetheless, the Pi-TBDM technique developed in Chapter 6 shows how Pi-CCSF prioritises survivability decisions when a VM's vitality drops below an expected threshold. Moreover, how a preceding decision impacts upon other subsequent decisions and countermeasure selection. The above is critical as Pi-CCSF the notion of decentralised management and control to be key.

- Pi-CCSF is not developed into a system prototype. Hence, the evaluations suffer from inherent limitations of simulation as compared to prototype. For instance, the simulations are performed in a constrained environment, which does not resemble real cloud computing environments. Nonetheless, the results indicate the behaviours of VMs in different states of compromise after employing prey-inspired escalation. These results provide a foundation for further development and research.

8.4 Future work

Despite the strengths of the simulated models, the simulation environment (local machine) and the model, is affected by state explosion where the size of the model exponentially increases with the size of the modelled system. Specifically, computational overheads limit the realistic scaling of the cloud environment. For instance, the python implementation simulates models associated with a shared memory configuration. However, a large system configuration that characterises the learning attributes of survivable preys makes the simulated system complex. Two further research directions are worth exploring to bring Pi-CCSF closer to cloud environments and improve the effectiveness and/efficiency of its various components described above.

Collective actions and strategic behaviour are two key components of survivability in natural preys. The former is central to the survival of prey animals who live in groups, while the latter is present in prey's escalating predation-avoidance and anti-predation techniques. Thus, collective action is seen as outcomes of local interactions among prey agents, while strategy construed as coordination protocols (which can be either explicit or implicit or both). Synthesising and integrating prey's survival behaviours requires effective translation of natural language ecological terms. This process is error-prone and thus can be improved upon.

Moreover, self-adaptation entails meeting goal changes and calls for automated synthesis to achieve survivability in dynamic and complex cloud environments. Automated synthesis generates survivability specifications to characterise a range of adaptation abilities. This approach will improve upon current survivability characterisation methods which rely on experts or an engineer's prior analytical models. Due to UUURs, prior analytical model analysis is currently prone to produce high occurrences of inaccurate information (false positives and false negatives). These inaccuracies are significant shortcomings in survivability engineering and design. The following research directions are worth exploring:

- Further development to integrate the DS into the Pi-CCSF simulator and investigate the overall efficacy of the complete prey inspiration as opposed to independent individual modules.
- Implementation of Pi-CCSF with extended algorithms and simulations models in a real or near environment. For instance, using machine learning approaches for the proposed prey mechanisms.
- It would be an interesting to develop prey-inspired survivability for testing and experimentation in any environment.

References

'Amiri, E. (2009) *CA Identity Manager: Capabilites and Architecture.*

Active, I., Systems, D. & Networks, P. (n.d.) *InfoSec Reading Room Implementing Active Defense Systems on Private.*

Adams, K.M. (2015) Understandability, Usability, Robustness and Survivability. In: *Nonfunctional Requirements in Systems Analysis and Design*. pp. 201–220.

Adhikari, V.K., Guo, Y., Hao, F., Hilt, V., et al. (2015) Measurement Study of Netflix, Hulu, and a Tale of Three CDNs. *IEEE/ACM Transactions on Networking*. [Online] Available from: doi:10.1109/TNET.2014.2354262.

Afrin, T. & Yodo, N. (2019) Resilience-Based Recovery Assessments of Networked Infrastructure Systems under Localized Attacks. *Infrastructures*. [Online] 4 (1), 11. Available from: doi:10.3390/infrastructures4010011.

Ahamed, F., Shahrestani, S. & Ginige, A. (2013) Cloud computing: Security and reliability issues. *Communications of the IBIMA*. 2013, 1.

Ahmad, M., Belloir, N. & Bruel, J.M. (2015) Modeling and verification of Functional and Non-Functional Requirements of ambient Self-Adaptive Systems. *Journal of Systems and Software*. [Online] 107, 50–70. Available from: doi:10.1016/j.jss.2015.05.028.

Ahmed, A.A., Sadiq, A.S. & Zolkipli, M.F. (2016) Traceback model for identifying sources of distributed attacks in real time. *Security and Communication Networks*. [Online] Available from: doi:10.1002/sec.1476.

Aibin, M., Walkowiak, K. & Sen, A. (2017) Software-defined adaptive survivability for elastic optical networks. *Optical Switching and Networking*. [Online] 23, 85–96. Available from: doi:10.1016/j.osn.2016.06.008.

Alam, T. (2019) Iot-fog: A communication framework using blockchain in the internet of things. *International Journal of Recent Technology and Engineering.*

Albanese, M., Jajodia, S. & Venkatesan, S. (2018) Defending from Stealthy Botnets Using Moving Target Defenses. *IEEE Security and Privacy*. [Online] Available from:

doi:10.1109/MSP.2018.1331034.

Alert Logic (2016) *DEFEND YOUR DATA FROM RANSOMWARE WITH ALERT LOGIC.* 4 (0).

Alert Logic (2015) *Vulnerability Response Best Practice: Shellshock Case Study*. 2015.

Ali, A.B., Robson, E. & Boukerche, A. (2016) Performance analysis of bio-inspired scheduling algorithms for cloud environments. In: *IEEE International Parallel and Distributed Processing Symposium Workshop (IPDPSW)*. 2016 pp. 776–785.

Ali, M., Khan, S.U. & Vasilakos, A. V (2015) Security in cloud computing: Opportunities and challenges. *Information Sciences*. 305, 357–383.

Almomani, A., Gupta, B.B., Atawneh, S., Meulenberg, A., et al. (2013) A survey of phishing email filtering techniques. *IEEE Communications Surveys and Tutorials*. [Online] 15 (4), 2070–2090. Available from: doi:10.1109/SURV.2013.030713.00020.

Almorsy, M., Grundy, J. & Müller, I. (2016) An {Analysis} of the {Cloud} {Computing} {Security} {Problem}. *arXiv:1609.01107 [cs]*.

Almorsy, M., Grundy, J. & Müller, I. (2010) An analysis of the cloud computing security problem. *17th Asia-Pacific Software Engineering Conference (APSEC 2010) Cloud Workshop,Sydney, Australia*. [Online] (December), 7. Available from: http://researchbank.swinburne.edu.au/vital/access/services/Download/swin:20103/SOURCE2.

Altshuller, G.S. (1999) *The innovation algorithm: TRIZ, systematic innovation and technical creativity*. [Online]. Available from: http://www.amazon.com/dp/0964074044.

AlZain, M., Pardede, E., Soh, B. & Thom, J. (2012) Cloud computing security: from single to multi-clouds. In: *System Science (HICSS), 2012 45th Hawaii International Conference on*. 2012 IEEE. pp. 5490–5499.

Amaratunga, D., Baldry, D., Sarshar, M. & Newton, R. (2002) Quantitative and qualitative research in the built environment: application of "mixed" research approach. *Work Study*. [Online] 51 (1), 17–31. Available from: doi:10.1108/00438020210415488.

Amazon Web Services (2011) Amazon Elastic Compute Cloud (Amazon EC2). *Amazon Web Services LLC*. [Online] 2010. Available from: http://aws.amazon.com/ec2/.

Anderson, D. & Ellenbogen, K.M. (2012) Learning science in informal contexts-epistemological perspectives and paradigms. In: *Second International Handbook of Science Education*. [Online]. pp. 1179–1187. Available from: doi:10.1007/978-1-4020-9041-7_78.

Andersson, J., De Lemos, R., Malek, S. & Weyns, D. (2009) Modeling dimensions of self-adaptive software systems. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. [Online]. 2009 pp. 27–47. Available from: doi:10.1007/978-3-642-02161-9_2.

Arcaini, P., Riccobene, E. & Scandurra, P. (2017) Formal Design and Verification of Self-Adaptive Systems with Decentralized Control. *ACM Transactions on Autonomous and Adaptive Systems*. [Online] 11 (4), 1–35. Available from: doi:10.1145/3019598.

Ardagna, C.A., Asal, R., Damiani, E. & Vu, Q.H. (2015a) From Security to Assurance in the Cloud: A Survey. *ACM Comput. Surv.* [Online] 48 (1), 2:1–2:50. Available from: doi:10.1145/2767005.

Ardagna, C.A., Asal, R., Damiani, E. & Vu, Q.H. (2015b) From Security to Assurance in the Cloud: A Survey. *ACM Computing Surveys (CSUR)*. 48 (1), 2.

Armbrust, B.Y.M., Fox, A., Griffith, R., Joseph, A.D., et al. (2010) of Cloud Computing. *Communications of the ACM*. [Online] 53 (4), 50–59. Available from: doi:10.1145/1721654.1721672.

Assunção, M.D., Calheiros, R.N., Bianchi, S., Netto, M.A.S., et al. (2015) Big Data computing and clouds: Trends and future directions. *Journal of Parallel and Distributed Computing*. [Online] Available from: doi:10.1016/j.jpdc.2014.08.003.

Author, M.K. & Author, J.A. (2013) *An Inter-VM Communication Model Supporting Live Migration*. [Online] 63–68. Available from: doi:10.1109/CUBE.2013.22.

Autili, M., Di Ruscio, D., Di Salle, A. & Perucci, A. (2014) CHOReOSynt: Enforcing Choreography Realizability in the Future Internet. *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. [Online] 723–726. Available from: doi:10.1145/2635868.2661667.

Auty, M. (2015) Anatomy of an advanced persistent threat. *Network Security*. [Online] Available from: doi:10.1016/S1353-4858(15)30028-3.

Aviram, A. & Ford, B. (n.d.) No Title. *Determinating timing channels in statistically multiplexed clouds, Mar.2010*.

Baars, T. & Spruit, M. (2012) Analysing the Security Risks of Cloud Adoption Using the SeCA Model: A Case Study. *J.UCS*. 18 (12), 1662–1678.

Balme, G., Hunter, L. & Slotow, R. (2007) Feeding habitat selection by hunting leopards Panthera pardus in a woodland savanna: prey catchability versus abundance. *Animal Behaviour*. 74 (3), 589–598.

Balusamy, B., Sridhar, J., Dhamodaran, D. & Krishna, P.V. (2015) Bio-inspired algorithms for cloud computing: a review. *International Journal of Innovative Computing and*

Applications. 6 (3–4), 181–202.

Baran, B. & Sosa, R. (2000) A new approach for AntNet routing. In: *Computer Communications and Networks, 2000. Proceedings. Ninth International Conference on*. 2000 IEEE. pp. 303–308.

Basu, S., Bardhan, A., Gupta, K., Saha, P., et al. (2018) Cloud computing security challenges & solutions-A survey. In: *2018 IEEE 8th Annual Computing and Communication Workshop and Conference, CCWC 2018*. [Online]. 2018 p. Available from: doi:10.1109/CCWC.2018.8301700.

Bays, L.R., Oliveira, R.R., Barcellos, M.P., Gaspary, L.P., et al. (2015) Virtual network security: threats, countermeasures, and challenges. *Journal of Internet Services and Applications*. [Online] 6 (1), 1. Available from: doi:10.1186/s13174-014-0015-z.

BBC online UK (2018) *Dixons Carphone admits huge data breach*. [Online]. 2018. BBC. Available from: https://www.bbc.co.uk/news/business-44465331 [Accessed: 13 June 2018].

BBC online UK (2015) *TalkTalk hack 'affected 157,000 customers'*. [Online]. 2015. BBC. Available from: https://www.bbc.co.uk/news/business-34743185 [Accessed: 10 June 2018].

BBC online UK (n.d.) *Vodafone Germany hack hits two million customers*. [Online]. BBC. Available from: https://www.bbc.co.uk/news/technology-24063621 [Accessed: 10 June 2018].

Beckmanna, H. (2015) Method for transferring the 40 inventive principles to information technology and software. In: *Procedia Engineering*. [Online]. 2015 pp. 993–1001. Available from: doi:10.1016/j.proeng.2015.12.413.

Behl, A. (2011) Emerging security challenges in cloud computing: An insight to cloud security challenges and their mitigation. In: *Information and Communication Technologies (WICT), 2011 World Congress on*. 2011 IEEE. pp. 217–222.

Behl, A. & Behl, K. (2012) An analysis of cloud computing security issues. In: *Information and Communication Technologies (WICT), 2012 World Congress on*. 2012 IEEE. pp. 109–114.

Bendovschi, A. (2015) Cyber-Attacks – Trends, Patterns and Security Countermeasures. *Procedia Economics and Finance*. [Online] Available from: doi:10.1016/s2212-5671(15)01077-1.

Bernsmed, K., Jaatun, M.G., Meland, P.H. & Undheim, A. (2011) Security SLAs for federated cloud services. In: *Availability, Reliability and Security (ARES), 2011 Sixth*

International Conference on. 2011 IEEE. pp. 202–209.

Betz, D.J. & Stevens, T. (2013) *Analogical reasoning and cyber security*. [Online] Available from: doi:10.1177/0967010613478323.

Bigham, J. (2010) *Security and Survivability of Large Scale Critical Infrastructures*. [Online] 84–85. Available from: doi:10.1007/3-540-36080-8_9.

Bitglass (2014a) *What is a CASB? Cloud Access Security Broker*. 2014.

Bitglass (2014b) *What is a CASB? Cloud Access Security Broker*. [Online]. 2014. Available from: https://www.bitglass.com/casb-cloud-access-security-broker [Accessed: 10 December 2018].

Blue Coat and Symantec (2015) *CloudSOC*. [Online]. 2015. Available from: https://www.elastica.net/applications [Accessed: 20 November 2016].

Bluecoat (2015a) *Cloud Data Protection*. 2015.

Bluecoat (2015b) *Cloud Data Protection*. [Online]. 2015. Available from: https://www.bluecoat.com/en-gb/resources/cloud-data-protection/life-cycle-protection [Accessed: 10 December 2018].

Bohn, R.B., Messina, J., Liu, F., Tong, J., et al. (2011) NIST cloud computing reference architecture. In: *Proceedings - 2011 IEEE World Congress on Services, SERVICES 2011*. [Online]. 2011 p. Available from: doi:10.1109/SERVICES.2011.105.

Bordley, R. & LiCalzi, M. (2000) Decision analysis using targets instead of utility functions. *Decisions in Economics and Finance*. [Online] Available from: doi:10.1007/s102030050005.

Botta, A., de Donato, W., Persico, V. & Pescapé, A. (2016) Integration of cloud computing and internet of things: a survey. *Future Generation Computer Systems*. 56, 684–700.

Boutin, S. (1995) Testing predator-prey theory by studying fluctuating populations of small mammals. *Wildlife Research*. [Online] Available from: doi:10.1071/WR9950089.

Brabazon, A. & O'Neill, M. (2006) *Biologically inspired algorithms for financial modelling*. Springer Science & Business Media.

Bracha, A. & Brown, D.J. (2012) Affective decision making: A theory of optimism bias. *Games and Economic Behavior*. [Online] 75 (1), 67–80. Available from: doi:10.1016/j.geb.2011.11.004.

Brief, S. (2015) *McAfee Cloud Security Platform*.

Buecker, A., Andreas, P. & Scott Paisley (2008) Understanding IT Perimeter Security. *Ibm*. 1–22.

Burke, D. (2007) *AN ABSTRACT OF THE DISSERTATION OF Title: An*

Autoethnography of Whiteness.

Buyya, R., Calheiros, R.N., Son, J., Dastjerdi, A.V., et al. (2014) Software-Defined Cloud Computing: Architectural elements and open challenges. *Proceedings of the 2014 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2014*. [Online] 1–12. Available from: doi:10.1109/ICACCI.2014.6968661.

CA Technologies (2014) *Identity and Access Management as-a-Service: Enabling and Protecting Digital Relationships*. [Online]. 2014. Available from: https://www.ca.com/us/register/forms/collateral/identity-and-access-management-as-a-service-enabling-and-protecting-digital-relationships.aspx [Accessed: 8 September 2015].

Calheiros, R.N., Ranjan, R., De Rose, C.A.F. & Buyya, R. (2009) CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services. *arXiv preprint arXiv:0903.2525*. [Online] 9. Available from: http://arxiv.org/abs/0903.2525.

Calinescu, R., Autili, M., Cámara, J., Di Marco, A., et al. (2017) Synthesis and Verification of Self-aware Computing Systems. *Self-Aware Computing Systems*. [Online] 337–373. Available from: doi:10.1007/978-3-319-47474-8_11.

Calinescu, R., Johnson, K. & Rafiq, Y. (2013) Developing self-verifying service-based systems. In: *2013 28th IEEE/ACM International Conference on Automated Software Engineering, ASE 2013 - Proceedings*. [Online]. 2013 pp. 734–737. Available from: doi:10.1109/ASE.2013.6693145.

Calinescu, R., Johnson, K., Rafiq, Y., Gerasimou, S., et al. (2013) Continual verification of non-functional properties in cloud-based systems invited paper. In: *CEUR Workshop Proceedings*. 2013 pp. 1–5.

Cámara, J., Peng, W., Garlan, D. & Schmerl, B. (2018) Reasoning about sensing uncertainty and its reduction in decision-making for self-adaptation. *Science of Computer Programming*. [Online] Available from: doi:10.1016/j.scico.2018.07.002.

Campillo, F. & Lobry, C. (2012) Effect of population size in a predator-prey model. *Ecological Modelling*. [Online] 246 (C), 1–10. Available from: doi:10.1016/j.ecolmodel.2012.07.015.

Canetti, R., Gennaro, R., Jarecki, S., Krawczyk, H., et al. (1999) Adaptive security for threshold cryptosystems. In: *Advances in Cryptology -CRYPTO - 99*. 1999 Springer. pp. 98–116.

Caro, G. Di & Dorigo, M. (1998) AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*. 317–365.

Caro, T.M. (1986) The functions of stotting in Thomson's gazelles: some tests of the predictions. *Animal Behaviour*. 34 (3), 663–684.

Caron, E., Le, A.D., Lefray, A. & Toinard, C. (2013) Definition of security metrics for the Cloud Computing and security-aware virtual machine placement algorithms. *Proceedings - 2013 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2013*. [Online] 125–131. Available from: doi:10.1109/CyberC.2013.28.

Carvalho, C.A.B. de, Andrade, R.M. de C., Castro, M.F. de, Coutinho, E.F., et al. (2017) State of the art and challenges of security SLA for cloud computing. *Computers and Electrical Engineering*. [Online] Available from: doi:10.1016/j.compeleceng.2016.12.030.

Castro, L.N. De & Zuben, F.J. Von (2002) Learning and optimization using the clonal selection principle. *Evolutionary Computation, IEEE Transactions on*. 6 (3), 239–251.

Chalmers, M. (2013) Design perspectives in visualising complex information. In: *Visual Database Systems 3*. [Online]. p. Available from: doi:10.1007/978-0-387-34905-3_7.

Chang, X., Lv, S., Rodriguez, R.J. & Trivedi, K. (2018) Survivability model for security and dependability analysis of a vulnerable critical system. *Proceedings - International Conference on Computer Communications and Networks, ICCCN*. [Online] 2018-July (August). Available from: doi:10.1109/ICCCN.2018.8487446.

Chang, Y.H., Yeh, C.H. & Chang, Y.W. (2013) A new method selection approach for fuzzy group multicriteria decision making. *Applied Soft Computing Journal*. [Online] Available from: doi:10.1016/j.asoc.2012.12.009.

Changa, X., Zhang, Z., Li, X. & Trivedi, K.S. (2016) *Model-based Survivability Analysis of a Virtualized System*. [Online] (November). Available from: doi:10.1109/LCN.2016.104.

Chapman, K.W., Lawless, H.T. & Boor, K.J. (2001) Quantitative descriptive analysis and principal component analysis for sensory characterization of ultrapasteurized milk. *Journal of dairy science*. [Online] 84 (1), 12–20. Available from: doi:10.3168/jds.S0022-0302(01)74446-3.

Chardonnens, T., Cudre-Mauroux, P., Grund, M. & Perroud, B. (2013) Big data analytics on high Velocity streams: A case study. In: *Proceedings - 2013 IEEE International Conference on Big Data, Big Data 2013*. [Online]. 2013 p. Available from: doi:10.1109/BigData.2013.6691653.

Checkland, P. (1981) *Systems Thinking, System practice*.

Chen, Y., Paxson, V. & Katz, R.H. (2010) What's new about cloud computing security. *University of California, Berkeley Report No.UCB/EECS-2010-5 January*. 20 (2010), 2010–2015.

Cheng, A.C., Chen, C.J. & Chen, C.Y. (2008) A fuzzy multiple criteria comparison of technology forecasting methods for predicting the new materials development. *Technological Forecasting and Social Change*. [Online] Available from: doi:10.1016/j.techfore.2006.08.002.

Cheng, L., Li, Y., Li, W., Holm, E., et al. (2013) Understanding the violation of IS security policy in organizations: An integrated model based on social control and deterrence theory. *Computers & Security*. 39, 447–459.

Cheng, R., He, C., Jin, Y. & Yao, X. (2018) Model-based evolutionary algorithms: a short survey. *Complex & Intelligent Systems*. [Online] Available from: doi:10.1007/s40747-018-0080-1.

Chiacchio, F., Pennisi, M., Russo, G., Motta, S., et al. (2014) Agent-based modeling of the immune system: NetLogo, a promising framework. *BioMed research international*. [Online] 2014, 907171. Available from: doi:10.1155/2014/907171 [doi].

Chow, R., Golle, P., Jakobsson, M., Shi, E., et al. (2009a) Controlling data in the cloud: outsourcing computation without outsourcing control. In: *Proceedings of the 2009 ACM workshop on Cloud computing security*. 2009 ACM. pp. 85–90.

