# ECONOMICS-DRIVEN APPROACH FOR SELF-SECURING ASSETS IN THE CLOUD

by

GIANNIS TZIAKOURIS


A thesis submitted to
The University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
College of Engineering and Physical Sciences
The University of Birmingham
September 2017

## ABSTRACT

This thesis proposes the engineering of an elastic self-adaptive security solution for the Cloud that considers assets as independent entities, with a need for customised, ad-hoc security. The solution exploits market-inspired methodologies and learning approaches for managing the changing security requirements of assets by considering the shared and on-demand nature of services and resources while catering for monetary and computational constraints. The usage of auction procedures allows the proposed framework to deal with the scale of the problem and the trade-offs that can arise between users and Cloud service provider(s). Whereas, the usage of a learning technique enables our framework to operate in a proactive, automated fashion and to arrive on more efficient bidding plans, informed by historical data. A variant of the proposed framework, grounded on a simulated university application environment, was developed to evaluate the applicability and effectiveness of this solution. As the proposed solution is grounded on market methods, this thesis is also concerned with examining the dependability of market mechanisms. We follow an experimentally driven approach to demonstrate the deficiency of existing market-oriented Cloud solutions in facing common market-specific security threats and provide candidate, lightweight defensive mechanisms for securing them against these attacks.

# ACKNOWLEDGEMENTS

Firstly, I would like to express my deepest gratitude to my supervisors Dr Rami Bahsoon and Dr Tom Chothia for their constant support, patience, motivation and knowledge. Their guidance throughout my research and thesis writing has been immense. I could not have imagined having better mentors for my PhD study.

In addition to my supervisors, I would like to thank the members of my thesis committee: Dr Eike Ritter and Dr Manfred Kerber for their intuitive comments and motivation, as well as their insightful questions which stimulated different perspectives of my research.

I also thank my friends Marios Zinonos, Carlos Mera-Gomez and Francisco Ramirez Mendez for their helpful comments and their assistance in engineering my proof of concept solutions.

Last but not the least, I would like to thank my parents: Andreas and Myria for their constant and enormous psychological and financial support throughout my entire life.

***I am indebted to my father for living, but to my teacher for living well.***

*Alexander the Great*

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# CHAPTER 1

# INTRODUCTION

Cloud has become the defacto solution for businesses and individuals, with an increasing number of users migrating from offline software systems to online Cloud-based services. The vast increase in consumers can be attributed to the numerous advantages that the Cloud introduces. Consumers can acquire on-demand access to various virtual computing services and resources on shared infrastructures at anytime from anywhere in the world. The Cloud can assist in lowering capital costs as it can scale to the runtime requirements of companies and individuals, providing them with ad-hoc solutions, while eliminating expensive activities such as the installation of hardware, software and its maintenance. Furthermore, the Cloud can be a dependable and robust solution for the protection of data due to the state-of-the-art security enforced by Cloud service providers in conjunction with their routine data back-ups.

Cloud hosting services can be classified under three main umbrellas, namely the Software as a Service (SaaS), the Infrastructure as a Service (IaaS) and the Platform as a Service (PaaS). SaaS solutions are concerned with the provision of ready-to-use web applications to their customers. Examples of such services are Gmail [1] and Microsoft Office 365 [2]. On the other hand, PaaS solutions (e.g. Microsoft Azure [3], Google App Engine [4] etc.) provide consumers with end-to-end IT solutions comprising hardware and software tools that allow them to develop and run their applications without the need for the complex and expensive construction and maintenance of an in-house infrastructure.

Finally, IaaS solutions deliver computing infrastructures to consumers, including computing resources and equipment such as networking, storage and datacenters. Examples of IaaS solutions are the Amazon EC2 [5], Rackspace [6] and Google Compute Engine [7].

The paradigm shift from offline software systems to online Cloud-based services has changed the way we engineer, run and continuously monitor applications and users for security. The ultra-large, dynamic and elastic nature of the Cloud has made it extremely complex, if not impossible to predict all potential malicious behaviour that might be encountered at runtime and consequently provide countermeasures for securing Cloud users in the face of these threats. As a result, the design of offline, static and reactive security strategies for the Cloud tend to be limited [8], [9], making self-adaptive security systems increasingly popular due to their potential support for dynamism in detecting and mitigating runtime threats.

## 1.1 Problem Statement

During the last decade, significant progress has been witnessed in self-adaptive security systems for the Cloud [50], [11], [12], [13]. In spite of the progress, the existing solutions tend to be limited in the way they treat security and its constraints (i.e. computational and monetary) as well as the way they discover and allocate resources to users.

Although some of the existing Cloud-based self-adaptive solutions allow the customization of their security they treat security as an "aggregated quality" by enforcing the deployment of "one service for all" while overlooking the security requirements and constraints of individual assets. This practice results in unnecessarily high costs due to the deployment of obsolete services and resources, as well as unmet security requirements per asset. The security requirements of assets can change along with the services and resources that a user access in the Cloud. They can also change with content (e.g. file modification) and contextual variations (e.g. time, location, etc.) which can emerge from

different Cloud users and the operating environment itself. Moreover, existing solutions do not facilitate effective mechanisms for the discovery and allocation of required services and resources to users. Instead, they exhaustively search vast search spaces to identify candidate solutions which can be ineffective for ultra-large environments, such as the Cloud.

Current solutions need to consider the varying nature of assets and the need for customised, ad-hoc security. This is a challenging undertaking as self-adaptive systems need to not only continuously meet the changing security requirements, priorities and constraints of multiple assets from different users. They also need to deploy lightweight mechanisms to enable users to short list and acquire suitable services and resources in an effective manner (even in the presence of scarce resources) without embarking on an exhaustive search. Finally, security software engineers need to revisit their assumption that Cloud systems maintain adequate resources for the continuous and concurrent satisfaction of large numbers of users. They need to ensure that their solutions are able to prioritise security requests based on their significance to converge to a better solution that maximises the utility of the whole environment (i.e. satisfy the largest number of requests possible) while guaranteeing the security of assets that face an imminent threat.

### 1.1.1 Defining Assets in the Context of the Cloud

The Cloud has changed the way we view and secure assets. Existing literature on asset-centric security defines assets as any data, device or any other component that supports information related operations, such as hardware, software and data. The current definition of assets [14] is outdated and unsuited for ultra-large and dynamic environments such as the Cloud. This rises from the shared and multitenant nature of these environments, where a majority of the physical and virtual/software resources can no longer be perceived as traditional assets, but as mere resources/tools serving the needs of multiple