Chow, R., Golle, P., Jakobsson, M., Shi, E., et al. (2009b) Controlling data in the cloud. *Proceedings of the 2009 ACM workshop on Cloud computing security - CCSW '09*. [Online] 85. Available from: doi:10.1145/1655008.1655020.

Chowdhury, M., Rahman, M.R. & Boutaba, R. (2012) ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Transactions on Networking*. [Online] Available from: doi:10.1109/TNET.2011.2159308.

Chraibi, M., Harroud, H. & Maach, A. (2013) Classification of Security Issues and Solutions in Cloud Environments. In: *Proceedings of International Conference on Information Integration and Web-based Applications & Services*. 2013 ACM. p. 560.

CipherCloud (2016) *CIPHERCLOUD TRUST PLATFORM CLOUD SECURITY BROKER*.

Citrix (2015) *Defend Web Properties from Modern Threats with Citrix NetScaler - White paper*. 10.

Cloud Security Alliance (2013a) The Notorious Nine. Cloud Computing Top Threats in 2013. *Security*. [Online] (February), 1–14. Available from:

doi:http://www.cloudsecurityalliance.org/topthreats.

Cloud Security Alliance (2013b) *The Notorious Nine*. 2013.

CloudLock (2015a) *Cisco Cloudlock is a Cloud Cybersecurity Platform*. 2015.

CloudLock (2015b) *Cisco Cloudlock is a Cloud Cybersecurity Platform*. [Online]. 2015. Available from: https://www.cloudlock.com/platform/ [Accessed: 10 December 2018].

Coiro, J. (2014) *Handbook of Research on New Literacies*. [Online]. Available from: doi:10.4324/9781410618894.

Colomer, M.A., Margalida, A., Sanuy, D. & Perez-Jimenez, M.J. (2011) A bio-inspired computing model as a new tool for modeling ecosystems: the avian scavengers as a case study. *Ecological Modelling*. 222 (1), 33–47.

Commission, E. and H.R. (2016) *Article 8: Respect for your private and family life*. [Online]. 2016. Available from: https://www.equalityhumanrights.com/en/human-rights-act/article-8-respect-your-private-and-family-life [Accessed: 1 January 2016].

Coppolino, L., D'Antonio, S., Mazzeo, G. & Romano, L. (2017) Cloud security: Emerging threats and current solutions. *Computers and Electrical Engineering*. [Online] 59, 126–140. Available from: doi:10.1016/j.compeleceng.2016.03.004.

Coty, S. (2014) *Shellshock vulnerability: impact, analysis, and protection*. [Online]. 2014. Available from: https://blog.alertlogic.com/blog/shellshock-vulnerability-impact,-analysis,-and-protection/ [Accessed: 22 December 2019].

Courbin, N., Loveridge, A.J. & Macdonald, D.W. (2015) Reactive responses of zebras to lion encounters shape their predator - prey space game at large scale. *Oikos*.

Couto, R. de S., Secci, S., Campista, M.E.M. & Costa, L.H.M.K. (2016) Reliability and survivability analysis of data center network topologies. *Journal of Network and System Management*. [Online] 24 (2), 346–392. Available from: doi:10.1007/s10922-015-9354-8.

Cox Louis Anthony (Tony), J. (2009) Game Theory and Risk Analysis. *Risk Analysis*. [Online] Available from: doi:10.1111/j.1539-6924.2009.01247.x.

Craig, S., Harrison, D., Cripps, A. & Knott, D. (2008) BioTRIZ Suggests Radiative Cooling of Buildings Can Be Done Passively by Changing the Structure of Roof Insulation to Let Longwave Infrared Pass. *Journal of Bionic Engineering*. [Online] 5 (1), 55–66. Available from: doi:10.1016/S1672-6529(08)60007-4.

Creativity, S. (2019a) *TRIZ40*. [Online]. 2019. Available from: http://www.triz40.com/TRIZ_GB.php [Accessed: 10 March 2019].

Creativity, S. (2019b) *TRIZ40*. 2019.

Cser, A. (2016) *The Eight Providers That Matter Most and How They Stack Up*.

Cybenko, G., Jajodia, S., Wellman, M.P. & Liu, P. (2014) *Adversarial and Uncertain Reasoning for Adaptive Cyber Defense: Building the Scientific Foundation*. In: Information Systems Security. Springer. pp. 1–8.

Darst, C.R., Menéndez-Guerrero, P.A., Coloma, L.A. & Cannatella, D.C. (2005) Evolution of Dietary Specialization and Chemical Defense in Poison Frogs (Dendrobatidae): A Comparative Analysis. *The American Naturalist*. [Online] 165 (1), 56–69. Available from: doi:10.1086/426599.

David, M. & Kris, D. (2016) *How to Leverage Cognitive Technology to Think Like a Security Expert*. [Online]. 2016. Available from: https://securityintelligence.com/events/leverage-ibm-cognitive-technology-ifa/ [Accessed: 21 December 2017].

Demchenko, Y., Ngo, C., Laat, C. De, Wlodarczyk, T.W., et al. (2011) Security infrastructure for on-demand provisioned cloud infrastructure services. In: *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*. 2011 IEEE. pp. 255–263.

Díaz, M., Martín, C. & Rubio, B. (2016) State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing. *Journal of Network and Computer Applications*. [Online]. Available from: doi:10.1016/j.jnca.2016.01.010.

Djenouri, D., Khelladi, L. & Badache, N. (2005) A survey of security issues in mobile ad hoc networks. *IEEE communications surveys*. 7 (4), 2–28.

Domb, E., Miller, J., MacGran, E. & Slocum, M. (2011a) *Explanation of the 39 Parameters of the Contradiction Table (Matrix)*. [Online] 1–12. Available from: http://triz40.com/aff_Principles.htm.

Domb, E., Miller, J., MacGran, E. & Slocum, M. (2011b) *Explanation of the 39 Parameters of the Contradiction Table (Matrix)*. 1–12.

Dong, W.M. & Wong, F.S. (1987) Fuzzy weighted averages and implementation of the extension principle. *Fuzzy Sets and Systems*. [Online] Available from: doi:10.1016/0165-0114(87)90163-1.

Dorey, P.G. & Leite, A. (2011) Commentary: Cloud computing-A security problem or solution? *information security technical report*. 16 (3), 89–96.

Dressler, F. & Akan, O.B. (2010) A survey on bio-inspired networking. *Computer Networks*. 54 (6), 881–900.

Eiger, M.I., Luss, H. & Shallcross, D.F. (2011) Network restoration under a single link or node failure using Preconfigured Virtual Cycles. *Telecommunication Systems*. [Online] Available from: doi:10.1007/s11235-009-9273-7.

Eiger, M.I., Luss, H. & Shallcross, D.F. (2012) Network restoration under dual failures using path-protecting preconfigured cycles. *Telecommunication Systems*. [Online] Available from: doi:10.1007/s11235-010-9374-3.

Eisner, T., Eisner, M., Rossini, C., Iyengar, V.K., et al. (2000) Chemical defense against predation in an insect egg. *Proceedings of the National Academy of Sciences of the United States of America*. [Online] Available from: doi:10.1073/pnas.030532797.

Ellison, R., Linger, R., Lipson, H., Mead, N., et al. (2002) Foundations for survivable systems engineering. *The Journal of Defense Software Engineering*. 10–15.

Ellison, R.J., Ellison, R.J., Fisher, D. a, Fisher, D. a, et al. (1997a) Survivable Network Systems: An Emerging Discipline. *1999 IEEE International Performance, Computing and Communications Conference (Cat. No.99CH36305)*. [Online] (May), 469–475. Available from: doi:CMU/SEI-97-TR-013.

Ellison, R.J., Fisher, D.A., Linger, R.C., Lispson, H.F., et al. (1997b) Survivable Network Systems: An Emerging Discipline. *1999 IEEE International Performance, Computing and Communications Conference (Cat. No.99CH36305)*.

Eoin, C., Taylor, D., John, F., German, L., et al. (2018) *McAfee Labs 2019 Threats Predictions Report*. [Online]. 2018. Available from: https://www.mcafee.com/blogs/other-blogs/mcafee-labs/mcafee-labs-2019-threats-predictions/ [Accessed: 22 December 2019].

Fackler, P.L. & Haight, R.G. (2014a) Monitoring as a partially observable decision problem. *Resource and Energy Economics*. [Online] 37, 226–241. Available from: doi:10.1016/j.reseneeco.2013.12.005.

Fackler, P.L. & Haight, R.G. (2014b) Monitoring as a partially observable decision problem. *Resource and Energy Economics*. [Online] 37, 226–241. Available from: doi:10.1016/j.reseneeco.2013.12.005.

Fan, G., Yu, H., Chen, L. & Liu, D. (2013) A game theoretic method to model and evaluate attack-defense strategy in cloud computing. In: *Proceedings - IEEE 10th International Conference on Services Computing, SCC 2013*. [Online]. 2013 p. Available from: doi:10.1109/SCC.2013.110.

Fang, X., Koceja, N., Zhan, J., Dozier, G., et al. (2012) An artificial immune system for phishing detection. In: *Evolutionary Computation (CEC), 2012 IEEE Congress on*. 2012 IEEE. pp. 1–7.

Farooq, M.U., Waseem, M., Khairi, A. & Mazhar, S. (2015) A Critical Analysis on the Security Concerns of Internet of Things (IoT). *International Journal of Computer Applications*. 111 (7).

Fenton, N. & Wang, W. (2006) Risk and confidence analysis for fuzzy multicriteria decision making. *Knowledge-Based Systems*. [Online] 19 (6), 430–437. Available from: doi:10.1016/j.knosys.2006.03.002.

Finstadsveen, J. & Begnum, K. (2011) What a webserver can learn from a zebra and what we learned in the process. In: *Proceedings of the 5th ACM Symposium on Computer Human Interaction for Management of Information Technology*. 2011 ACM. p. 5.

Firdhous, M., Ghazali, O. & Hassan, S. (2012) Trust Management in Cloud Computing: A Critical Review. *International Journal on Advances in ICT for Emerging Regions (ICTer)*. [Online] 4 (02), 24–36. Available from: doi:10.4038/icter.v4i2.4674.

Fitzgibbon, C.D. (1990a) Anti-predator strategies of immature Thomson's gazelles: hiding and the prone response. *Animal Behaviour*. 40 (5), 846–855.

Fitzgibbon, C.D. (1990b) Why do hunting cheetahs prefer male gazelles? *Animal Behaviour*. 40 (5), 837–845.

Floratou, A., Potti, N. & Patel, J.M. (2014) Online replica placement in Cloud Environments. In: *ACM Symposium on Cloud Computing*. 2014 ACM Press. p.

Foote, R. (2007) Mathematics and complex systems. *Science*. [Online]. 318 (5849) pp.410–412. Available from: doi:10.1126/science.1141754.

Fryxell, J.M., Mosser, A., Sinclair, A.R.E. & Packer, C. (2007) Group formation stabilizes predator-prey dynamics. *Nature*. [Online] Available from: doi:10.1038/nature06177.

Fu, K., Moreno, D., Yang, M. & Wood, K.L. (2014) Bio-Inspired Design: An Overview Investigating Open Questions From the Broader Field of Design-by-Analogy. *Journal of Mechanical Design*. [Online] Available from: doi:10.1115/1.4028289.

Furuncu, E. & Sogukpinar, I. (2015) Scalable risk assessment method for cloud computing using game theory (CCRAM). *Computer Standards & Interfaces*. [Online] 38, 44–50. Available from: doi:10.1016/j.csi.2014.08.007.

G, S. & S, M. (2013) Securing Software as a Service Model of Cloud Computing: Issues and Solutions. *International Journal on Cloud Computing: Services and Architecture*. [Online] Available from: doi:10.5121/ijccsa.2013.3401.

García, S., Ramírez-Gallego, S., Luengo, J., Benítez, J.M., et al. (2016) Big data preprocessing: methods and prospects. *Big Data Analytics*. [Online] Available from: doi:10.1186/s41044-016-0014-0.

Gebhardt, C. & Tomlinson, A. (2010) *Challenges for inter virtual machine communication*. [Online] (September), 0–17. Available from: http://www.rhul.ac.uk/mathematics/techreports.

Glier, M. & McAdams, D. (2011) Concepts in biomimetic design: methods and tools to incorporate into a biomimetic design course. *ASME 2011*. [Online] Available from: http://proceedings.asmedigitalcollection.asme.org/proceeding.aspx?articleid=1641340 [Accessed: 15 June 2017].

Gonzalez, F.A. & Dasgupta, D. (2003) Anomaly Detection Using Real-Valued Negative Selection. *Genetic Programming and Evolvable Machines*. [Online] 4 (4), 383–403. Available from: doi:doi:10.1023/A:1026195112518.

Gonzalez, N., Miers, C., Redigolo, F., Simplicio, M., et al. (2012) A quantitative analysis of current security concerns and solutions for cloud computing. *Journal of Cloud Computing*. 1 (1), 1–18.

Gopal, R.D. & Sanders, G.L. (2000) Global software piracy: You can't get blood out of a turnip. *Communications of the ACM*. 43 (9), 82–89.

Gorman, S.P., Kulkarni, R.G., Schintler, L.A. & Stough, R.R. (2004) A predator prey approach to the network structure of cyberspace. In: *Proceedings of the winter international synposium on Information and communication technologies*. 2004 Trinity College Dublin. pp. 1–6.

Grand, T.C. & Dill, L.M. (1999) The effect of group size on the foraging behaviour of juvenile coho salmon: Reduction of predation risk or increased competition? *Animal Behaviour*. [Online] Available from: doi:10.1006/anbe.1999.1174.

Grant, C. & Osanloo, A. (2014) UNDERSTANDING, SELECTING, AND INTEGRATING A THEORETICAL FRAMEWORK IN DISSERTATION RESEARCH: CREATING THE BLUEPRINT FOR YOUR "HOUSE". *Administrative Issues Journal Education Practice and Research*. [Online] Available from: doi:10.5929/2014.4.2.9.

Grant, T. (2017) Speeding up parliamentary decision making for cyber counter-attack. In: *Proceedings of the 12th International Conference on Cyber Warfare and Security, ICCWS 2017*. 2017 p.

Gras, R., Devaurs, D., Wozniak, A. & Aspinall, A. (2009) An Individual-Based Evolving Predator-Prey Ecosystem Simulation Using a Fuzzy Cognitive Map as the Behavior Model. *Artificial Life*. [Online] 15 (4), 423–463. Available from: doi:10.1162/artl.2009.Gras.012.

Gregory, D. (2011) From a View to a Kill: Drones and Late Modern War. *Theory, Culture & Society*. [Online] 28 (7–8), 188–215. Available from: doi:10.1177/0263276411423027.

Grimes, R. (2001) *Malicious mobile code: Virus protection for Windows*. ' O'Reilly Media, Inc.'

Grobauer, B., Walloschek, T. & Stocker, E. (2011) Understanding cloud computing vulnerabilities. *Security & privacy, IEEE*. 9 (2), 50–57.

Gu, F., Shaban, K., Ghani, N., Khan, S., et al. (2015) Survivable cloud network mapping for disaster recovery support. *IEEE Transactions on Computers*. [Online] 64 (8), 2353–2366. Available from: doi:10.1109/TC.2014.2360542.

Guh, Y.Y., Hon, C.C., Wang, K.M. & Lee, E.S. (1996) Fuzzy weighted average: A max-min paired elimination method. *Computers and Mathematics with Applications*. [Online] Available from: doi:10.1016/0898-1221(96)00171-X.

Guillaume, S., Charnomordic, B. & Lablée, J.-L. (2002) *FisPro Fuzzy inference system design and optimization*. [Online]. 2002. Available from: https://www.fispro.org/en/contributors/ [Accessed: 10 January 2019].

Gulla, J. (2011a) *Gaining efficiency and business value through better management of your IT infrastructure*. 2011.

Gulla, J. (2011b) *Gaining efficiency and business value through better management of your IT infrastructure*. [Online]. 2011. Available from: http://www-01.ibm.com/common/ssi/cgibin/ssialias?infotype=SA&subtype=WH&htmlfid=SSW03005USEN [Accessed: 29 August 2015].

Gupta, B., Agrawal, D.P. and Yamaguchi, S. eds. (2016) *Handbook of research on modern cryptographic solutions for computer and cyber security*. IGI Global.

Gupta, A. & DuVarney, D.C. (2004) Using predators to combat worms and viruses: A simulation-based study. In: *Computer Security Applications Conference, 2004. 20th Annual*. 2004 IEEE. pp. 116–125.

Gupta, B.B., Arachchilage, N.A.G. & Psannis, K.E. (2017) Defending against phishing attacks: taxonomy of methods, current issues and future directions. *Telecommunication Systems*. [Online] Available from: doi:10.1007/s11235-017-0334-z.

Haas, C. (2014) *Incentives and two-sided matching: Engineering coordination mechanisms for social clouds*. [Online]. Available from: doi:10.5445/KSP/1000041861.

Habib, M.F., Tornatore, M., Dikbiyik, F. & Mukherjee, B. (2013) Disaster survivability in optical communication networks. *Computer Communications*. [Online] 36 (6), 630–644. Available from: doi:10.1016/j.comcom.2013.01.004.

Hansman, S. & Hunt, R. (2005) A taxonomy of network and computer attacks. *Computers and Security*. [Online] Available from: doi:10.1016/j.cose.2004.06.011.

Hariri, S., Eltoweissy, M. & Al-Nashif, Y. (2011) Biorac: biologically inspired resilient autonomic cloud. In: *Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research*. 2011 ACM. p. 80.

Harknett, R.J. & Stever, J.A. (2011) The New Policy World of Cybersecurity. *Public Administration Review*. [Online] 71 (3), 455–460. Available from: doi:10.1111/j.1540-6210.2011.02366.x.

Hashizume, K., Yoshioka, N. & Fernandez, E.B. (2013) Three Misuse Patterns for Cloud Computing. *Security Engineering for Cloud Computing: Approaches and Tools*. [Online] 36–53. Available from: doi:10.4018/978-1-4666-2125-1.ch003.

Havens, T. (2019) Netflix. In: *From Networks to Netflix*. [Online]. p. Available from: doi:10.4324/9781315658643-30.

Hegselmann, R. & Flache, A. (1998) Understanding Complex Social Dynamics: A Plea For Cellular Automata Based Modelling. *Journal of Artificial Societies and Social Simulation*. 1(13), 30.

Heimbigner, D. (1990) Proscription versus prescription in process-centered environments. In: *Software Process Workshop, 1990.'Support for the Software Process'., Proceedings of the 6th International*. 1990 IEEE. pp. 99–102.

Heinrich, B. (1979) Foraging strategies of caterpillars. *Oecologia*. 42 (3), 325–337.

Heiser, G., Goldwasser, S., Micali, S. & Franson, J.D. (2015) Communications acm. *ACM Computing Surveys*. [Online] 58 (1), 1–5. Available from: doi:10.1017/CBO9781107415324.004.

Helms, M., Vattam, S. & Goel, A. (2009) Biologically inspired design: process and products. *Design studies*. [Online] Available from: http://www.sciencedirect.com/science/article/pii/S0142694X09000283 [Accessed: 15 June 2017].

Henderson, R. (2006) Design, simulation, and testing of a novel hydraulic power take-off system for the Pelamis wave energy converter. In: *Renewable Energy*. [Online]. 2006 p. Available from: doi:10.1016/j.renene.2005.08.021.

Henning, M. (2007) API design matters. *Queue*. 5 (4), 24–36.

Himma, K.E. (2007) *The Ethics of 'Hacking Back': Active Response to Computer Intrusion*. In: Anderson Tim (ed.). Internet Security: Hacking, Counterhaking, and Society. 1st edition. Jones and Barlett. pp. 99–100.

Hirai, Y. (2017) Defining the ethereum virtual machine for interactive theorem provers. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. [Online]. 2017 p. Available from: doi:10.1007/978-3-319-70278-0_33.

Hogan, K. & Maglienti, M. (2001) Comparing the epistemological underpinnings of students' and scientists' reasoning about conclusions. *Journal of Research in Science Teaching*. [Online] 38 (6), 663–687. Available from: doi:10.1002/tea.1025.

Hone, D.W.E. & Benton, M.J. (2005) The evolution of large size: how does Cope's Rule work? *Trends in Ecology & Evolution*. 20 (1), 4–6.

Hordijk, W. (2005) An Overview of Biologically Inspired Computing in Information Security. In: *Proceedings of the National Conference on Information Security, Coimbatore, India*. 2005 pp. 1–14.

Horikoshi, H., Nakagawa, H., Tahara, Y. & Ohsuga, A. (2012) Dynamic reconfiguration in self-adaptive systems considering non-functional properties. *Proceedings of the 27th Annual ACM Symposium on Applied Computing - SAC '12*. [Online] (i), 1144. Available from: doi:10.1145/2245276.2231956.

Houidi, I., Louati, W., Ben Ameur, W. & Zeghlache, D. (2011) Virtual network provisioning across multiple substrate networks. *Computer Networks*. [Online] Available from: doi:10.1016/j.comnet.2010.12.011.

Howard, J.D. (1998) *An Analysis of Security Incidents on the Internet 1989-1995*.

Howard, J.D. & Longstaff, T.A. (1998) A common language for computer security incidents. *Sandia National Laboratories*.

Hsieh, H.-N., Chen, J.-F. & Do, Q. (2015) Applying TRIZ and Fuzzy AHP Based on Lean Production to Develop an Innovative Design of a New Shape for Machine Tools. *Information*. [Online] Available from: doi:10.3390/info6010089.

Hsu, H., Tsai, B. & Chen, K. (2013) A TRIZ Approach to Business Management Formulation - A Case of HRMS Industry. In: *Proceedings of the International MultiConference of Engineers and Computer Scientists*. 2013 p.

Hsu, W. & Liu, B. (2000) Conceptual design: issues and challenges. *Computer-Aided Design*. [Online] Available from: doi:10.1016/s0010-4485(00)00074-9.

Huang, D. & Wu, H. (2018) Mobile Cloud Computing Taxonomy. In: *Mobile Cloud Computing*. [Online]. p. Available from: doi:10.3109/13813458509070424.

Huang, W., Ganjali, A., Kim, B.H., Oh, S., et al. (2015) The State of Public Infrastructure-as-a-Service Cloud Security. *ACM Computing Surveys (CSUR)*. 47 (4), 68.

Hulu (2016) ABOUT HULU. *Hulu Press.*

Hummaida, A.R., Paton, N.W. & Sakellariou, R. (2016) Adaptation in cloud resource configuration: a survey. *Journal of Cloud Computing.* [Online]. Available from: doi:10.1186/s13677-016-0057-9.

Hussain, S.A., Fatima, M., Saeed, A., Raza, I., et al. (2017) Multilevel classification of security concerns in cloud computing. *Applied Computing and Informatics.* [Online] 13 (1), 57–65. Available from: doi:10.1016/j.aci.2016.03.001.

Huston, M., DeAngelis, D. & Post, W. (1988) New computer models unify ecological theory. *Bioscience.* 38 (10), 682–691.

Hutchins, E.M., Cloppert, M.J. & Amin, R.M. (2011) Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains. *6th Annual International Conference on Information Warfare and Security.* [Online] (July 2005), 1–14. Available from: http://papers.rohanamin.com/wp-content/uploads/papers.rohanamin.com/2011/08/iciw2011.pdf%5Cnhttp://www.lockheedmartin.com/content/dam/lockheed/data/corporate/documents/LM-White-Paper-Intel-Driven-Defense.pdf.

Huynh, V., Ryoke, M. & Ho, T.B. (2007) *Decision making under uncertainty with fuzzy targets.* [Online] (May 2014). Available from: doi:10.1007/s10700-007-9011-0.

Hwang, K. & Li, D. (2010) Trusted cloud computing with secure resources and data coloring. *IEEE Internet Computing.* [Online] Available from: doi:10.1109/MIC.2010.86.

II, J.C.R. & Al-Hamdani, W. (2011) Who can you trust in the cloud?: a review of security issues within cloud computing. In: *Proceedings of the 2011 Information Security Curriculum Development Conference.* 2011 ACM. pp. 15–19.

Iivari, J. (2007) A Paradigmatic Analysis of Information Systems As a Design Science A Paradigmatic Analysis of Information Systems As a Design Science. *Scandanavian Journal of Information Systems.* [Online] 19 (2), 5. Available from: doi:10.1.1.218.2636.

Ilevbare, I.M., Probert, D. & Phaal, R. (2013) A review of TRIZ, and its benefits and challenges in practice. *Technovation.* [Online]. 33 (2–3) pp.30–37. Available from: doi:10.1016/j.technovation.2012.11.003.

Isasi, P. & Hernandez, J.C. (2004) Introduction to the applications of evolutionary computation in computer security and cryptography. *Computational Intelligence.* 20 (3), 445–449.

Isbell, L.A. (1994) Predation on primates: ecological patterns and evolutionary consequences. *Evolutionary Anthropology: Issues, News, and Reviews*. 3 (2), 61–71.

Jabir, R.M., Khanji, S.I.R., Ahmad, L.A., Alfandi, O., et al. (2016) Analysis of cloud computing attacks and countermeasures. *International Conference on Advanced Communication Technology, ICACT*. [Online] 2016-March, 117–123. Available from: doi:10.1109/ICACT.2016.7423296.

Jamal, M.H., Qadeer, A., Mahmood, W., Waheed, A., et al. (2009) *Virtual Machine Scalability on Multi-Core Processors Based Servers for Cloud Computing Workloads*. [Online] Available from: doi:10.1109/NAS.2009.20.

Jamil, D. & Zaki, H. (2011) Security Issues in Cloud Computing and Countermeasures. *International Journal of Engineering Science and Technology*. [Online] 3 (4), 2672–2676. Available from: doi:10.1109/GSIS.2011.6043978.

Jansen, W. (2011) Cloud hooks: Security and privacy issues in cloud computing. In: *System Sciences (HICSS), 2011 44th Hawaii International Conference on*. 2011 IEEE. pp. 1–10.

Janson, C.H. & Goldsmith, M.L. (1995) Predicting group size in primates: foraging costs and predation risks. *Behavioral Ecology*. 6 (3), 326–336.

Jeff, B. (2015) *AWS Config Rules – Dynamic Compliance Checking for Cloud Resources*. [Online]. 2015. AWS News Blog. Available from: https://aws.amazon.com/blogs/aws/aws-config-rules-dynamic-compliance-checking-for-cloud-resources/ [Accessed: 20 December 2017].

Jefferson, A., Bortolotti, L. & Kuzmanovic, B. (2017) What is unrealistic optimism? *Consciousness and Cognition*. [Online] 50, 3–11. Available from: doi:10.1016/j.concog.2016.10.005.

Ji, Z. & Dasgupta, D. (2009) V-detector: An efficient negative selection algorithm with probably adequate detector coverage. *Information Sciences*. 179 (10), 1390–1406.

Jiang, X., Xu, D. & Wang, Y.M. (2006) Collapsar: A VM-based honeyfarm and reverse honeyfarm architecture for network attack capture and detention. *Journal of Parallel and Distributed Computing*. [Online] 66 (9), 1165–1180. Available from: doi:10.1016/j.jpdc.2006.04.012.

Jiao, W. & Sun, Y. (2016) Self-adaptation of multi-agent systems in dynamic environments based on experience exchanges. *Journal of Systems and Software*. [Online] 122, 165–179. Available from: doi:10.1016/j.jss.2016.09.025.

Jing, X., Bi, Y. & Deng, H. (2016) An innovative two-stage fuzzy kNN-DST classifier for unknown intrusion detection. *International Arab Journal of Information Technology*.

Jinyin, C. & Dongyong, Y. (2013) Data security strategy based on artificial immune algorithm for cloud computing. *Appl.Math*. 7 (1L), 149–153.

Johns, M. (2011) Code-injection Vulnerabilities in Web Applications-Exemplified at Cross-site Scripting. *it-Information Technology Methoden und innovative Anwendungen der Informatik und Informationstechnik*. 53 (5), 256–260.

Jormakka, J. & Mölsä, J.V.E. (2005) Modelling Information Warfare as a Game. *Journal of Information Warfare*.

Joshi, B. & Joshi, B.K. (2012) Securing cloud computing environment against DDoS attacks. In: *Computer Communication and Informatics (ICCCI), 2012 International Conference on*. 2012 IEEE. pp. 1–5.

Joshi, B., Vijayan, a. S. & Joshi, B.K. (2012) Securing cloud computing environment against DDoS attacks. *2012 International Conference on Computer Communication and Informatics*. [Online] 1–5. Available from: doi:10.1109/ICCCI.2012.6158817.

Jr, E.D.B., Jr, D.R.F. & III, E.D.B. (1991) Predator avoidance and antipredator mechanisms: distinct pathways to survival. *Ethology Ecology & Evolution*. 3 (1), 73–77.

Kamhoua, C.A., Kwiat, L., Kwiat, K.A., Park, J.S., et al. (2014) Game theoretic modeling of security and interdependency in a public cloud. In: *IEEE International Conference on Cloud Computing, CLOUD*. [Online]. 2014 p. Available from: doi:10.1109/CLOUD.2014.75.

Kao, C. & Liu, S.-T. (2002) Fractional programming approach to fuzzy weighted average. *Fuzzy Sets and Systems*. [Online] Available from: doi:10.1016/s0165-0114(99)00137-2.

Kasravi, K. & Fellow, H. (2010) Applications of TRIZ to IT: Cases and Lessons Learned. *trizjournal*.

Kauffman, S.A. (1993) *THE ORIGINS OF ORDER. SELF-ORGANIZATION AND SELECTION IN EVOLUTION*.

Keele, S. (2007) *Guidelines for performing systematic literature reviews in software engineering*. In: Technical report, Ver. 2.3 EBSE Technical Report. EBSE. p.

Kephart, J.O. (1994) A biologically inspired immune system for computers. In: *Artificial Life IV: proceedings of the fourth international workshop on the synthesis and simulation of living systems*. 1994 pp. 130–139.

Khalil, I., Khreishah, A. & Azeem, M. (2014) Cloud Computing Security: A Survey. *Computers*. [Online] 3 (1), 1–35. Available from: doi:10.3390/computers3010001.

Khan, M.M.A., Shahriar, N., Ahmed, R. & Boutaba, R. (2015) SiMPLE: Survivability in multi-path link embedding. In: *Proceedings of the 11th International Conference on Network and Service Management, CNSM 2015*. [Online]. 2015 p. Available from:

doi:10.1109/CNSM.2015.7367361.

Kim, J., Bentley, P.J., Aickelin, U., Greensmith, J., et al. (2007) Immune system approaches to intrusion detection - A review. *Natural Computing*. [Online]. 6 (4) pp.413–466. Available from: doi:10.1007/s11047-006-9026-4.

Kingsolver, J.G. & Pfennig, D.W. (2004) Individual-level selection as a cause of cope's rule of phyletic size increase. *Evolution*. [Online] Available from: doi:10.1111/j.0014-3820.2004.tb01740.x.

Kloeden, P.E. & Pötzsche, C. (2010) Dynamics of modified predator-prey models. In: *International Journal of Bifurcation and Chaos*. [Online]. 2010 pp. 2657–2669. Available from: doi:10.1142/S0218127410027271.

Klugl, F. & Bazzan, A.L.C. (2012) Agent-Based Modeling and Simulation. *AI Magazine*. [Online] 33 (3), 29–40. Available from: doi:0738-4602.

Knemeyer, A.M., Zinn, W. & Eroglu, C. (2009) Proactive planning for catastrophic events in supply chains. *Journal of Operations Management*. [Online] Available from: doi:10.1016/j.jom.2008.06.002.

Knight, C. & Munro, M. (1999) Comprehension with[in] virtual environment visualisations. In: *Proceedings - 7th International Workshop on Program Comprehension, IWPC 1999*. [Online]. 1999 p. Available from: doi:10.1109/WPC.1999.777733.

Ko, R.K.L., Jagadpramana, P., Mowbray, M., Pearson, S., et al. (2011) TrustCloud: A framework for accountability and trust in cloud computing. In: *Services (SERVICES), 2011 IEEE World Congress on*. 2011 IEEE. pp. 584–588.

Koch, C. (2003) Efficient processing of expressive node-selecting queries on XML data in secondary storage: A tree automata-based approach. In: *Proceedings of the 29th international conference on Very large data bases-Volume 29*. [Online]. 2003 pp. 249–260. Available from: http://portal.acm.org/citation.cfm?id=1315474.

Koga, T., Backwell, P.R.Y., Christy, J.H., Murai, M., et al. (2001) Male-biased predation of a fiddler crab. *Animal Behaviour*. 62 (2), 201–207.

Kreidl, O.P. & Frazier, T.M. (2004a) Feedback control applied to survivability: A host-based autonomic defense system. *IEEE Transactions on Reliability*. [Online] 53 (1), 148–166. Available from: doi:10.1109/TR.2004.824833.

Kreidl, O.P. & Frazier, T.M. (2004b) Feedback control applied to survivability: A host-based autonomic defense system. *IEEE Transactions on Reliability*. [Online] Available from: doi:10.1109/TR.2004.824833.

Krsul, I.V. (2014) Software Vulnerability Analysis. *Uma ética para quantos?*. [Online]

Available from: doi:10.1007/s13398-014-0173-7.2.

Kushida, K.E., Murray, J. & Zysman, J. (2012) The gathering storm: Analyzing the cloud computing ecosystem and implications for public policy. *Communications & Strategies*. (85), 63–85.

Kwiatkowska, M., Parker, D. & Qu, H. (2011) Incremental quantitative verification for Markov decision processes. In: *Proceedings of the International Conference on Dependable Systems and Networks*. [Online]. 2011 p. Available from: doi:10.1109/DSN.2011.5958249.

Labuda, I. (2015) Possibilities of applying TRIZ methodology elements (the 40 Inventive Principles) in the process of architectural design. In: *Procedia Engineering*. [Online]. 2015 pp. 476–499. Available from: doi:10.1016/j.proeng.2015.12.443.

Landau, I.D. (1999) From robust control to adaptive control. *Control Engineering Practice*. [Online] 7, 1113–1124. Available from: doi:10.1016/S0967-0661(99)00076-3.

León, M., Rodriguez, C., García, M.M., Bello, R., et al. (2010) Fuzzy cognitive maps for modeling complex systems. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. [Online]. 2010 p. Available from: doi:10.1007/978-3-642-16761-4_15.

Leone, W. (2015) IBM Cloud. *Agenda*. 7793.

Lertpalangsunti, N., Chan, C.W., Mason, R. & Tontiwachwuthikul, P. (1999) Toolset for construction of hybrid intelligent forecasting systems: Application for water demand prediction. *Artificial Intelligence in Engineering*. [Online] Available from: doi:10.1016/S0954-1810(98)00008-9.

Levitin, G., Hausken, K., Taboada, H.A. & Coit, D.W. (2012) Data survivability vs. security in information systems. *Reliability Engineering and System Safety*. [Online] 100, 19–27. Available from: doi:10.1016/j.ress.2011.12.015.

Levy, J. (2018) *Sophoslabs 2019 Threat Report*. [Online] Available from: https://www.sophos.com/en-us/medialibrary/PDFs/technical-papers/sophoslabs-2019-threat-report.pdf.

Li, M., Zhang, Y., Bai, K., Zang, W., et al. (2012) Improving Cloud Survivability through Dependency based Virtual Machine Placement. *Proceedings of the International Conference on Security and Cryptography (SECRYPT)*. (Dom 0), 321–326.

Liang, X. & Fengbin, Z. (2013) Detector optimization algorithm with co-evolution in immunity-based intrusion detection system. In: *Measurement, Information and Control (ICMIC), 2013 International Conference on*. 2013 IEEE. pp. 620–623.

Liao, D., Sun, G., Anand, V. & Yu, H. (2014) Survivable provisioning for multicast service oriented virtual network requests in cloud-based data centers. *Optical Switching and Networking*. [Online] 14 (PART 3), 260–273. Available from: doi:10.1016/j.osn.2014.05.019.

Lindqvist, U. & Jonsson, E. (1997) *How to Systematically Classify Computer Security Intrusions*. 154–163.

Liou, T.S. & Wang, M.J.J. (1992) Fuzzy weighted average: An improved algorithm. *Fuzzy Sets and Systems*. [Online] Available from: doi:10.1016/0165-0114(92)90282-9.

Lipson, H.F. & Fisher, D. a. (1999) Survivability - a New Technical and Business Perspective on Security. *Proceedings of the 1999 workshop on New security paradigms - NSPW '99*. [Online] 33–39. Available from: doi:10.1145/335169.335187.

Lipson, H.F. & Fisher, D.A. (2004) *Survivability---a new technical and business perspective on security*. In: [Online]. 2004 p. Available from: doi:10.1145/335169.335187.

Lischka, J. & Karl, H. (2009) A virtual network mapping algorithm based on subgraph isomorphism detection. In: *SIGCOMM 2009 - Proceedings of the 2009 SIGCOMM Conference and Co-Located Workshops, VISA 2009*. [Online]. 2009 p. Available from: doi:10.1145/1592648.1592662.

Liu, F., Tong, J., Mao, J., Bohn, R., et al. (2011) NIST cloud computing reference architecture. *NIST special publication*. 500, 292.

Liu, H. & Wang, S. (2012) The analysis and design of trusted computing applied into cloud. In: *Proceedings - 2012 IEEE Control and System Graduate Research Colloquium, ICSGRC 2012*. [Online]. 2012 pp. 5–9. Available from: doi:10.1109/ICSGRC.2012.6287124.

Liu, H.C., You, J.X., Zhen, L. & Fan, X.J. (2014) A novel hybrid multiple criteria decision making model for material selection with target-based criteria. *Materials and Design*. [Online] Available from: doi:10.1016/j.matdes.2014.03.071.

Liu, K. & Jiang, L. (2011) Bio-inspired design of multiscale structures for function integration. *Nano Today*. [Online] 6 (2), 155–175. Available from: doi:10.1016/j.nantod.2011.02.002.

Liu, W. (2012) Research on cloud computing security problem and strategy. In: *Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on*. 2012 IEEE. pp. 1216–1219.

Liu, Y., Sun, Y., Ryoo, J., Rizvi, S., et al. (2015) A survey of security and privacy challenges in cloud computing: Solutions and future directions. *Journal of Computing Science and Engineering*. [Online] Available from: doi:10.5626/JCSE.2015.9.3.119.

Lo, C.C., Huang, C.C. & Ku, J. (2010) A cooperative intrusion detection system framework for cloud computing networks. In: *Proceedings of the International Conference on Parallel Processing Workshops*. [Online]. 2010 pp. 280–284. Available from: doi:10.1109/ICPPW.2010.46.

Lo, H.-Y. & Liao, W. (2017) CALM: Survivable Virtual Data Center Allocation in Cloud Networks. *IEEE Transactions on Services Computing*. [Online] 1–1. Available from: doi:10.1109/TSC.2017.2777979.

Lombardi, F. & Pietro, R. Di (2011) Secure virtualization for cloud computing. *Journal of Network and Computer Applications*. 34 (4), 1113–1122.

Longo, F., Ghosh, R., Naik, V.K. & Trivedi, K.S. (2011) A scalable availability model for Infrastructure-as-a-Service cloud. In: *Proceedings of the International Conference on Dependable Systems and Networks*. [Online]. 2011 pp. 335–346. Available from: doi:10.1109/DSN.2011.5958247.

Lu, W., Xu, S. & Yi, X. (2013) Optimizing active cyber defense. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. [Online]. 2013 pp. 206–225. Available from: doi:10.1007/978-3-319-02786-9_13.

Lui, F., Tong, J., Mao, J., Bohn, R., et al. (2011) NIST Cloud Computing Reference Architecture: Recommendations of the National Institute of Standards and Technology. *NIST Special Publication 500-292*. [Online] Available from: doi:500-299.

Luo, Z., He, Z.R. & Li, W.T. (2004) Optimal birth control for predator-prey system of three species with age-structure. *Applied Mathematics and Computation*. [Online] 155 (3), 665–685. Available from: doi:10.1016/S0096-3003(03)00808-7.

Luttbeg, B. & Schmitz, O.J. (2000) Predator and Prey Models with Flexible Individual Behavior and Imperfect Information. *The American Naturalist*. [Online] Available from: doi:10.1086/303344.

Lytinen, S.L. & Railsback, S.F. (2012) The evolution of agent-based simulation platforms: a review of NetLogo 5.0 and ReLogo. In: *Proceedings of the fourth international symposium on agent-based modeling and simulation*. 2012 Citeseer. p.

Lytinen, S.L. & Railsback, S.F. (2010) The evolution of agent-based simulation platforms: a review of NetLogo 5.0 and ReLogo. *European Meetings on Cybernetics and Systems Research*. 1–11.

Ma, Z. (2010) Towards a unified definition for reliability, survivability and resilience (I): The conceptual framework inspired by the handicap principle and ecological stability.

IEEE Aerospace Conference Proceedings. [Online] (I). Available from: doi:10.1109/AERO.2010.5446843.

Ma, Z. (Sam), Krings, A.W. & Sheldon, F.T. (2009) An outline of the three-layer survivability analysis architecture for strategic information warfare research. In: *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research Cyber Security and Information Intelligence Challenges and Strategies - CSIIRW '09*. [Online]. 2009 p. 1. Available from: doi:10.1145/1558607.1558639.

Ma, Z., Yang, L., Neilson, R.P., Hess, A., et al. (2014) A survivability-centered research agenda for cloud computing supported emergency response and management systems. In: *2014 IEEE Aerospace Conference*. [Online]. 2014 pp. 1–17. Available from: doi:10.1109/AERO.2014.6836515.

Ma, Z.S. & Krings, A.W. (2011) Dynamic hybrid fault modeling and extended evolutionary game theory for reliability, survivability and fault tolerance analyses. *IEEE Transactions on Reliability*. [Online] 60 (1), 180–196. Available from: doi:10.1109/TR.2011.2104997.

Magalhães, D., Calheiros, R.N., Buyya, R. & Gomes, D.G. (2015) Workload modeling for resource usage analysis and simulation in cloud computing. *Computers and Electrical Engineering*. [Online] Available from: doi:10.1016/j.compeleceng.2015.08.016.

Malecic, A. (2017) *Footprints of General Systems Theory final*. [Online] 636 (September), 631–636. Available from: doi:10.1002/sres.2484.

Malterud, K. (2001) Qualitative research: standards, challenges, and guidelines. *The lancet*. 358 (9280), 483–488.

Manapragada, C., Webb, G. & Salehi, M. (2018) *Extremely Fast Decision Tree*. [Online] (August). Available from: http://arxiv.org/abs/1802.08780.

Mandiki, S.N.M., Babiak, I., Krol, J., Rasolo, J.F.R., et al. (2007) How initial predator-prey ratio affects intra-cohort cannibalism and growth in Eurasian perch Perca fluviatilis L larvae and juveniles under controlled conditions. *Aquaculture*. [Online] 268 (1-4 SPEC. ISS.), 149–155. Available from: doi:10.1016/j.aquaculture.2007.04.036.

Mansouri, K., Alti, A., Roose, P. & Laborie, S. (2018) Dynamic semantic-based green bio-inspired approach for optimizing energy and cloud services qualities. *Transactions on Emerging Telecommunications Technologies*. [Online] Available from: doi:10.1002/ett.3305.

Marmol, F.G., Perez, G.M. & Skarmeta, A.F.G. (2009) TACS, a trust model for P2P networks. *Wireless personal communications*. 51 (1), 153–164.

Marshak, D. & Duer, K. (2016) *Intelligent Finding Analytics: Your Cognitive Computing*

Application Security Expert. [Online]. 2016. Application Security. Available from: https://securityintelligence.com/intelligent-finding-analytics-cognitive-computing-application-security-expert/ [Accessed: 21 December 2017].

Mateos, C., Pacini, E. & Garino, C.G. (2013) An ACO-inspired algorithm for minimizing weighted flowtime in cloud-based parameter sweep experiments. *Advances in Engineering Software*. 56, 38–50.

Mathisen, E. (2011) Security challenges and solutions in cloud computing. In: *In Digital Ecosystems and Technologies Conference (DEST), 2011 Proceedings of the 5th IEEE International Conference*. 2011 pp. 208–212.

Matsuda, H., Hori, M. & Abrams, P.A. (1996) Effects of predator-specific defence on biodiversity and community complexity in two-trophic-level communities. *Evolutionary Ecology*. 10 (1), 13–28.

Mazur, S., Blasch, E., Chen, Y. & Skormin, V. (2011) Mitigating cloud computing security risks using a self-monitoring defensive scheme. In: *Aerospace and Electronics Conference (NAECON), Proceedings of the 2011 IEEE National*. 2011 IEEE. pp. 39–45.

McGee, S., Sabett, R. V & Shah, A. (2013) Adequate Attribution: A Framework for Developing a National Policy for Private Sector Use of Active Defense. *J.Bus.& Tech.L*. 8, 1.

McQueen, M.A. & Boyer, W.F. (2009) Deception used for cyber defense of control systems. In: *Proceedings of the 2nd conference on Human System Interactions, HSI*. 2009 p.

Mead, N.R., Ellison, R.J., Linger, R.C., Longstaff, T., et al. (2000) *Survivable Network Analysis Method*. [Online] (September), 1–59. Available from: http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA383771%5Cnhttp://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA383771.

Medathati, N.V.K., Neumann, H., Masson, G.S. & Kornprobst, P. (2015) Bio-inspired computer vision: Towards a synergistic approach of artificial and biological vision. *Computer Vision and Image Understanding*. [Online] Available from: doi:10.1016/j.cviu.2016.04.009.

Mehresh, R. & Upadhyaya, S. (2012) A Deception Framework for Survivability Against Next Generation Cyber Attacks. *Proceedings of the International Conference on Security and Management (SAM). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp)*. (Section 5).

Mehresh, R. & Upadhyaya, S. (2015) Surviving advanced persistent threats in a distributed environment – Architecture and analysis. *Information Systems Frontiers*. [Online] 17 (5), 987–995. Available from: doi:10.1007/s10796-015-9569-y.

Meisel, M., Pappas, V. & Zhang, L. (2010) A taxonomy of biologically inspired research in computer networking. *Computer Networks*. 54 (6), 901–916.

Mekdeci, B., Ross, A.M., Rhodes, D.H. & Hastings, D.E. (2011) Examining Survivability of Systems of Systems. *Symposium A Quarterly Journal In Modern Foreign Literatures*. [Online] Available from: doi:10.1002/j.2334-5837.2011.tb01226.x.

Mell, P. & Grance, T. (2011) *The NIST definition of cloud computing*.

Mezzetti, C. & Samuelson, L. (2006) Evolutionary Games and Equilibrium Selection. *Southern Economic Journal*. [Online] Available from: doi:10.2307/1061220.

Microsoft Mobility Management (2016) *Cloud App Security What does Cloud App Security provide ?*

Mirza, R.S. & Chivers, D.P. (2001) Chemical alarm signals enhance survival of brook charr (Salvelinus fontinalis) during encounters with predatory chain pickerel (Esox niger). *Ethology*. [Online] 107 (11), 989–1005. Available from: doi:10.1046/j.1439-0310.2001.00729.x.

Modi, C., Patel, D., Borisaniya, B., Patel, H., et al. (2013) A survey of intrusion detection techniques in cloud. *Journal of Network and Computer Applications*. 36 (1), 42–57.

Moradi, M. (2016) A centralized reinforcement learning method for multi-agent job scheduling in Grid. In: *2016 6th International Conference on Computer and Knowledge Engineering, ICCKE 2016*. [Online]. 2016 p. Available from: doi:10.1109/ICCKE.2016.7802135.

Moreno, C., Aquino, R., Ibarreche, J. & Perez, I. (2019) Rivercore: IoT device for river water level monitoring over cellular communications. *Sensors (Switzerland)*. [Online] Available from: doi:10.3390/s19010127.

Morrow, B. (2012) BYOD security challenges: Control and protect your most sensitive data. *Network Security*. [Online] Available from: doi:10.1016/S1353-4858(12)70111-3.

Mosharaf, N.M., Chowdhury, K., Rahman, M.R. & Boutaba, R. (2009) Virtual network embedding with coordinated node and link mapping. In: *Proceedings - IEEE INFOCOM*. [Online]. 2009 p. Available from: doi:10.1109/INFCOM.2009.5061987.

Motulsky, H. (2002) The link between error bars and statistical significance. *Graphpad.com*. [Online] Available from: doi:10.1007/s10350-007-9150-y.

Mthunzi, S.N. & Benkhelifa, E. (2017) Trends towards Bio-Inspired Security Countermeasures for Cloud Environments. In: *Proceedings - 2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems, FAS*W 2017*.

[Online]. 2017 pp. 341–347. Available from: doi:10.1109/FAS-W.2017.170.

Mthunzi, S.N., Benkhelifa, E., Alsmirat, M.A. & Jararweh, Y. (2018) *Analysis of VM Communication for VM-based Cloud Security Systems **. [Online] 182–188. Available from: doi:10.1109/SDS.2018.8370441.

Munaretto, D., An, C., Widmer, J. & Timm-Giel, A. (2011) Resilient data gathering and communication algorithms for emergency scenarios. In: *Telecommunication Systems*. [Online]. 2011 p. Available from: doi:10.1007/s11235-010-9346-7.

Munger, D. (2008) *Most researchers [do not] understand error bars*. 2008. ScienceBlogs.

MURAT, M.T. (2015) *Diffusion of Innovation and Collective Action in Complex Networks*.

Nagarajan, A. (n.d.) *Realizing Cyber Resilience with Hybrid Intrusion Tolerance Architectures*.

Nagel, J.K.S. & Stone, R.B. (2011a) A systematic approach to biologically-inspired engineering design. In: *Proceedings of the ASME Design Engineering Technical Conference*. [Online]. 2011 pp. 153–164. Available from: doi:10.1115/DETC2011-47398.

Nagel, J.K.S. & Stone, R.B. (2011b) A systematic approach to biologically-inspired engineering design. In: *Proceedings of the ASME Design Engineering Technical Conference*. [Online]. 2011 pp. 153–164. Available from: doi:10.1115/DETC2011-47398.

Nagel, J.K.S., Stone, R.B. & Mcadams, D.A. (2010) An Engineering-To-Biology Thesaurus for Engineering Design. *ASME 2010 International Design Engineering Technical Conference & Computers and INformation in Engineering Conference*. [Online] (October), 1–11. Available from: doi:10.1115/DETC2010-28233.

Nakano, T. (2011) Biologically inspired network systems: a review and future prospects. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*. 41 (5), 630–643.

Nasr, K., Abou El Kalam, A. & Fraboul, C. (2011) A holistic methodology for evaluating wireless Intrusion Detection Systems. *Proceedings - 2011 5th International Conference on Network and System Security, NSS 2011*. [Online] Available from: doi:10.1109/ICNSS.2011.6059954.

National Institute of Standards and Technology (2018) *Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1*. [Online] Available from: doi:10.6028/NIST.CSWP.04162018.

Nguyen, T.A., Kim, D.S. & Park, J.S. (2016) Availability modeling and analysis of a data center for disaster tolerance. *Future Generation Computer Systems*. [Online] 56, 27–50. Available from: doi:10.1016/j.future.2015.08.017.

Ni, N.M., McCalley, J.D., Vittal, V. & Tayyib, T. (2002) Online Risk-Based Security

Assessment. *IEEE Power Engineering Review*. [Online]. Available from: doi:10.1109/MPER.2002.4311832.

Nickerson, R.C., Varshney, U. & Muntermann, J. (2013) A method for taxonomy development and its application in information systems. *European Journal of Information Systems*. [Online] Available from: doi:10.1057/ejis.2012.26.

NIST (2016) *NIST Definition of Cloud Computing*. [Online]. 2016. The National Institute of Standards and Technology (NIST). Available from: http://www.nist.gov/itl/cloud/.

O'Connell, P.E., Cunge, J.A., Erlich, M. & Bomel, P. (2014) Assessment of large transport infrastructure projects: The CBA-DK model. *Journal of Construction Engineering and Management*. [Online] 2 (1), 131–137. Available from: doi:10.1007/s11270-013-1706-y.

O'Connor, P.D.T. & Kleyner, A. (2011) *Practical Reliability Engineering: Fifth Edition*. [Online]. Available from: doi:10.1002/9781119961260.

Oates, R., Milford, M., Wyeth, G., Kendall, G., et al. (2009) The implementation of a novel, bio-inspired, robotic security system. In: *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. 2009 IEEE. pp. 1875–1880.

Oberkampf, W.L. & Roy, C.J. (2011) *Verification and validation in scientific computing*. [Online]. Available from: doi:10.1017/CBO9780511760396.

Olumide, A., Alsadoon, A., Prasad, P.W.C. & Pham, L. (2015) A hybrid encryption model for secure cloud computing. In: *ICT and Knowledge Engineering (ICT & Knowledge Engineering 2015), 2015 13th International Conference on*. 2015 IEEE. pp. 24–32.

Oreifej, R.S., Al-Haddad, R., Zand, R., Ashraf, R.A., et al. (2018) Survivability Modeling and Resource Planning for Self-Repairing Reconfigurable Device Fabrics. *IEEE Transactions on Cybernetics*. [Online] 48 (2), 780–792. Available from: doi:10.1109/TCYB.2017.2655878.

Over, S. (2014) *Symantec Data Center Security :* 1–5.

Padhy, R.P., Patra, M.R. & Satapathy, S.C. (2011) Virtualization techniques & technologies: State-of-the-art. *Journal of Global Research in Computer Science*. [Online] 2, 29–43. Available from: http://www.jgrcs.info/index.php/jgrcs/article/view/269/233.

Panigrahi, D. (2013) *Survivable Network Design Problems in Wireless Networks*. In: [Online]. 2013 p. Available from: doi:10.1137/1.9781611973082.78.

Paquette, S., Jaeger, P.T. & Wilson, S.C. (2010) Identifying the security risks associated with governmental use of cloud computing. *Government Information Quarterly*. [Online] Available from: doi:10.1016/j.giq.2010.01.002.

Park, S. & Ruighaver, T. (2008) Strategic approach to information security in organizations. In: *Information Science and Security, 2008. ICISS. International Conference on*. 2008 IEEE. pp. 26–31.

Parker, M. & Kamenev, A. (2009) Extinction in the lotka-volterra model. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*. [Online] Available from: doi:10.1103/PhysRevE.80.021129.

Pearce, M., Zeadally, S. & Hunt, R. (2013) Virtualization: Issues, security threats, and solutions. *ACM Computing Surveys (CSUR)*. [Online] Available from: doi:10.1145/2431211.2431216.

Pearson, S. & Benameur, A. (2010) Privacy, Security and Trust Issues Arising from Cloud Computing. *2010 IEEE Second International Conference on Cloud Computing Technology and Science*. [Online] 693–702. Available from: doi:10.1109/CloudCom.2010.66.

Petchey, O.L. (2000) Prey diversity, prey composition, and predator population dynamics in experimental microcosms. *Journal of Animal Ecology*. [Online] Available from: doi:10.1046/j.1365-2656.2000.00446.x.

Petukhov, A. & Kozlov, D. (2008) Detecting security vulnerabilities in web applications using dynamic analysis with penetration testing. In: *Application Security Conference (OWASP)*. 2008 p.

Pinol, C.M.N. & Banzon, R.S. (2011) Stability in a population model without random deaths by the Verhulst factor. *Physica A: Statistical Mechanics and its Applications*. 390 (7), 1295–1299.

Pokharel, M., Lee, S.L.S. & Park, J.S.P.J.S. (2010) Disaster Recovery for System Architecture Using Cloud Computing. In: *Applications and the Internet (SAINT), 2010 10th IEEE/IPSJ International Symposium on*. [Online]. 2010 pp. 304–307. Available from: doi:10.1109/SAINT.2010.23.

Polash, F., Abuhussein, A. & Shiva, S. (2014) A survey of cloud computing taxonomies: Rationale and overview. In: *Internet Technology and Secured Transactions (ICITST), 2014 9th International Conference for*. 2014 IEEE. pp. 459–465.

Prasad, P., Ojha, B., Shahi, R.R., Lal, R., et al. (2011) 3 Dimensional security in cloud computing. In: *ICCRD2011 - 2011 3rd International Conference on Computer Research and Development*. [Online]. 2011 p. Available from: doi:10.1109/ICCRD.2011.5764279.

Priami, C. (2009) Algorithmic systems biology. *Communications of the ACM*. 52 (5), 80–88.

Prusty, A.R., Sethi, S. & Nayak, A.K. (2016) Analysis of energy complexity under mobility for cluster routing in Wireless Ad hoc Sensor Networks. In: *2015 International*

Conference on Microwave, Optical and Communication Engineering, ICMOCE 2015. [Online]. 2016 p. Available from: doi:10.1109/ICMOCE.2015.7489755.

PwC (2015) *INFORMATION SECURITY BREACHES SURVEY: Technical-Report*. [Online]. Available from: https://www.pwc.co.uk/assets/pdf/2015-isbs-technical-report-blue-03.pdf.

Quach, D.Q., Willemse, J.M., Preez, V. Du & Hawick, K.A. (2013) *Species Survivability and Altitude Dependence in a Lotka-Volterra Predator-Prey Spatial-Agent Based System Altitudes and the Spatial Lotka-*. [Online] 7. Available from: http://worldcomp-proceedings.com/proc/p2013/BIC7290.pdf.

Rabai, L.B.A., Jouini, M., Aissa, A. Ben & Mili, A. (2013) A cybersecurity model in cloud computing environments. *Journal of King Saud University-Computer and Information Sciences*. 25 (1), 63–75.

Rafsanjani, M.K. & Fatemidokht, H. (2015) FBeeAdHoc: A secure routing protocol for BeeAdHoc based on fuzzy logic in MANETs. *AEU-International Journal of Electronics and Communications*. 69 (11), 1613–1621.

Ragin, C.C. (2014) *The Comparative Method - Moving Beyond Qualitative and Quantitative Strategies*. [Online]. Available from: http://wisc.eblib.com/patron/FullRecord.aspx?p=1698820.

Rahman, M.R. & Boutaba, R. (2013) SVNE: Survivable virtual network embedding algorithms for network virtualization. *IEEE Transactions on Network and Service Management*. [Online] 10 (2), 105–118. Available from: doi:10.1109/TNSM.2013.013013.110202.

Rahman, N.H.A. & Choo, K.-K.R. (2015) A survey of information security incident handling in the cloud. *Computers & Security*. 49, 45–69.

Railsback, S.F., Lytinen, S.L. & Jackson, S.K. (2006) Agent-based simulation platforms: Review and development recommendations. *Simulation*. 82 (9), 609–623.

Rao, J.R., Chari, S.N., Pendarakis, D., Sailer, R., Stoecklin, M.P., Teiken, W. and Wespi, A. (2016) Security 360°: Enterprise security for the cognitive era. *IBM Journal of Research and Development*. 60 (4), 1–1.

Rao, R.V. & Selvamani, K. (2015) Data Security Challenges and Its Solutions in Cloud Computing. *Procedia Computer Science*. [Online] 48 (Iccc), 204–209. Available from: doi:10.1016/j.procs.2015.04.171.

Razzaq, A., Latif, K., Farooq Ahmad, H., Hur, A., et al. (2014) Semantic security against web application attacks. *Information Sciences*. [Online] Available from:

doi:10.1016/j.ins.2013.08.007.

Redman, J., Warren, M. & Hutchinson, W. (2005) System survivability: a critical security problem. *Information Management & Computer Security*. [Online] Available from: doi:10.1108/09685220510602004.

Ren, S., Yu, Y., Kwiat, K.A. & Tsai, J. (2007) A coordination model for improving software system attack-tolerance and survivability in open hostile environments. *International Journal of Distributed Sensor Networks*. [Online] 3 (2), 175–199. Available from: doi:10.1080/15501320701205068.

Ren, Y., Liu, L., Liu, X., Kong, J., et al. (2012) A fast and transparent communication protocol for co-resident virtual machines. *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2012 8th International Conference on*. [Online] 70–79. Available from: doi:10.4108/icst.collaboratecom.2012.250405.

Resetarits, W.J. (2001) Colonization under threat of predation: avoidance of fish by an aquatic beetle, Tropisternus lateralis (Coleoptera: Hydrophilidae). *Oecologia*. 129 (1), 155–160.

Ribeiro, C.C. & Hansen, P. (2012) *Essays and surveys in metaheuristics*. Springer Science & Business Media.

Richards, M.G., Hastings, D.E., Rhodes, D.H. & Weigel, A.L. (2007) Defining Survivability for Engineering Systems. *Conference on Systems Engineering Research*. 1–12.

Rieke, R., Coppolino, L., Hutchison, A., Prieto, E., et al. (2012) Security and reliability requirements for advanced security event management. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. [Online]. 2012 p. Available from: doi:10.1007/978-3-642-33704-8-15.

Ristenpart, T., Tromer, E., Shacham, H. & Savage, S. (2009) Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: *Proceedings of the 16th ACM conference on Computer and communications security*. 2009 ACM. pp. 199–212.

Rodríguez, R.J., Merseguer, J. & Bernardi, S. (2014) Modelling Security of Critical Infrastructures: A Survivability Assessment. *Computer Journal*. [Online] 58 (10), 2313–2327. Available from: doi:10.1093/comjnl/bxu096.

Romine, C.H. (2019) *Cyber Crime: An Existential Threat to Small Business*. [Online]. 2019. Information Technology Laboratory National Institute of Standards and Technology. Available from: https://www.nist.gov/speech-testimony/cyber-crime-existential-threat-small-business [Accessed: 22 May 2019].

Rong, C., Nguyen, S.T. & Jaatun, M.G. (2013a) Beyond lightning: A survey on security challenges in cloud computing. *Computers & Electrical Engineering*. 39 (1), 47–54.

Rong, C., Nguyen, S.T. & Jaatun, M.G. (2013b) Beyond lightning: A survey on security challenges in cloud computing. *Computers & Electrical Engineering*. [Online] 39 (1), 47–54. Available from: doi:10.1016/j.compeleceng.2012.04.015.

Roux, A. Le, Cherry, M.I., Gygax, L. & Manser, M.B. (2009) Vigilance behaviour and fitness consequences: comparing a solitary foraging and an obligate group-foraging mammal. *Behavioral Ecology and Sociobiology*. 63 (8), 1097–1107.

Roy, A., Sarkar, S., Ganesan, R. & Goel, G. (2015) Secure the Cloud. *ACM Computing Surveys*. [Online] 47 (3), 1–30. Available from: doi:10.1145/2693841.

Rozenfeld, A.F., Luis Gruver, J., Albano, E. V. & Havlin, S. (2006) Altruism: A natural strategy for enhancing survival. *Physica A: Statistical Mechanics and its Applications*. [Online] 369 (2), 817–822. Available from: doi:10.1016/j.physa.2006.01.088.

Russo, D. & Spreafico, C. (2015) TRIZ 40 Inventive principles classification through FBS ontology. *Procedia Engineering*. [Online] 131, 737–746. Available from: doi:10.1016/j.proeng.2015.12.367.

Ryan, M.D. (2013) Cloud computing security: The scientific challenge, and a survey of solutions. *Journal of Systems and Software*. [Online] 86 (9), 2263–2268. Available from: doi:10.1016/j.jss.2012.12.025.

Sabahi, F. (2011) Cloud computing security threats and responses. In: *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*. 2011 IEEE. pp. 245–249.

Sadava, D.M., Hillis, D., Heller, H.C. & Berenbaum, M.R. (2011) *Life: The science of biology. 9th edition*. [Online]. Available from: doi:10.1073/pnas.0404206101.

Saghafian, S. (2018) Ambiguous partially observable Markov decision processes: Structural results and applications. *Journal of Economic Theory*. [Online] 178 (Cmmi), 1–35. Available from: doi:10.1016/j.jet.2018.08.006.

Sahoo, J., Mohapatra, S. & Lath, R. (2010) Virtualization: A survey on concepts, taxonomy and associated security issues. In: *2nd International Conference on Computer and Network Technology, ICCNT 2010*. [Online]. 2010 p. Available from: doi:10.1109/ICCNT.2010.49.

Sakellari, G. & Loukas, G. (2013) A survey of mathematical models, simulation approaches and testbeds used for research in cloud computing. *Simulation Modelling Practice and Theory*. [Online] 39, 92–103. Available from: doi:10.1016/j.simpat.2013.04.002.

SANS (2016) *No TitleCIS Critical Security Controls*. 2016. The CIS Critical Security Controls for Effective Cyber Defense.

Sarkar, A., Bhattacharya, A., Dutta, S. & Parikh, K.K. (2019) Recent Trends of Data Mining in Cloud Computing. In: *Data Mining and Information Security*. Singapore, Springer. pp. 565–578.

Saudi, M.M., Woodward, M., Cullen, A.J. & Noor, H.M. (2008) An overview of apoptosis for computer security. In: *Information Technology, 2008. ITSim 2008. International Symposium on*. 2008 IEEE. pp. 1–6.

Saunders, M.N.K. (2009) Research Onion. *Unversity of Derby*. [Online]. Available from: doi:10.1007/s13398-014-0173-7.2.

Savransky, S.D. (2000) *Engineering of creativity: Introduction to TRIZ methodology of inventive problem solving*. CRC Press.

Sayed, A.H. (2014) Adaptive networks. *Proceedings of the IEEE*. 102 (4), 460–497.

Schoener, T.W., Spiller, D.A. & Losos, J.B. (2001) Predators increase the risk of catastrophic extinction of prey populations. *Nature*. [Online] 412 (6843), 183–186. Available from: doi:10.1038/35084071.

Security, F.N., Fireeye, W., Security, N., Windows, M., et al. (2018) *FireEye Network Security*. 1–12.

Serageldin, A., Krings, A. & Abdel-Rahim, A. (2013) *A survivable critical infrastructure control application*. In: [Online]. 2013 p. Available from: doi:10.1145/2459976.2460015.

Server, S. (2015) *Data Sheet Symantec Data Center*. 1–4.

Shahriar, N., Ahmed, R., Chowdhury, S.R., Khan, A., et al. (2017) Generalized recovery from node failure in virtual network embedding. *IEEE Transactions on Network and Service Management*. [Online] 14 (2), 261–274. Available from: doi:10.1109/TNSM.2017.2693404.

Shahzad, F. (2014) State-of-the-art Survey on Cloud Computing Security Challenges, Approaches and Solutions. *Procedia Computer Science*. [Online] 37, 357–362. Available from: doi:10.1016/j.procs.2014.08.053.

Shahzad, K. & Woodhead, S. (2014) Towards automated distributed containment of zero-day network worms. *5th International Conference on Computing Communication and Networking Technologies, ICCCNT 2014*. [Online] Available from: doi:10.1109/ICCCNT.2014.6963119.

Sharif, A.M. & Irani, Z. (2006) Exploring Fuzzy Cognitive Mapping for IS Evaluation. *European Journal of Operational Research*. [Online] Available from: doi:10.1016/j.ejor.2005.07.011.

Shehabat, A. & Mitew, T. (2017) *Distributed Swarming and Stigmergic Effects on ISIS Networks : OODA Loop Model.* 10 (December), 0–20.

Shi, L., Jia, C., Lu, S. & Liu, Z. (2007) *Port and address hopping for active cyber-defense.* In: Intelligence and Security Informatics. Springer. pp. 295–300.

Shi, T., Zhao, J., Yin, X. & Wang, J. (2008) Research on telecommunication switching system survivability based on stochastic petri net. In: *3rd International Conference on Innovative Computing Information and Control, ICICIC'08*. [Online]. 2008 p. Available from: doi:10.1109/ICICIC.2008.460.

Shi, Y., Chen, G. & Li, J. (2018) Malicious Domain Name Detection Based on Extreme Machine Learning. *Neural Processing Letters.* [Online] Available from: doi:10.1007/s11063-017-9666-7.

Shon, T., Kovah, X. & Moon, J. (2006) Applying genetic algorithm for classifying anomalous TCP/IP packets. *Neurocomputing.* [Online] 69 (16–18), 2429–2433. Available from: doi:10.1016/j.neucom.2006.01.023.

Shorov, A. & Kotenko, I. (2014) The Framework for Simulation of Bioinspired Security Mechanisms against Network Infrastructure Attacks. *The Scientific World Journal.* 2014.

da Silva Peixoto, M., de Barros, L.C. & Bassanezi, R.C. (2008) Predator-prey fuzzy model. *Ecological Modelling.* [Online] 214 (1), 39–44. Available from: doi:10.1016/j.ecolmodel.2008.01.009.

Sim, K.M. (2012) Agent-based cloud computing. *IEEE Transactions on Services Computing.* [Online] 5 (4), 564–577. Available from: doi:10.1109/TSC.2011.52.

Singh, A.K. & Srivastava, S. (2018) A survey and classification of controller placement problem in SDN. *International Journal of Network Management.* [Online] Available from: doi:10.1002/nem.2018.

Singh, R.K., Joshi, R. & Singhal, M. (2013) Analysis of security threats and vulnerabilities in mobile ad hoc network (manet). *International Journal of Computer Applications.* 68 (4).

Singh, S., Jeong, Y.S. & Park, J.H. (2016) A survey on cloud computing security: Issues, threats, and solutions. *Journal of Network and Computer Applications.* [Online] 75, 200–222. Available from: doi:10.1016/j.jnca.2016.09.002.

Skyttner, L. (2010) *General Systems Theory - Problems, Perspectives, Practice.* [Online]. Available from: doi:10.1142/9789812774750.

Smith, C.L. (1979) FUNDAMENTALS OF CONTROL THEORY. *Chemical Engineering (New York).* [Online]. 86 (22) pp.11–39. Available from: doi:doi:10.1201/9781420036572.ch14.

Smith, R.J.F. (1992) Alarm signals in fishes. *Reviews in Fish Biology and Fisheries*. 2 (1), 33–63.

Sobh, T.S. & Mostafa, W.M. (2011) A cooperative immunological approach for detecting network anomaly. *Applied Soft Computing*. 11 (1), 1275–1283.

Sochor, T. & Zuzcak, M. (2014) *Study of internet threats and attack methods using honeypots and honeynets*. In: Computer Networks. Springer. pp. 118–127.

Solutions, V.E. (2015) 2015 Dbir. *Verizon Business Journal*. [Online] Available from: http://www.verizonenterprise.com/resources/reports/rp_data-breach-investigation-report_2015_en_xg.pdf.

Somayaji, A., Locasto, M. & Feyereisl, J. (2007) Panel : The Future of Biologically-Inspired Security : Is There Anything Left to Learn? In: *NSPW'07, September 18-21, 2007, North Conway, NH, USA*. [Online]. 2007 pp. 49–54. Available from: doi:10.1145/1600176.1600185.

Sommarive, V. & Report, T. (2013) *KEYWORD QUERY TO GRAPH Zekarias Kefato Matteo Lissandrini Davide Mottin Themis Palpanas*. (October).

Srinivasan, M.K., Sarukesi, K., Rodrigues, P., Manoj, M.S., et al. (2012) State-of-the-art cloud computing security taxonomies - A classification of security challenges in the present cloud computing environment. *Proceedings of the International Conference on Advances in Computing, Communications and Informatics - ICACCI '12*. [Online] 470. Available from: doi:10.1145/2345396.2345474.

Stage, F.K. & Manning, K. (2003) *Research in the college context: Approaches and methods*. [Online]. Available from: doi:10.4324/9780203952740.

Sterbenz, J.P.G., Hutchison, D., Cetinkaya, E.K. & Jabbar, A. (2010a) Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines. *Computer Networks*. [Online] 54 (8), 1245–1265. Available from: doi:10.1016/j.comnet.2010.03.005.

Sterbenz, J.P.G., Hutchison, D., Jabbar, A., Rohrer, J.P., et al. (2010b) Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines. *Computer Networks*. 54 (8), 1245–1265.

Stoykov, G. (2015) *A Biomorphic model for automated cloud adaptation*.

Stoykov, G. & Yazidi, A. (2016) A Biomorphic Model for Automated Cloud Adaptation. *Proceedings - 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing, UCC 2015*. [Online] 179–185. Available from: doi:10.1109/UCC.2015.34.

Subashini, S. & Kavitha, V. (2011) A survey on security issues in service delivery models of cloud computing. *Journal of network and computer applications*. 34 (1), 1–11.

Sullivan, T. & Regan, F. (2011) Biomimetic design of novel antifouling materials for application to environmental sensing technologies. *Journal of Ocean Technology*. [Online] 6 (4), 42–54. Available from: doi:10.1016/j.cirp.2011.06.001.

Sun, D., Chang, G., Sun, L., Li, F., et al. (2011) A dynamic multi-dimensional trust evaluation model to enhance security of cloud computing environments. *International Journal of Innovative Computing and Applications*. [Online] 3 (4), 200–212. Available from: doi:10.1504/IJICA.2011.044529.

Sun, F. & Cheng, S. (2009) A gene technology inspired paradigm for user authentication. In: *Bioinformatics and Biomedical Engineering, 2009. ICBBE 2009. 3rd International Conference on*. 2009 IEEE. pp. 1–3.

Sundell, J., Dudek, D., Klemme, I., Koivisto, E., et al. (2004) Variation in predation risk and vole feeding behaviour: A field test of the risk allocation hypothesis. *Oecologia*. [Online] Available from: doi:10.1007/s00442-004-1490-x.

Systems, C. (2006) *思科 Adaptive Security Appliance 产品动手培训实验指南 实验环境说明*. 1–52.

Tahvildari, M.S. and L. (2009) “Self-adaptive software: Landscape and research challenges”. *ACM Transactions on Autonomous and Adaptive Systems*.

Takabi, H., Joshi, J.B.D. & Ahn, G.-J. (2010) Security and privacy challenges in cloud computing environments. *IEEE Security & Privacy*. (6), 24–31.

Taleb, T., Samdanis, K., Mada, B., Flinck, H., et al. (2017) On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration. *IEEE Communications Surveys and Tutorials*. [Online] Available from: doi:10.1109/COMST.2017.2705720.

Talia, D. (2012) Clouds meet agents: Toward intelligent cloud services. *IEEE Internet Computing*. [Online]. 16 (2) pp.78–81. Available from: doi:10.1109/MIC.2012.28.

Tara, S. (2018) *Cloud security concerns surge*. [Online]. 2018. Infosecurity Manazine. Available from: https://www.infosecurity-magazine.com/news/cloud-security-concerns-surge/ [Accessed: 22 December 2019].

Thornton, A. & Clutton-Brock, T. (2011) Social learning and the development of individual and group behaviour in mammal societies. *Philosophical transactions of the Royal Society of London.Series B, Biological sciences*. [Online] 366 (1567), 978–987. Available from: doi:10.1098/rstb.2010.0312 [doi].

Tian, H., Chen, Z., Chang, C.C., Huang, Y., et al. (2019) Public audit for operation behavior logs with error locating in cloud storage. *Soft Computing*. [Online] Available from: doi:10.1007/s00500-018-3038-8.

Toemeh, R. & Arumugam, S. (2015) Breaking Transposition Cipher with Genetic Algorithm. *Elektronika ir Elektrotechnika*. 79 (7), 75–78.

Vakili, V. & Shu, L.H. (2001) Towards Biomimetic Concept Generation. *ASME 2001 Design Engineering Technical Conferences*. [Online] 4 (February), 1–9. Available from: http://www.mie.utoronto.ca/labs/bidlab/pubs/Vakili_Shu_DTM_01.pdf.

Vassalos, D. (2019) Damage stability of passenger ships—notions and truths. In: *Fluid Mechanics and its Applications*. [Online]. pp. 779–801. Available from: doi:10.1007/978-3-030-00516-0_46.

Verizon (2016) *Evolution of the Verizon Data Breach Investigations Report: 2008-2016*. [Online]. 2016. Available from: http://www.verizonenterprise.com/verizon-insights-lab/dbir/ [Accessed: 10 December 2016].

Verizon Business (2011) 2011 Data Breach Investigations Report (DBIR). *Trends*. [Online] 1–72. Available from: doi:10.1109/CyberSec.2012.6246130.

Verma, A.K., Dave, M. & Joshi, R.C. (2007) Genetic algorithm and tabu search attack on the mono-alphabetic substitution cipher i adhoc networks. In: *Journal of Computer science*. 2007 Citeseer. p.

Vermeij, G.J. (1994) The evolutionary interaction among species: Selection, escalation, and coevolution. *Annual Review of Ecology and Systematics*. [Online]. Available from: doi:10.1146/annurev.es.25.110194.001251.

Vessey, I., Ramesh, V. & Glass, R.L. (2005) A unified classification system for research in the computing disciplines. *Information and Software Technology*. [Online] Available from: doi:10.1016/j.infsof.2004.08.006.

Vincent, J.F.V., Bogatyreva, O.A., Bogatyrev, N.R., Bowyer, A., et al. (2006) Biomimetics: Its practice and theory. *Journal of the Royal Society Interface*. [Online] 3 (9), 471–482. Available from: doi:10.1098/rsif.2006.0127.

Virvilis, N. & Gritzalis, D. (2013) The big four-what we did wrong in advanced persistent threat detection? In: *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on*. 2013 IEEE. pp. 248–254.

Vitanov, N.K., Dimitrova, Z.I. & Ausloos, M. (2010) Verhulst-Lotka-Volterra (VLV) model of ideological struggle. *Physica A: Statistical Mechanics and its Applications*. [Online] 389 (21), 4970–4980. Available from: doi:10.1016/j.physa.2010.06.032.

Voellmy, I.K., Goncalves, I.B., Barrette, M.-F., Monfort, S.L., et al. (2014) Mean fecal glucocorticoid metabolites are associated with vigilance, whereas immediate cortisol levels better reflect acute anti-predator responses in meerkats. *Hormones and behavior*. 66 (5), 759–765.

Wagle, S.S., Guzek, M., Bouvry, P. & Bisdorff, R. (2016) An evaluation model for selecting cloud services from commercially available cloud providers. In: *Proceedings - IEEE 7th International Conference on Cloud Computing Technology and Science, CloudCom 2015*. [Online]. 2016 pp. 107–114. Available from: doi:10.1109/CloudCom.2015.94.

Wainer, J., Novoa Barsottini, C.G., Lacerda, D. & Magalhães de Marco, L.R. (2009) Empirical evaluation in Computer Science research published by ACM. *Information and Software Technology*. [Online] Available from: doi:10.1016/j.infsof.2009.01.002.

Wallom, D., Turilli, M., Martin, A., Raun, A., et al. (2011) MyTrustedCloud: Trusted cloud infrastructure for security-critical computation and data managment. In: *Proceedings - 2011 3rd IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2011*. [Online]. 2011 p. Available from: doi:10.1109/CloudCom.2011.41.

Walraven, S., Truyen, E. & Joosen, W. (2011) A middleware layer for flexible and cost-efficient multi-tenant applications. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. [Online]. 2011 p. Available from: doi:10.1007/978-3-642-25821-3_19.

Walther, F.R. (1969) Flight behaviour and avoidance of predators in Thomson's gazelle (Gazella thomsoni Guenther 1884). *Behaviour*. 34 (3), 184–220.

Waltman, P., Braselton, J. & Braselton, L. (2002) A mathematical model of a biological arms race with a dangerous prey. *Journal of Theoretical Biology*. [Online] 218 (1), 55–70. Available from: doi:10.1006/jtbi.2002.3057.

Wang, C., Fang, L., Dai, Y., Ming, L., et al. (2012a) Network survivability evaluation model based on immune evolution and multiple criteria decision making. *Proceedings of the 2012 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2012*. [Online] 178–184. Available from: doi:10.1109/CyberC.2012.37.

Wang, C., Ming, L., Zhao, J. & Wang, D. (2012b) Shielding wireless sensor network using Markovian intrusion detection system with attack pattern mining. *Ecological Modelling*. [Online] 11 (1), 32–44. Available from: doi:10.1016/j.ins.2011.03.014.

Wang, C. & Yan, H. (2010) Study of cloud computing security based on private face recognition. In: *Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on*. 2010 IEEE. pp. 1–5.

Wang, J. & Mu, S. (2011) Security issues and countermeasures in cloud computing. In: *Grey Systems and Intelligent Services (GSIS), 2011 IEEE International Conference on*. 2011 IEEE. pp. 843–846.

Wang, P., Lin, W.-H., Kuo, P.-T., Lin, H.-T., et al. (2012c) Threat risk analysis for cloud security based on Attack-Defense Trees. In: *Computing Technology and Information Management (ICCM), 2012 8th International Conference on*. 2012 IEEE. pp. 106–111.

Wang, T., Ye, B., Li, Y. & Yang, Y. (2010) Family gene based Cloud Trust model. In: *ICENT 2010 - 2010 International Conference on Educational and Network Technology*. [Online]. 2010 pp. 540–544. Available from: doi:10.1109/ICENT.2010.5532096.

Wang, W., Zeng, G. & Yuan, L. (2006) Ant-based reputation evidence distribution in P2P networks. In: *Grid and Cooperative Computing, 2006. GCC 2006. Fifth International Conference*. 2006 IEEE. pp. 129–132.

Wang, Y., Wang, Y., Liu, J. & Huang, Z. (2014) A network gene-based framework for detecting advanced persistent threats. In: *Proceedings - 2014 9th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC 2014*. [Online]. 2014 p. Available from: doi:10.1109/3PGCIC.2014.41.

Wangersky, P.J. (2003) Lotka-Volterra Population Models. *Annual Review of Ecology and Systematics*. [Online] Available from: doi:10.1146/annurev.es.09.110178.001201.

Warfield, J.N. & Staley, S.M. (1996) Structural thinking: Organizing complexity through disciplined activity. *Systems Research and Behavioral Science*.

Wen, G., Yu, W., Yu, X. & Lü, J. (2017) Complex cyber-physical networks: From cybersecurity to security control. *Journal of Systems Science and Complexity*. [Online] 30 (1), 46–67. Available from: doi:10.1007/s11424-017-6181-x.

Weyns, D., Schmerl, B., Grassi, V., Malek, S., et al. (2013) On patterns for decentralized control in self-adaptive systems. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. [Online]. 2013 pp. 76–107. Available from: doi:10.1007/978-3-642-35813-5_4.

Wilensky, U. (1999) *NetLogo*. [Online]. 1999. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. Available from: https://ccl.northwestern.edu/netlogo/ [Accessed: 23 June 2016].

Wiles, J. (2018) Top Risks for Legal and Compliance Leaders in 2018. *2018 Legal and Compliance Hot Spots*. [Online] 13 March. Available from: https://www.gartner.com/smarterwithgartner/top-risks-for-legal-and-compliance-leaders-in-2018/.

Williams, A.C. & Flaxman, S.M. (2012) Can predators assess the quality of their prey�s resource? *Animal Behaviour*. 83 (4), 883–890.

Xiao, L., Xu, D., Mandayam, N.B. & Poor, H.V. (2018) Attacker-Centric View of a Detection Game against Advanced Persistent Threats. *IEEE Transactions on Mobile Computing*. [Online] Available from: doi:10.1109/TMC.2018.2814052.

Xiao, T., Zhang, J., Yang, K., Peng, Y., et al. (2014) Error-Driven Incremental Learning in Deep Convolutional Neural Network for Large-Scale Image Classification. *Proceedings of the ACM International Conference on Multimedia - MM '14*. [Online] (November), 177–186. Available from: doi:10.1145/2647868.2654926.

Xu, J., Tang, J., Kwiat, K., Zhang, W., et al. (2013) Enhancing survivability in virtualized data centers: A service-aware approach. *IEEE Journal on Selected Areas in Communications*. [Online] 31 (12), 2610–2619. Available from: doi:10.1109/JSAC.2013.131203.

Xu, J., Tang, J., Kwiat, K., Zhang, W., et al. (2012) Survivable virtual infrastructure mapping in virtualized data centers. In: *Proceedings - 2012 IEEE 5th International Conference on Cloud Computing, CLOUD 2012*. [Online]. 2012 pp. 196–203. Available from: doi:10.1109/CLOUD.2012.100.

Xu, Y., Bailey, M., Jahanian, F., Joshi, K., et al. (2011) An exploration of L2 cache covert channels in virtualized environments. In: *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*. 2011 ACM. pp. 29–40.

Yager, R.R. (1995) An approach to ordinal decision making. *International Journal of Approximate Reasoning*. [Online] 12 (3–4), 237–261. Available from: doi:10.1016/0888-613X(94)00035-2.

Yager, R.R. (2003) *Fuzzy Modeling for Multicriteria Decision Making Under Uncertainty*. 30 (1), 60–70.

Yager, R.R. (2004) Uncertainty modeling and decision support. In: *Reliability Engineering and System Safety*. [Online]. 2004 p. Available from: doi:10.1016/j.ress.2004.03.022.

Yallouz, J. & Orda, A. (2017) Tunable QoS-Aware Network Survivability. *IEEE/ACM Transactions on Networking*. [Online] 25 (1), 139–149. Available from: doi:10.1109/TNET.2016.2606342.

Yallouz, J., Rottenstreich, O. & Orda, A. (2014) Tunable survivable spanning trees. In: *Performance Evaluation Review*. [Online]. 2014 p. Available from: doi:10.1145/2637364.2591997.

Yang, H., Luo, H., Ye, F., Lu, S., et al. (2004) Security in mobile ad hoc networks:

challenges and solutions. *Wireless Communications, IEEE*. 11 (1), 38–47.

Yang, J., Liu, X., Li, T., Liang, G., et al. (2009) Distributed agents model for intrusion detection based on AIS. *Knowledge-Based Systems*. 22 (2), 115–119.

Yannakogeorgos, P., Lowther, A. & Hayden, M. (2013) The Future of Things Cyber. In: *Conflict and Cooperation in Cyberspace*. [Online]. p. Available from: doi:10.1201/b15253-3.

Yao-Min Chen & Yanyan Yang (2004) *Policy management for network-based intrusion detection and prevention*. In: [Online]. 2004 p. Available from: doi:10.1109/noms.2004.1317855.

Yesil, E., Urbas, L. & Demirsoy, A. (2014) FCM-GUI: A graphical user interface for big bang-big crunch learning of FCM. *Intelligent Systems Reference Library*. [Online] 54, 177–198. Available from: doi:10.1007/978-3-642-39739-4_11.

Yoon, J., Yang, E., Lee, J., Hwang, S.J., et al. (2018) *Lifelong Learning with Dynamically Expandable Networks*. 1–11.

Yu, H., Anand, V., Qiao, C. & Sun, G. (2011) Cost efficient design of survivable virtual infrastructure to recover from facility node failures. In: *IEEE International Conference on Communications*. [Online]. 2011 p. Available from: doi:10.1109/icc.2011.5962604.

Yu, M., Wang, A.H., Zang, W. & Liu, P. (2010) Evaluating survivability and costs of three virtual machine based server architectures. *2010 International Conference on Security and Cryptography (SECRYPT)*. 1–8.

Yu, R., Huang, X., Kang, J., Ding, J., et al. (2015) Cooperative resource management in cloud-enabled vehicular networks. *IEEE Transactions on Industrial Electronics*. [Online] 62 (12), 7938–7951. Available from: doi:10.1109/TIE.2015.2481792.

Yu, R., Xue, G., Zhang, X. & Li, D. (2017) Survivable and bandwidth-guaranteed embedding of virtual clusters in cloud data centers. In: *Proceedings - IEEE INFOCOM*. [Online]. 2017 p. Available from: doi:10.1109/INFOCOM.2017.8056945.

Yuill, J., Denning, D.E. & Feer, F. (2006) No Title. *Using deception to hide things from hackers: Processes, principles, and techniques*.

Yurcik, W. & Doss, D. (2002) *A Survivability-Over-Security (SOS) Approach to Holistic Cyber-Ecosystem Assurance*. (June).

Zadeh, L.A. (2013) Fuzzy logic. In: *Computational Complexity: Theory, Techniques, and Applications*. [Online]. p. Available from: doi:10.1007/978-1-4614-1800-9_73.

Zadeh, L.A. (1978) Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*. [Online] Available from: doi:10.1016/0165-0114(78)90029-5.

Zafar, F., Khan, A., Malik, S.U.R., Ahmed, M., et al. (2017a) A survey of cloud computing data integrity schemes: Design challenges, taxonomy and future trends. *Computers and Security*. [Online] 65, 29–49. Available from: doi:10.1016/j.cose.2016.10.006.

Zafar, F., Khan, A., Malik, S.U.R., Ahmed, M., et al. (2017b) A survey of cloud computing data integrity schemes: Design challenges, taxonomy and future trends. *Computers and Security*. [Online] 65, 29–49. Available from: doi:10.1016/j.cose.2016.10.006.

Zanakis, S.H., Solomon, A., Wishart, N. & Dublish, S. (1998) Multi-attribute decision making: A simulation comparison of select methods. *European Journal of Operational Research*. [Online] Available from: doi:10.1016/S0377-2217(97)00147-1.

Zhan, Z.H., Liu, X.F., Gong, Y.J., Zhang, J., et al. (2015) Cloud computing resource scheduling and a survey of its evolutionary approaches. *ACM Computing Surveys*. [Online] Available from: doi:10.1145/2788397.

Zhang, Q., Liu, L., Ren, Y., Lee, K., et al. (2013) Residency aware inter-VM communication in virtualized cloud: Performance measurement and analysis. In: *IEEE International Conference on Cloud Computing, CLOUD*. [Online]. 2013 pp. 204–211. Available from: doi:10.1109/CLOUD.2013.116.

Zheng, C. & Sicker, D.C. (2013) A survey on biologically inspired algorithms for computer networking. *Communications Surveys & Tutorials, IEEE*. 15 (3), 1160–1191.

Zheng, J., Okamura, H. & Dohi, T. (2015) Survivability analysis of VM-based intrusion tolerant systems. *IEICE Transactions on Information and Systems*. [Online] E98D (12), 2082–2090. Available from: doi:10.1587/transinf.2015PAP0007.

Zhou, X. (2009) Research on immune pathology in artificial immune system. In: *Control and Decision Conference, 2009. CCDC'09. Chinese*. 2009 IEEE. pp. 1366–1370.

Zhu, J., Li, D., Wu, J., Liu, H., et al. (2012) Towards bandwidth guarantee in multi-tenancy cloud computing networks. In: *Proceedings - International Conference on Network Protocols, ICNP*. [Online]. 2012 p. Available from: doi:10.1109/ICNP.2012.6459986.

Zhuo, T., Zhengding, L. & Kai, L. (2006) Time-based dynamic trust model using ant colony algorithm. *Wuhan University Journal of Natural Sciences*. 11 (6), 1462–1466.

Ziring, N. (2015) The Future of Cyber Operations and Defense. *Warfare*. 14, 1–7.

Zissis, D. & Lekkas, D. (2012) Addressing cloud computing security issues. *Future Generation computer systems*. 28 (3), 583–592.

Zott, C., Amit, R., Massa, L. & Zhu, Y. (2012) Financial Consumer Protection and the Global Financial Crisis. *Journal of Consumer Research*. [Online] 15 (2), 1–6. Available from: doi:10.1007/978-3-642-13757-0.

Appendices

Appendix A – Definitions and Concepts

Figure 85 demonstrates the relationship between proactive, reactive and hybrid approaches. There are briefly described below, and examples presented where possible.

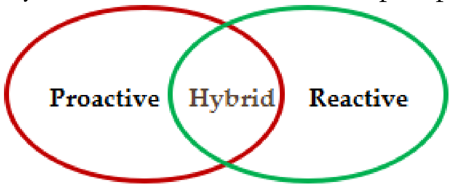


Figure 85. Proactive, hybrid and reactive security approaches

- Proactive approaches are implemented such that security systems are continuously operational and aware of changes in their environment throughout their operation (Djenouri, Khelladi & Badache, 2005). As suggested by (Yang *et al.*, 2004), these mechanisms intuitively aim to stop the occurrence of a security incident well before it occurs.
- Reactive approaches operate intermittently; active after an incident occurs and perhaps after the detection of an intrusion (Himma, 2007). These approaches aim to stop further damage to an infrastructure after intrusion. It is not uncommon for reactive approaches execute overlapping processes, i.e. the detection and response processes may run concurrent with the intrusion itself (Himma, 2007).
- Hybrid approaches offer integration opportunities, i.e. proactive and reactive processes. Depending on their implementation, hybrid approaches possess both inherent deficiencies and strengths of both (proactive and reactive approaches).

SANS defined security controls as the recommended set of actions that provide specific and actionable ways to stop attacks (SANS, 2016). Security controls are briefly described below, along with references to known examples.

- Preventative controls include devices such as firewalls, which are present to stop the occurrence of a security incident (Ko *et al.*, 2011), or limit further damage when an attack occurs.
- Detective controls include security audit trails, security analysis tools, IDS, etc. that are present to identify the presence of a security risk (Ko *et al.*, 2011).
- Corrective controls are implemented to Minimise or fix the effect of a security incident that has occurred (Ko *et al.*, 2011).
- Pre-emptive controls are described as intelligent systems that are self-defending and capable of real-time monitoring, detection and prevention of known and zero-day attacks (Behl, 2011).
- Deterrent controls may be implemented as formal goals to prevent deliberate attacks, or against violation of Information Systems (IS) security policy (Cheng *et al.*, 2013). For instance, those commonly used against software piracy, including legal aspects of copyright law, or collaboration between governments pertaining to law enforcement (Gopal & Sanders, 2000).
- Prescriptive controls can be invoked prior or post risk detection, in real-time or otherwise, to prevent any damage from a security occurrence. Such controls are exemplified in process environments, whereupon a prescription describes how a task should be completed and the order it should be performed (Heimbigner,

1990). A key advantage of prescriptive controls is their specificity and the provision of precise and immediate feedback where there is a deviation from the process (Heimbigner, 1990).

- Adaptive controls describe real-time dynamic threat intelligence, capable of identifying and blocking threats in real-time without the use of signatures. (Canetti *et al.*, 1999) discussed adaptive controls based on an attacker-security model scenario, i.e. where the attacker is dynamic, they can carry out their exploit within a permissible threshold. According to the authors, adaptive controls enable capturing of real threats (Canetti *et al.*, 1999). (Almorsy, Grundy & Müller, 2010) suggest adaptive measures to exist where security is based on current and expected threat levels in changing environmental conditions (Almorsy, Grundy & Müller, 2010).
- Deceptive controls by definition are deliberate acts to distort entity A's perception of reality as perpetrated by entity B (McQueen & Boyer, 2009). In computing terms, deceptive controls include randomization and obfuscation techniques, as well as simulation measures such as mimicking and decoying (McQueen & Boyer, 2009).

The summary of security approaches and controls are presented to highlight the security processes and procedures necessary to protect information technology infrastructures and their alignment with business objectives. This research suggests the above considerations as key for cloud computing since processes for implementing and enforcing security involve not only the organisations, but also other third parties. Since cloud computing has its foundations in traditional computing systems, cloud computing security enterprise is contrasted against traditional computing security enterprise. Traditional perimeter security, i.e. where security technologies and end-users exist within the boundaries of an organisation to protect assets inside the organisation, is considered as relatively successful. In contrast, cloud computing security, i.e. considered in relation to where control-based technology and compliancy adherence to protect data, applications and infrastructures associated with the cloud, as facing some challenges.

Appendix B - Cloud computing

Cloud computing's key cloud players (NIST, 2016) represent individuals or organisations that interact to perform tasks (Lui *et al.*, 2011). Figure 85 illustrates a generic view of the cloud computing paradigm. In practice, CSP such as Amazon EC2 is responsible for the security and availability of services to their customers, including security incidents that affect the customers' infrastructure (AlZain *et al.*, 2012).

Cloud computing is designed to facilitate optimal resource utilisation through augmenting virtualisation technologies (Hummaida, Paton & Sakellariou, 2016), both logically and physically (Zhu *et al.*, 2012)(Almorsy, Grundy & Müller, 2016). The capabilities of the tenancy/multi-tenancy concepts mean that, by co-locating users/tenants, cloud computing's compute costs are lower (Walraven, Truyen & Joosen, 2011). Multi-tenancy, as illustrated in Figure 3, enables the sharing of an application instance among multiple tenants, but with isolated regarding performance and data privacy (Coppolino *et al.*, 2017). Despite its usefulness, the literature shows that a CC's data in a multi-tenancy environment is at risk of compromise through lax access management or malicious attack. A side-channel attacker can implant arbitrary code into a neighbour's VM environment with little to no chance of detection (Ristenpart *et al.*, 2009). Moreover, multi-tenancy complicates vulnerability management due to interactions among different service domains with different service requirements (Behl & Behl, 2012). While multi-tenancy ensures that consumers are completely unaware of a neighbour's identity, their security profile or intentions, this introduces risks for co-located consumers.

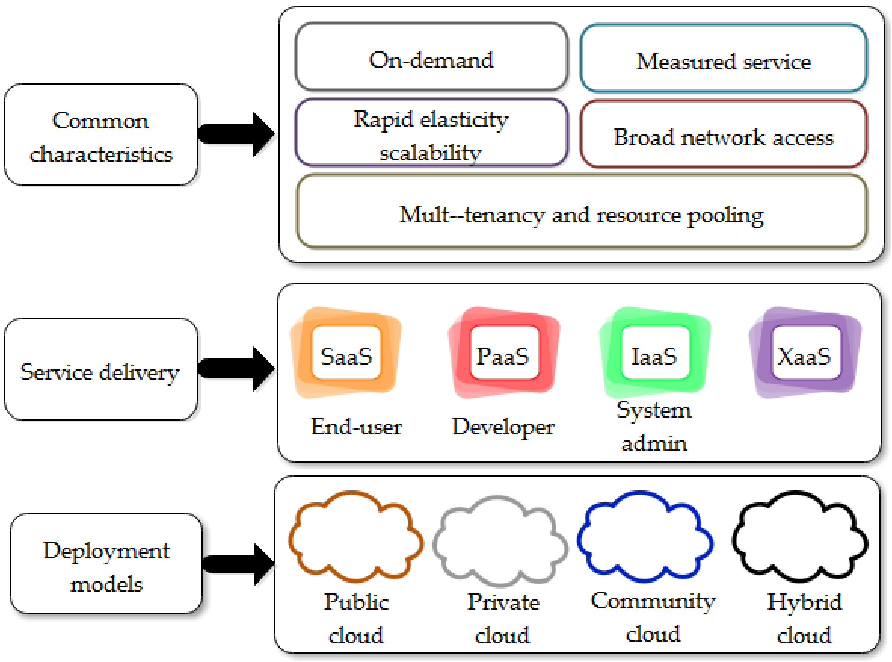


Figure 86. A generic cloud computing model

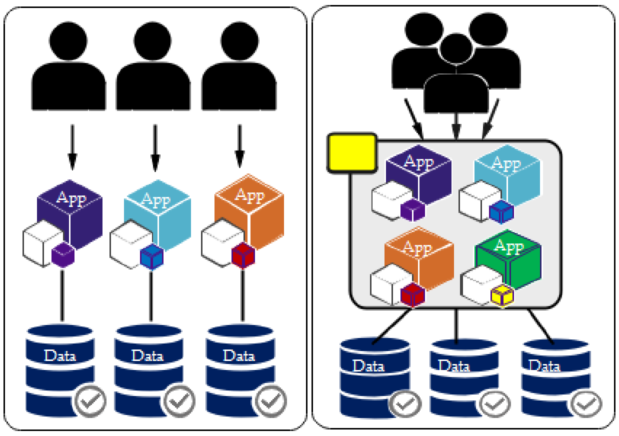


Figure 87. The tenancy concept; single tenancy (left) and multi-tenancy (right).

Cloud deployment models, including private, public, community and hybrid, offer cloud consumers a level of flexibility at the amount of control they have on data. The entity performing the role of CSP will vary according to the type of deployment model. The role of CSP may be assigned to more than one entity in a community or hybrid cloud scenario, whereas private clouds' superior security credentials are due to the amount of control an organisation has on the cloud infrastructure. To understand the assigning of responsibilities in a deployment model, the following are brief descriptions of each.
Private cloud: is managed on-premise, the CSP role is within the client organisation.

Community cloud: the CSP could be one of the client organisations within the community or a separate third party
Public cloud: the CSP is a third party that is an organisationally separate entity to its clients.
Hybrid cloud: the CSP's role is related to both internal and third-party entities for different elements of the overall cloud infrastructure.
Cloud services are provided as virtualised or abstracted infrastructures of resources (Zissis & Lekkas, 2012) and encapsulated as virtualised data, virtual machines (VMs) and hypervisors (Liu *et al.*, 2011). Responsibilities for implementing, operating and managing security controls are shared differently across the cloud's service models and thus needs to be clearly understood by both the CC and CSP.
SaaS: enables a one-to-many delivery of services and applications through a browser over the internet. SaaS allows organisations to outsource services to other parties, arguably offsetting licensing costs, servers' e.tc. Scalability means organisations can improve their infrastructural requirements at minimal additional costs.
PaaS: gives access and control to a CC, to devices and platforms that help them build applications on the internet.
IaaS: provisions the infrastructural agreements about computing systems, storage, processing power, hardware, and networking infrastructure as a service. CCs have no prior requirement for the maintenance and performance of such hardware.

The traditional and cloud security view

With the recent growth of cloud computing (Yannakogeorgos, Lowther & Hayden, 2013) and a plethora of emerging technologies driven by (Taleb *et al.*, 2017) (Zafar *et al.*, 2017b), incidents in German and UK telecommunications giants Vodafone (BBC online UK, n.d.) and TalkTalk (BBC online UK, 2015), respectively, have become commonplace. Most recently, in the UK, Dixons Carphone suffered a huge data breach involving 5.9 million payment cards and 1.2 million personal data records (BBC online UK, 2018). According to (Solutions, 2015), exploited vulnerabilities rapidly rise every year. (Yao-Min Chen & Yanyan Yang, 2004)'s opinion is that traditional patching strategies are no longer effective. Consistent prevention against data breaches through centrally managed security processes, constrained user permissions, restricted software and controlling network traffic through firewall policies,. (Buecker, Andreas & Scott Paisley, 2008)'s opinion is that these approaches are only effective for networks and devices within a security perimeter.

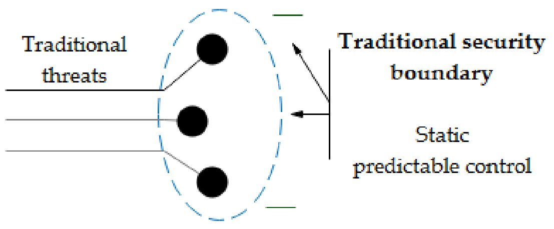


Figure 88. A traditional security view

The existing literature offers varying rationales as to why traditional security mechanisms are no longer adequate in current computing environments. One is the suggestion that traditional security systems are simply failing due to their inherent architecture. Figure 87 demonstrates this traditional notion in which traditional threats are managed and controlled within a static security boundary or perimeter.

Table 29 is a summary of some example, traditional security systems and brief descriptions of their method for protection.

Table 29. Summary of traditional security systems

System	Method
Signature Scanner	Search files for known malicious signatures. Low/High if unknown
Static behaviour analysis	Analysis executable functions for suspiciousness. High/Low
Dynamic behaviour analysis	Emulates malicious software and observes behaviour. High/Low
Integrity checker	Compare files against a cryptographically secure hash. Low/Low If kept updated
Misuse detection	Compare activity against known attacks. Low/High if unknown
Anomaly detection	Compare activity against a nominal model. High/Low If kept updated
Firewall	Enforces access rules according. Low/low – will strictly enforce rules as instructed but may be subverted

Review of cloud security countermeasures

Overwhelming efforts in the literature, both academic and industry, is dedicated to addressing security challenges in cloud computing environments. This section aims to interrogate some these works and later analyse them in subsequent subsections. The main challenges for cloud security are discussed and analysed in a contributing publication (PR 2). The current author posits that multiple perspectives on security issues imply that countermeasures are developed only for specific perspectives of security issues. A recent example is the collation of user-data security and countermeasures survey by (Basu *et al.*, 2018).

Countermeasures in academia

A framework solution proposed by (Jabir *et al.*, 2016) aims to achieve the following. First, a penetration testing component that highlights vulnerabilities within a private cloud infrastructure and an attack simulation component that attempts to discovers exploits for identified vulnerabilities. Finally, through applying countermeasures, this framework can proffer best practice protection mechanism.
(Singh, Jeong & Park, 2016) Comprehensively survey security issues affecting public and private cloud computing entities and propose requirements for security management. A 3-tier security architecture is encompassing the application, cloud-service middleware and infrastructure layer proposed towards security mitigation. According to the authors, focusing on the application level provides end-to-end security of data against client-end scripting attacks such as XSS, network data security, API security and malicious programs in general. Cloud-service middleware countermeasures against susceptibility to protocol standards attacks such as Simple Mail Transfer Protocol (SMTP), Transmission Control

Protocol (TCP), Hyper T HTTP, Dynamic Host Control Protocol (DHCP) to name a few, whose vulnerabilities are well-known. The authors suggest cryptographic solutions as well as enhanced authentication between users and middleware. Also, they suggest a Cloud Trust Protocol (CTP) between end-users and service providers (Singh, Jeong & Park, 2016).

Whereas the solutions proposed address the challenges identifies, they are limited to the public cloud context. For instance, process and regulatory challenges concerns in the perspective of outsourcing cloud services (Srinivasan *et al.*, 2012) vary among different cloud deployment models. Along the middleware solutions proposed above, (Lombardi & Pietro, 2011) propose an Advanced Cloud Protection System (ACPS) for guest Virtual Machine (VM) and middleware. Middleware represents a layer of data storage technologies, including those in cloud computing which is generally prone to unauthorised access, DDoS attacks and threats of malicious insiders (Farooq *et al.*, 2015),. Their experimental results offer some insights into future VM security in general and behavioural analysis in proactive VM protection systems. By allowing host-based asynchronous monitoring, ACPS is said to detect behaviour and performance on a guest VM while staying immune to timing attacks (Lombardi & Pietro, 2011). However, the assumption of trust is a major concern, since a compromised host will inadvertently or otherwise compromise the guest VM and possibly the infrastructure.

The authors (Joshi, Vijayan & Joshi, 2012) propose Cloud Trace Back model (CTB) for traffic detection and filtering against DDoS attacks using routing mechanisms. By appending an extra tag, Cloud Trace Back Mark (CTM) on a packet header and propagate with routing information, it possible to prevent an attack before it occurs. In the event of an attack, with CTB handling service requests between the cloud infrastructure and its peripherals, the CTB functionality can anticipate an attack before it traverses into the cloud network. Additionally, the authors advance the possibility to reveal an attack source by analysing the CTM tag. Theoretically, the CTB model achieves its design aim; however, it a real cloud environment, CTM deployment in this form comes with some operational challenges. One of such challenges is that it is quite possible for routing traffic in large networks to grow exponentially and hence introduce undesired operational overheads. As (Ahmed, Sadiq & Zolkipli, 2016) note in their paper, it is on its own often time-consuming to locate the source of a distributed attack.

A survey by (Subashini & Kavitha, 2011) focuses on security issues in service delivery models including PaaS, IaaS and SaaS. In their conclusion, the authors argue that security issues arising from all aspects of the cloud require an integrated approach to securing the cloud. The authors propose a dynamic and localised security model, which adapts to different security environments allowing users to tailor their security requirements (Subashini & Kavitha, 2011). Subashini & Kavitha's notion of unpredictability is subsumed to reduce the vulnerability factor. The notion of unpredictability is also posited by (Levitin *et al.*, 2012) who contrasts data security according to an attacker's ability to steal or destroy data.

While both notions perhaps encapsulate obfuscation and diversity as a solution, there are two apparent limitations. Foremost, diversity and heterogeneity in cloud environments mean that assert value and vulnerability factor vary from one application to another. Besides, as a common security methodology, obfuscation motivates hackers to attack the cloud (Subashini & Kavitha, 2011). As noted by (Auty, 2015), well-resourced attackers such as nation-state actors can launch a persistent attack, and hence require a different approach.

The authors; (Bendovschi, 2015) proposes security countermeasures to support business and organisations in defence against attackers from an information security perspective.

According to the author," external" countermeasures include risk awareness through organisations such as Secure Domain Foundation. "Internal" countermeasures according to these authors include continuous risk assessment, internal security posture including patching and updates, authentication methods including biometrics, access control and other security controls (preventative, detective and corrective).

In the works by (Bernsmed *et al.*, 2011), the Security SLA Management for cloud computing as a life cycle has the following key stages; publishing, negotiation, commitment, provisioning, monitoring and termination. From the commitment stage, the life-cycle approach ensures that iteration revisits a later stage. In typical cloud services with several levels of abstraction, Security SLA Management enables SLA negotiations in multiple levels (Rong, Nguyen & Jaatun, 2013a) including security, QoS, and others mitigation aspects of the lock-in effect. Since SLA is a critical component for service provision, this approach serves to address SLA-related challenges. However, the assurance of service levels is a challenge considering ongoing security threats within cloud computing. Most recently, (Carvalho *et al.*, 2017) map several solutions to open SLA issues in cloud computing.

Isolation measures on several levels mitigate risks and vulnerabilities associated with multi-tenancy (Behl & Behl, 2012). First, they propose isolation of tenants' data in IaaS (VMs, storage, processing, memory, and access paths networks). Also, the authors propose isolation of tenants' data in PaaS API calls, Operating System (OS) level calls and running services. These authors further propose isolation in SaaS tenants' transactions on the same instance and their respective data. In conclusion, the authors suggest relevant isolation policies and security measures to be implemented by the CSP.

Along similar lines, (Rao & Selvamani, 2015) demonstrate that data segregation and protection and data leakage protection, are some of the major data security challenges in cloud computing environments. Heterogeneous data-centric security controls such as encryption is argued as adequate measures to control the access of data. The authors propose a data security model encompassing authentication, encryption, and integrity, recovery and user protection. They suggest the integration of identity-based cryptography and RSA Signature enhance data integrity. In addition, a network-based intrusion detection system is suggested as adequate for real-time presentation of against threats. In the former conception, there is limited detail on how to achieve isolation in each domain. Whereas the latter provides greater details, a recent study by (Shi, Chen & Li, 2018) shows that signature-based security is not suited to address complex multi-stage attacks whose signatures are often unknown.

A trust framework is developed to enhance efficiency in capturing a generic set of parameters required for establishing trust and manage evolving trust and interaction/sharing requirements (Mell & Grance, 2011). The cloud's policy integration tasks should be able to address challenges such as semantic heterogeneity, secure interoperability, and policy evolution management. Furthermore, customers' behaviours can evolve rapidly, thereby affecting established trust values. This suggests a need for an integrated, trust-based, secure interoperation framework that helps establish, negotiate, and maintain trust to adaptively support policy integration (Mell & Grance, 2011).

A Trusted Third Party (TTP) authored by (Demchenko *et al.*, 2011) addresses trust in a distributed system by allowing other TTP domains to span different geographical locations. However, a point of contention in implementations of trust in the cloud using TTP rests in inherent contradictions and interpretations of trust, from one perspective to the next. For instance, some argue that trust relations between the server and the client should be established dynamically (Demchenko *et al.*, 2011). From a CC and CSP perspective, trust relations are the acknowledgement of risk factor by the replying party

(Zissis & Lekkas, 2012). As the authors contend, trust is not a new research topic in computer science, spanning areas as diverse as security and access control in computer networks, reliability in distributed systems, game theory and agent systems, and policies for decision making under uncertainty (Zissis & Lekkas, 2012). However, a shift from traditionally managed infrastructures where security controls exist within contained environments makes trust solutions in cloud computing contentious (Zissis & Lekkas, 2012).

The author is Ryan (2013) propose an incident-based solution to mitigate access to customer data. The proposed solution relies on the assumption that good practice entails a customer trusting the CPS and their employees, whereas legislation and contracts act as deterrents for unlawful disclosure (Ryan, 2013). Ryan fosters a debate for fully homomorphic encryption where in theory, a cloud customer maintains the privacy, integrity and confidentiality of their data by encrypting it, while on the other hand, the cloud service provider can perform processing operations on the same data but without access to the data itself. However, this practice is not viable in the cloud, based upon the mechanisms' efficiency and viability limitations. Moreover, the notion that the technical and service maturity of a CSP is an essential assurance that CSP will not disclose user information is both debatable and an inadequate countermeasure.

(Mazur *et al.*, 2011) Posit a three-pronged approach to managing security in the cloud. In this conception, the authors argue cloud security as a balance between the potential impact of cloud computing on the competitive business edge, and the evaluation of business-critical security components. Whereas the authors view cloud security from a business perspective, this approach not a generalisable approach for cloud security countermeasures.

Furthermore, their classification lacks detail on which security solutions or techniques are at each management level. According to the authors, a trusted monitor audits a cloud server to provide proof of compliance with the data owner. The Integrity of the trusted monitor is itself ensured by running in isolation, the monitor bootstrap along with the Operating system and Applications. Additionally, the authors argue for the implementation of cryptographic mechanisms such as homomorphic encryption, searchable encryption, that allow CSP to process ciphertext. Although in theory, techniques (Mazur *et al.*, 2011) ensure integrity and compliance in computing, in practice, these techniques have so far proven to be usable in the cloud.

Authors in (Mazur *et al.*, 2011) posit a compelling argument for a self-monitoring defensive mechanism; a contract of intelligent agents that collect data within the cloud, including code execution. Also, they suggest computational intelligence that uses ontology-based fusion engines for situational analysis and brokering, synonymous with how human breath without consciously having to think through every breath (Mazur *et al.*, 2011). The authors posit the need for ontologies, not only as a useful tool for identifying malware, but also individual components that perform specific functions. For instance, a generative ontology for future malware signatures which will be useful in future detections using comparative analysis. (Rao, J.R., Chari, S.N., Pendarakis, D., Sailer, R., Stoecklin, M.P., Teiken, W. and Wespi, 2016) It is Security 3600, is an operational model developed to protect services, enterprise and data in cloud computing environments. Through passive monitoring of network traffic and near-field monitoring of disk images, VMs and the hypervisor, Security 3600 can provide comprehensive end-to-end contextual view of the security of an enterprise.

(Sabahi, 2011) Proposes formal procedures to mitigate cloud security challenges. According to the authors, since traditional mechanisms need to be changed to work efficiently in the cloud, solutions should be viewed in terms of access control and incident

response. However, the proposals presented here require deeper details as some aspects remain arguable. For instance, access control in a SaaS model is restricted to their software as the customer is only responsible for managing how they authenticate to the cloud service and what level privilege. In the PaaS where the CSP retains control to the network, server and applications, Sabahi argues that the cloud customer retains and manages access rights to the applications they run. Likewise, in the IaaS scenario, the cloud customer is responsible for access control management of the virtual platforms they run, including virtual servers, virtual network and virtual storage (Sabahi, 2011).
(Bays *et al.*, 2015) Whose work investigated countermeasures in virtual networks observe an unbalanced correlation between security threats and security countermeasures in the available literature. For instance, the authors note disruption and disclosure (both security threats) and availability and confidentiality (both security countermeasures) as dominant across most of the literature. The author also notes that across the literature, most publications propose a single countermeasure for a single threat (Bays *et al.*, 2015). Is also clear as demonstrated in these authors' finding, that no publication addressed the issue of nonrepudiation. Nonrepudiation is an indispensable component for user behaviour credibility attribution in the forensic investigation process. As alluded to by (Tian *et al.*, 2019), nonrepudiation is particularly critical for cloud storage scenarios where verification of the integrity of logs files before the investigation is important. The table below summarises the distribution of countermeasures according to these authors. In the following section, the review of industry countermeasures follow criteria set out by (Cser, 2016); restricted to leading vendors dominating cloud security world markets.

Countermeasures in industry

CloudLock (CloudLock, 2015b) is lauded as the only complete security platform capable of protecting assets within the cloud. Content monitoring service employs heuristic algorithms to monitor and determine the sensitivity of data across SaaS, PaaS and IaaS environments. Behaviour analytics detects anomalies in user activity and usage, based on thresholds prescribed in centralised policies. In addition, CloudLock uses a policy automation service for classifying data based on its unique attributes, automated notifications, and remediation management capabilities to end-users. Central auditing provides evidence for compliance with internal policies, by recording an audit trail of all actions. In addition, CloudLock uses a central incident management service to investigate, prioritise, and "track to closure" security incidents. To leverage AES-256 encryption and key management, CloudLock argues that automating encryption will enable users to secure their sensitive data effectively.
On the other hand, they posit their security analytics service to leverage security compliance with both internal and external auditors. Two clear limitations with CloudLock are centred on its centralised nature. (Moradi, 2016) Shows that centralised management introduces a single point of failure and fails to scale in distributed systems. In additions, like traditional systems, CloudLock's firewall systems only discover and controls security risk in deployed black and whitelists which require knowledge of the risk before deployment. Uncertainty due to the sophistication of attacks limits the amount of knowledge about an attacker, countermeasures possess.
Blue Coat's Cloud Data Protection Gateway (Bluecoat, 2015b) protects data in transit, data at rest and data in use. According to Blue Coat, their solution support searching, sorting, or reporting of strongly encrypted or tokenised data. Protecting data in transit is aimed at mitigating eavesdropping by third parties. Cryptographic methods such as Secure Socket Layer (SSL) and Transport Layer Security (TLS) are some common techniques

implemented to protect data in transit. Protecting data at rest on the other end is about preventing the access of persistent data, ensuring that data maintains the same form or format, as a file or in a database. Blue Coat argue the Heartbleed vulnerability in which usernames and passwords were leaked through exploiting vulnerabilities in OpenSSL necessitates the need to protect data being processed in memory. Elastica CloudSOC (Blue Coat and Symantec, 2015) produced by Blue Coat and Symantec to audit, detect, protect, and investigate security incidents in cloud environments. According to Blue Coat, their audit module facilitates the evaluation of cloud services adopted by an organisation, identifying the types of devices they are accessed with, as well as their business readiness. In addition, they posit their detection module to utilise machine learning and data science to detect threats, activities, and users within an organisation's cloud network. The investigation module is useful after an incident occurs, ensuring that cloud-based data is available and accessible for an organisation's requirements analysis, including legal, compliance, and human resources.

Microsoft's Cloud App Security (Microsoft Mobility Management, 2016) ensures that cloud users can maintain visibility, control, and protection of their cloud application data. According to Microsoft, this is primarily due to the failure by traditional security solutions such as firewalls and intrusion prevention systems (IPSs), "do not offer visibility into transactions that are unique to each application, and traffic off-premises". Thus, Microsoft Cloud App Security is aimed for SaaS offerings to enhance auditing, visibility, and the amount of control CC have on-premises, to cloud applications. Microsoft Cloud Apps security is argued to provide risk scoring and assessment of all identified applications within a CC network, including all devices. In addition to network and device discovery, this solution provides granular controls and policies for data sharing. Finally, Cloud App Security's threat protection component adds intelligence using machine-learning techniques to identify high-risk security incidents and abnormal behaviour detection.

IBM's security paradigm, developed in response to the changing security threat domain, including botnets, advanced persistent threats and polymorphic malware (Gulla, 2011b). In their work, IBM postulates the notion of a quick response as critical in dealing with data breaches in the cloud. IBM's security paradigm consists of the following: Fine-grain contextual security, provenance: Honey pot.

Alert Logic's response plan is premised on a case study of Shellshock vulnerability in 2014, culminated into the development of their 6-step approach. According to Alert Logic, their outline is a template for a vulnerability response action plan (Alert Logic, 2015, 2016). Comprehension: Exposure: Communication: Security Content Creation: Patching: Lessons Learned: BitGlass (Bitglass, 2014b) argues their solution to provide end-to-end security for any device on the cloud network, through access controls, limit sharing, and to mitigate data leakage. The authors suggest that a single-pane view of applications in the network, enables organisations to gain instant visibility over their cloud data (data at rest), detect abnormal user and usage behaviours, and can anticipate emerging threats. Using a cloud DLP engine, BitGlass protects data in transit using watermarking, DRM, and blocking techniques, as well as create custom security policies. BitGlass suggest it is possible to identify managed and unmanaged devices, and limit device access based on contextual information, e.g. location, device type and network.

According to Citrix's white paper (Citrix, 2015), their product is highly scalable and designed to mitigate against traditional and modern DoS attacks, and application-layer attacks on web properties. Citrix postulates that solutions should prioritise establishing how well resourced (organisation, funding) an adversary is, as opposed to the exploit they are going to use. Citrix® NetScaler Application Delivering Control (ADC) exhibits important capabilities that ensure security in web applications on at least two fronts. On

one end, load balancing algorithms that dynamically route traffic during outages or failures ensure the availability of services. Additionally, health monitoring components proactively monitor services for efficient operation and performance. According to the authors, NetScaler is a useful disaster recovery tool; Global Server Load Balancing (GSLB) ensures availability of services by routing traffic to other datacentres during a disaster. On the other end, a combination of a reactive firewall (NetScaler App Firewall), protocol defences, and NetScaler's compliance with other third-party products such as the Payment Card Industry Data Security Standard (PCI- DSS) prevents application-layer threats.
CA Technologies' automated identity management lifecycle ensures efficient management of identities throughout their lifecycle, including policy and identity auditing thereby supporting compliance concerns ('Amiri, 2009). By logically layering and separating components, CA's lifecycle ensures effective disaster recovery and high availability. Furthermore, Advanced Encryption Standard (AES), Federal Information Processing Standard (FIPS) 140-2, and information standard protocols such as HTTPS and LDAPS, ensure that data is secured in accordance to industry-standard best practices. Additionally, correlation capabilities with real-world data improve security visibility.
The Data Loss Prevention component utilises data mining tools to optimise policy creation through the effective classification of data at rest. Similarly, McAfee Email and Web Protection enable bidirectional protection against inbound threats, data loss prevention in emails, and secure communication for web application through proactive reputation and intent-based protection.

Appendix C – Pi-CCSF simulator code

```
import random
import math
from operator import add
import matplotlib.pyplot as plt
from creation_default_variables import *
import copy
import csv

random.seed(101);
#_actionsI = [ random.randint(1,NbrAction) for i in range(Timestamp)]
class Static_simulation:
 def __init__(self, Timestamp, Nbr_VM, actionsI= None, NbrAction=5):
 # print("The timestamp is of: "+ str(Timestamp))
# print("The Nbr_VM is of: "+ str(Nbr_VM))
 self._V_VMs = [ [-100 for i in range(Timestamp)] for i in range(Nbr_VM) ]
 self._A_VMs = [ [-100 for i in range(Timestamp)] for i in range(Nbr_VM) ]
 self._S_VMs = [ [-100 for i in range(Timestamp)] for i in range(Nbr_VM) ]
 self._C_VMs = [ [-100 for i in range(Timestamp)] for i in range(Nbr_VM) ]
 self._A_eval = [ [-100 for i in range(Timestamp)] for i in range(Nbr_VM) ]
 self._C_eval = [ [-100 for i in range(Timestamp)] for i in range(Nbr_VM) ]
 self._phis = [ [-100 for i in range(Timestamp)] for i in range(Nbr_VM) ]
 ##### Creation of the vectors to store random values ######
 self._alphas = [ [-100 for i in range(Timestamp)] for i in range(Nbr_VM) ]
 self._Betas = [ random.uniform(-1, 1) for i in range(Nbr_VM) ]
 self._Gammas = [ random.uniform(-1, 1) for i in range(Nbr_VM) ]
 self._Ethas = [ random.uniform(-1, 1) for i in range(Nbr_VM) ]
 if actionsI == None:
 self._actionsI = [random.randint(1,NbrAction) for i in range(Timestamp)]
 else:
 self._actionsI = actionsI
 self._actions = [[ random.randint(1,NbrAction) for i in range(Timestamp)] for i in range(Nbr_VM) ]
 self._Memory = None
 self._Timestamp = Timestamp
 self._Nbr_VM = Nbr_VM
 self._NbrActions = NbrAction
 # initialise the first value for each VM
 for i in range(Nbr_VM):
 invalid = True
  while invalid:
 # First value generated randomly
 self._V_VMs[i][0] = random.uniform(0, 1);
 self._A_VMs[i][0] = random.uniform(0, 1);
# print(self._V_VMs[i][0])
# print(self._A_VMs[i][0])
# print(self._phis[i][0])
  self._phis[i][0] = self._V_VMs[i][0]+ self._A_VMs[i][0];
 self._alphas[i][0] = self._V_VMs[i][0]+self._A_VMs[i][0]
 # print(1- alphas[i][0])
 # To simplify the equation : C_VM[0] =0
 self._S_VMs[i][0] = 1 - self._alphas[i][0];
 self._C_VMs[i][0] = 0;
 if self._S_VMs[i][0]>0:
  invalid = False
 def get_Access_VM_Value(self, VM_id):
return(self._Betas[VM_id],self._Gammas[VM_id], self._Ethas[VM_id], self._V_VMs[VM_id], self._A_VMs[
VM_id], self._C_VMs[VM_id], self._S_VMs[VM_id], self._alphas[VM_id])
 def replaceNAN(self, list, time_id):
 max_id = len(list)
 if (time_id>0 and time_id< max_id):
 if list[time_id] < 0:
 list[time_id] = 0
 if time_id+1 < max_id:
  list[time_id+1] = 0
 else:
 if list[time_id] > 1:
  list[time_id] = 1
  if time_id+1 < max_id:
  list[time_id+1] = 1
 def V_VM_AVM_iteration(self, time_indice, VM_id):
current_Beta, current_Gamma, current_Etha, V_VM, A_VM, C_VM, S_VM, alpha = self.get_Access_VM_
Value(VM_id)
V_VM[time_indice+1] = V_VM[time_indice] - current_Beta*V_VM[time_indice]*A_VM[time_indice] + cur
rent_Gamma*A_VM[time_indice]
A_VM[time_indice+1] = A_VM[time_indice] + current_Beta*V_VM[time_indice]*A_VM[time_indice] - cur
rent_Gamma*A_VM[time_indice]
 self.replaceNAN(V_VM,time_indice )
 # print("A_VM:")
 self.replaceNAN(A_VM,time_indice )
 alpha[time_indice+1] = V_VM[time_indice+1] + A_VM[time_indice+1]
 self._phis[VM_id][time_indice] = self._V_VMs[VM_id][time_indice]+self._A_VMs[VM_id][time_indice];
self._A_eval[VM_id][time_indice] = current_Beta*(self._phis[VM_id][time_indice] - self._A_VMs[VM_id][ti
me_indice])*self._A_VMs[VM_id][time_indice] - current_Gamma*self._A_VMs[VM_id][time_indice]
self._C_eval[VM_id][time_indice] = self._actions[VM_id][time_indice] *((1 - self._phis[VM_id][time_indice]
- self._C_VMs[VM_id][time_indice] ) *self._A_VMs[VM_id][time_indice] )/self._actionsI[time_indice] - curr
ent_Etha*self._C_VMs[VM_id][time_indice]
 def S_VM C_VM_iteration(self, time_indice, VM_id ):
current_Beta, current_Gamma, current_Etha, V_VM, A_VM, C_VM, S_VM, alpha = self.get_Access_VM_
Value(VM_id)
 #Si the VM attack is dead
 if V_VM[time_indice] <= 0:
 S_VM[time_indice+1] = 0
 S_VM[time_indice] = 0
  else:
 if V_VM[time_indice] == 1:
 S_VM[time_indice+1] =1
 S_VM[time_indice] = 1
 else:
S_VM[time_indice+1] = S_VM[time_indice] - (alpha[time_indice]*S_VM[time_indice]*A_VM[time_indice])/self._actionsI[time_indice] + current_Etha*C_VM[time_indice]
  #If the Attacking VM is shutting down
 if A_VM[time_indice] <= 0:
 C_VM[time_indice+1] =0
 C_VM[time_indice] =0
  else:
 if A_VM[time_indice] == 1:
 C_VM[time_indice+1] = 1
 C_VM[time_indice] = 1
 else:
C_VM[time_indice+1] = C_VM[time_indice] + (alpha[time_indice]*S_VM[time_indice]*A_VM[time_indice])/self._actionsI[time_indice] - current_Etha*C_VM[time_indice]
 self.replaceNAN(S_VM, time_indice)
 self.replaceNAN(C_VM, time_indice)
  def one_iteration_WithMem(self, time_indice, VM_id):
 self.V_VM A_VM_iteration(time_indice, VM_id)
 if self._V_VMs[VM_id][time_indice-1] >0: #If the VM was alive
# print(time_indice)
 if time_indice >= int(self._Memory._nbrAtomicAction-2):# If I have performed enough action to saved in memory
# print("here")
 #Get the last actions to compute the key in memory
 lastActionKey = []
 for a in range(self._Memory._nbrAtomicAction-1):
  actionNb = self._Memory._nbrAtomicAction - a
  lastActionKey.append(self._actions[VM_id][time_indice-actionNb])
 # get the best action if exist, random value in another case
 lastActionKey = tuple(lastActionKey)
 newAction = self._Memory.getBestAction(lastActionKey)
 lastActionKey= list(lastActionKey)
 lastActionKey.append(newAction)
 lastActionKey = tuple(lastActionKey)
newGain = self._V_VMs[VM_id][time_indice+1] - self._V_VMs[VM_id][time_indice- (self._Memory._nbrAtomicAction-2 )]
 if lastActionKey in self._Memory._Memory.keys():
 # if the action is known, add the score
self._Memory._Memory[lastActionKey] = (self._Memory._Memory[lastActionKey][0]+newGain, self._Memory._Memory[lastActionKey][1]+1 )
 else: # otherwise, add the new key
  self._Memory._Memory[lastActionKey] = (newGain, 1)
# print("action added")
 self.SVM_CVM_iteration(time_indice, VM_id)
 def one_iteration_WithoutMem(self, time_indice, VM_id):
 self.VVM_AVM_iteration(time_indice, VM_id)
 self.SVM_CVM_iteration(time_indice, VM_id )
 def run_simulation(self):
 return (self.run_simulationFromInit())
 def run_simulationFromInit(self):
 #### compute VM evolution with memory access
 for time_indice in range(self._Timestamp-1 ): #Delay_betweens_share,
 #print(time_indice)
 for VM_id in range(self._Nbr_VM):
 #print(self._Memory)
 if(self._Memory != None):
  if(self._Memory._nbrAtomicAction > 0):
  #print("use memory")
  self.one_iteration_WithMem(time_indice, VM_id)
  #self._Memory.savedIn("test.json")
  else:
  self.one_iteration_WithoutMem(time_indice, VM_id)
 def run_BestResult(self):
```

```
for VM_id in range(self._Nbr_VM):
self._actions[VM_id] = self._actionsI
for time_indice in range(self._Timestamp-1 ):
self.one_iteration_WithoutMem(time_indice, VM_id)
def setMemory(self, Memory):
self._Memory = Memory
```

Model memory.py

```
import json
import random
class Memory:
 def __init__(self, path="", nbrSaved=3, nbrActions = 5):
 self._Memory = {}
 self._nbrAtomicAction = nbrSaved
 self._nbrActions = nbrActions
 if path != "":
 with open(path) as currentFile:
 data = json.load(currentFile)
 currentFile.close()
 for key in data.keys():
 newkey = copy.deepcopy(key)
 newkey = newkey.replace('(', '').replace(')', '')
 if len(newkey)>= 3:
  newkey = newkey.split(",")
  newkey = list(map(int,newkey))
 else:
  newkey = newkey.replace(",", "")
  newkey = [int(newkey)]
 self._Memory[tuple(newkey)] = tuple( [ sum(data[key]) , len(data[key])])
 def savedIn(self, filename):
 with open(filename, 'w') as file:
 file.write("{ \"nbrActionsSaved\": \" "+str(self._nbrAtomicAction)+"\",\n\"Data\": ")
 file.write(json.dumps({str(k): str(v) for k, v in self._Memory.items()}))
 file.write("}")
 file.close()
 def chargeMemory(self, filename):
 #print("charge file TTOTO")
 with open(filename) as currentFile:
 data = json.loads(currentFile.read())
 #print("after open the file")
 #self._nbrSaved = int(data['nbrActionsSaved'])
 for key in data['Data']:
 value = data['Data'][key]
 key = key[1:-1]
 key= key.replace(' ', '')
 key = key.split(',')
 if(self._nbrAtomicAction == 1 ):
 key = int(key[0])
 else:
 #key.split(',')
 for elem in key:
  if type(elem) == int:
  #print(elem)
  elem = int(elem)
 key = tuple(key)
  value = value.replace('(', '').replace(')', '')
 value = value.split(",")
  #print("key : "+str(key))
 #print("value : "+str(value))
  self._Memory[key] = tuple( [float(value[0]) , int(value[1])] )
 currentFile.close()
  def getMoyOf(self, key):
 return( self._Memory[key][0]/self._Memory[key][1])
 def getBestAction(self, pastActions):
 listActions = list(self._Memory.keys())
```

```
nextList = []
for i in range(len(pastActions)):
for j in range(len(listActions)):
if pastActions[i] == listActions[j][i]:
 nextList.append(listActions[j])
listActions = nextList
nextList = []
if len(listActions)< self._nbrActions:
return random.randint(1,self._nbrActions)
bestScore = self.getMoyOf(listActions[0])
bestAction = listActions[0][-1]
for key in listActions:
newScore = self.getMoyOf(key)
if bestScore < newScore:
bestScore = newScore
bestAction = key[-1]
if bestScore > 0:
return bestAction
else:
return random.randint(1,self._nbrActions)
```

Controller.py

```
import copy
from matplotlib import rcParams
import os, os.path
from model_Memory import Memory
from model_StandardSimulation import Static_simulation
import matplotlib.pyplot as plt

rcParams['font.family'] = 'Palatino Linotype'
choiceDone = False
def generate_memory(numberWishVM, Timestamp, Nbr_VM, NbrAction, TimeRaisedSearch, rateVitalityWished, NbrAction_historic ):
 counter = 0
 memory = Memory(nbrSaved=NbrAction_historic)
 Actions_saved = {}
 while counter < numberWishVM:
 simulation = Static_simulation(Timestamp, Nbr_VM)
 simulation.run_simulation()
 V_VMs = simulation._V_VMs
 S_VMs = simulation._S_VMs
 actions = simulation._actions
 # Search in the Vulnerable VMs the one with the the hiest survivability
 indice_VM =0
 # 1. for each Vulnerable VM
 while indice_VM < (len(V_VMs)) and counter< numberWishVM:
 #print(NbrAction_historic)
 #If the VM survived and nbr actions to save >0
 if V_VMs[indice_VM][-1] >0 and NbrAction_historic>0 :
  #If at a time, her vitaliy raise rapidly
 raiseSpeed = False
  t= NbrAction_historic
 while (not raiseSpeed) and (t<Timestamp):
  if (S_VMs[indice_VM][t] - S_VMs[indice_VM][t-NbrAction_historic]) > rateVitalityWished:
  #saved the actions performed
  key = []
  for i in range(TimeRaisedSearch):
  key.append(actions[indice_VM][t-(TimeRaisedSearch-i)])
  key = tuple(key)
  value = (S_VMs[indice_VM][t] - S_VMs[indice_VM][t-NbrAction_historic])
 # print(str(S_VMs_saved[indice][t])+" - "+str( S_VMs_saved[indice][t-TimeRaisedSearch])+" = "+str(value))
   if key not in Actions_saved.keys():
  memory._Memory[key] = (value, 1)
  else:
```

```
  memory._Memory[key] = (memory._Memory[key][0]+ value, memory._Memory[key][1]+1)
   #count in result
  raiseSpeed= True
  counter+=1
  t+=1
 indice_VM+=1
 return memory
def summariseResult(Simulation, evolution_nbrSurviveV, evolution_nbrSurviveA, evolution_moyVitalityV, evolution_moyVitalityA ):
 nbrSurvivedV = sum([1 for i in range(Simulation._Nbr_VM) if Simulation._V_VMs[i][-1]>0])/Simulation._Nbr_VM
 nbrSurvived = sum([1 for i in range(Simulation._Nbr_VM) if Simulation._A_VMs[i][-1]>0])/Simulation._Nbr_VM
 moySurvivabilityV = sum([Simulation._V_VMs[i][-1] for i in range(Simulation._Nbr_VM)])/Simulation._Nbr_VM
 moySurvivabilityA = sum([Simulation._A_VMs[i][-1] for i in range(Simulation._Nbr_VM)])/Simulation._Nbr_VM
 evolution_nbrSurviveV.append(nbrSurvivedV)
 evolution_nbrSurviveA.append(nbrSurvivedA)
 evolution_moyVitalityV.append(moySurvivabilityV)
 evolution_moyVitalityA.append(moySurvivabilityA)
def saving(currentplot, prefix):
 directory = "./img"
 try:
 os.makedirs(directory)
 except FileExistsError:
 pass
 nbr_file_in_directory = len([name for name in os.listdir(directory) if os.path.isfile(os.path.join(directory, name))])
 plt.savefig("./img/"+prefix+str(nbr_file_in_directory)+".png")
def plotSummaryOne(evolution_nbrSurviveV, evolution_nbrSurviveA, evolution_moyVitalityV, evolution_moyVitalityA, nameFile, champValue):
 fig1, axes = plt.subplots(nrows = 2, ncols=2)
 plt.figure(figsize=(800/96, 800/96), dpi=96)
 ax1 = plt.subplot(2, 1, 1)
 ax1.plot(evolution_nbrSurviveV, color='b', label=champValue[4])
 ax1.plot(evolution_nbrSurviveA, color='r', label=champValue[5])
 ax1.legend(loc="upper right")
 ax1.set_title(champValue[0])
 plt.xlabel(champValue[1])
 plt.ylabel(champValue[2])
 ax2 = plt.subplot(2, 1, 2)
 ax2.plot(evolution_moyVitalityV, color='b', label=champValue[4])
 ax2.plot(evolution_moyVitalityA, color='r', label=champValue[5])
 ax2.legend(loc="upper right")
 plt.ylabel(champValue[3])
 saveimg(plt, nameFile )
 #plt.show()
def plotSummaryOneliner(evolution_nbrSurviveV, evolution_nbrSurviveA, evolution_moyVitalityV, evolution_moyVitalityA, nameFile, champValue):
 evolution_nbrSurviveV2 = []
 evolution_nbrSurviveA2 = []
 evolution_moyVitalityV2 = []
 evolution_moyVitalityA2 = []
 for i in range(len(evolution_nbrSurviveV)//100):

 print(evolution_nbrSurviveV2)
plotSumaryOne(evolution_nbrSurviveV2, evolution_nbrSurviveA2, evolution_moyVitalityV2, evolution_moyVitalityA2, nameFile, champValue)
def plotSummaryTwo(evolution_nbrSurviveV,evolution_nbrSurviveV2, evolution_nbrSurviveA,evolution_nbrSurviveA2, evolution_moyVitalityV,evolution_moyVitalityV2, evolution_moyVitalityA, evolution_moyVitalityA2, nameFile, champValue):
 fig1, axes = plt.subplots(nrows = 2, ncols=2)
 plt.figure(figsize=(800/96, 800/96), dpi=96)
 ax1 = plt.subplot(2, 1, 1)
 ax1.plot(evolution_nbrSurviveV, color='b', label=champValue[1][5])
 ax1.plot(evolution_nbrSurviveV2, color='k', label= champValue[2][5])
 ax1.plot(evolution_nbrSurviveA, color='r', label=champValue[1][6])
 ax1.plot(evolution_nbrSurviveA2, color='m', label=champValue[2][6])
 ax1.legend(loc="upper right")
 ax1.set_title(champValue[0])
 plt.xlabel(champValue[1][1])
 plt.ylabel(champValue[1][2])
 ax2 = plt.subplot(2, 1, 2)
 ax2.plot(evolution_moyVitalityV, color='b', label=champValue[1][5])
 ax2.plot(evolution_moyVitalityV2, color='k', label=champValue[2][5])
 ax2.plot(evolution_moyVitalityA, color='r', label=champValue[1][6])
 ax2.plot(evolution_moyVitalityA2, color='m', label=champValue[2][6])
 ax2.legend(loc="upper right")
 plt.xlabel(champValue[2][1])
 plt.ylabel(champValue[1][3])
 saveimg(plt, nameFile)
 #plt.show()

def plotSumaryTwoliner(evolution_nbrSurviveV,evolution_nbrSurviveV2, evolution_nbrSurviveA,evolution_nbrSurviveA2, evolution_moyVitalityV,evolution_moyVitalityV2, evolution_moyVitalityA, evolution_moyVitalityA2, nameFile, champValue):
 evolution_nbrSurviveV_2 = []
 evolution_nbrSurviveA_2 = []
 evolution_moyVitalityV_2 = []
 evolution_moyVitalityA_2 = []
 evolution_nbrSurviveV2_2 = []
 evolution_nbrSurviveA2_2 = []
 evolution_moyVitalityV2_2 = []
 evolution_moyVitalityA2_2 = []
 for i in range(len(evolution_nbrSurviveV)//100):

def run(valueInt, Timestamp, NbrTimestamp, Nbr_VM, NbrAction, NbrAction_historic, Delay_betweens_share,numberWishVM,rateVitalityWished, nameMemoryFile,nameMemoryFile2):
 from model_Memory import Memory
 #print("-----------------")
 if valueInt == 1:
 #print("Begin running simulation")
 evolution_nbrSurviveV = []
 evolution_nbrSurviveA = []
 evolution_moyVitalityV = []
 evolution_moyVitalityA = []
 memory= Memory(nbrSaved = NbrAction_historic)
 # print("The NbrTimestamp is of: "+ str(NbrTimestamp))
# print("The timestamp is of: "+ str(Timestamp))
# print("The Nbr_VM is of: "+ str(Nbr_VM))
 try:
 #print("try begin")
 memory.chargeMemory(nameMemoryFile)
 #print("after charge")
 #print("truc: \n"+str(memory._Memory))
 except:
 pass
 for i in range(NbrTimestamp):
 Simulation = Static_simulation(Timestamp=Timestamp, Nbr_VM=Nbr_VM)
 Simulation._Memory = memory
 Simulation.run_simulation()
summariseResult(Simulation, evolution_nbrSurviveV, evolution_nbrSurviveA, evolution_moyVitalityV, evolution_moyVitalityA )
plotSumaryOne(evolution_nbrSurviveV, evolution_nbrSurviveA, evolution_moyVitalityV, evolution_moyVitalityA)
 #print("Ending running simulation")
 else:
 if valueInt == 2:
 #print("Begin generate memory")
```

```
memory = generate_memory(numberWishVM, Timestamp, Nbr_VM, NbrAction, NbrAction_historic, rate
VitalityWished, NbrAction_historic )
 memory.savedIn(nameMemoryFile)
 #print("Memory saved")
  else:
 #print("begin comparing simulation")
 evolution_nbrSurviveV = []
 evolution_nbrSurviveA = []
 evolution_moyVitalityV = []
 evolution_moyVitalityA = []
  evolution_nbrSurviveV2 = []
 evolution_nbrSurviveA2 = []
 evolution_moyVitalityV2 = []
 evolution_moyVitalityA2 = []
 memory = Memory(nbrSaved= NbrAction_historic)
 try:
 #print(nameMemoryFile)
 Memory = chargeMemory(nameMemoryFile)
 except:
 pass
 Memory2 = Memory(nbrSaved=NbrAction_historic)
 try:
 Memory2 = chargeMemory(nameMemoryFile2)
 except:
 pass
# print(NbrAction_historic)
# print(NbrAction_historic2)
  for i in range(NbrTimestamp):
 #print("""""""""""""""""""""""")
 memory = Memory(nbrSaved= NbrAction)
 Simulation = Static_simulation(Timestamp, Nbr_VM)
 Simulation.setMemory(memory)
  Simulation2 = copy.deepcopy(Simulation)
# run_simulationFromInit(Timestamp,Nbr_VM, NbrAction, NbrAction_historic, Memory, V_VMs,A_VMs
,S_VMs, C_VMs,alphas,Betas, Gammas, Ethas, actions)
 Simulation.run_simulationFromInit()
 Simulation2.run_simulationFromInit()
summariseResult(Simulation, evolution_nbrSurviveV, evolution_nbrSurviveA, evolution_moyVitalityV, evol
ution_moyVitalityA )
sumerizeResult(Simulation2, evolution_nbrSurviveV2, evolution_nbrSurviveA2, evolution_moyVitalityV2, ev
olution_moyVitalityA2 )
  #print(Simulation._actions == Simulation._actions)
plotSumaryTwo(evolution_nbrSurviveV,evolution_nbrSurviveV2, evolution_nbrSurviveA,evolution_nbrSur
viveA2, evolution_moyVitalityV,evolution_moyVitalityV2, evolution_moyVitalityA, evolution_moyVitalityA
2)
  #print("ending comparing simulation")
```

Default variables

```
import json
from pprint import pprint
 # get the default variables from the files
with open('default_variable.json') as file:
 data = json.load(file)
file.close()
Timestamp = data['timestamp']
Nbr_timestamp = data['Nbr_timestamp']
Nbr_VM = data['Nbr_VM']
NbrAction = data['NbrAction']
NbrAction_historic= data['NbrAction_historic']
NbrAction_historic2 = data['NbrAction_historic2']
Delay_betweens_share = data['Delay_betweens_share']
```

```
numberWishVM = data['Nbr_VM_For_Memory']
rateVitalityWished = data['Min_Vitality_raised']
Memory_fileName = data['Memory_fileName']
Memory_fileName2 = data['Memory_fileName2']
```

Appendix D - Simulation results

Table 30. Sample results for Group 7 -VM 491 and group 0 -VM 67

group 7 gen491				group 0 generation 67			
V_VM	A_VM	C_VM	S_VM	V_VM	A_VM	C_VM	S_VM
0.001	0.995	0.000	0.003	0.025	0.384	0.000	0.590
0.853	0.144	0.002	0.002	0.366	0.044	0.093	0.497
0.968	0.029	0.002	0.001	0.408	0.002	0.009	0.582
0.991	0.006	0.002	0.001	0.410	0.000	0.000	0.590
0.996	0.001	0.002	0.001	0.410	0.000	0.000	0.590
0.997	0.000	0.003	0.001	0.410	0.000	0.000	0.590
0.997	0.000	0.003	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.003	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.004	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.004	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.005	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.005	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.006	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.006	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.007	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.008	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.009	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.010	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.011	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.013	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.014	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.016	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.018	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.020	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.022	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.025	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.028	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.031	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.034	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.039	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.043	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.048	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.054	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.060	0.000	0.410	0.000	0.000	0.590

0.997	0.000	0.068	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.076	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.085	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.095	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.106	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.119	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.133	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.148	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.166	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.186	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.208	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.291	0.000	0.410	0.000	0.000	0.590
0.997	0.000	0.326	0.000	0.410	0.000	0.000	0.590

Table 31. Sample results for best Action synthesis

Attacking VMs	0.334	0.321	0.357	0.389	0.414	0.432	0.445	0.454	0.460	0.465	0.469	0.472	0.474	0.481	0.483	0.484	0.485	0.485	0.486
Corrupted VMs	0.000	0.013	0.020	0.040	0.049	0.063	0.086	0.111	0.133	0.152	0.168	0.181	0.193	0.218	0.223	0.225	0.228	0.230	0.232
Survival VMs	0.335	0.322	0.316	0.295	0.287	0.279	0.271	0.264	0.258	0.253	0.249	0.246	0.242	0.235	0.233	0.234	0.233	0.233	0.232
Vulnerable VMs	0.331	0.364	0.357	0.348	0.343	0.339	0.336	0.334	0.333	0.332	0.331	0.330	0.330	0.329	0.328	0.328	0.328	0.328	0.328
Attacking VMs	0.333	0.322	0.358	0.391	0.414	0.430	0.442	0.450	0.456	0.460	0.464	0.467	0.469	0.476	0.477	0.478	0.478	0.479	0.480
Corrupted VMs	0.000	0.020	0.031	0.041	0.056	0.074	0.098	0.123	0.147	0.165	0.180	0.192	0.202	0.226	0.230	0.234	0.237	0.239	0.242
Survival VMs	0.331	0.312	0.301	0.291	0.279	0.270	0.262	0.255	0.248	0.244	0.240	0.237	0.234	0.228	0.227	0.226	0.225	0.225	0.224
Vulnerable VMs	0.336	0.368	0.359	0.352	0.347	0.343	0.341	0.340	0.338	0.338	0.337	0.336	0.336	0.335	0.335	0.335	0.335	0.335	0.334
Attacking VMs	1000	958	924	916	914	914	914	914	914	914	914	914	914	914	914	914	914	914	914
Corrupted VMs	0	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
Survival VMs	1000	1000	1000	992	961	928	895	870	852	834	822	813	804	781	778	776	773	771	770
Vulnerable VMs	1000	845	746	684	645	619	600	588	581	573	568	564	560	550	549	548	547	546	545
Attacking VMs	1000.000	955.900	923.600	916.400	914.900	914.700	914.600	914.600	914.600	914.600	914.600	914.600	914.600	914.600	914.600	914.600	914.600	914.600	914.600
Corrupted VMs	0.000	1000.000	1000.000	1000.000	1000.000	1000.000	1000.000	1000.000	1000.000	1000.000	1000.000	1000.000	1000.000	1000.000	1000.000	1000.000	1000.000	1000.000	1000.000
Survival VMs	1000.000	1000.000	996.200	981.400	950.600	915.900	886.500	860.700	838.100	823.400	810.800	801.000	794.000	775.200	772.300	769.500	765.800	763.500	762.100
Vulnerable VMs	1000.000	848.200	745.900	684.500	651.300	625.700	608.000	597.700	589.500	582.100	576.800	571.500	568.900	560.300	559.300	558.200	556.600	555.600	555.200

END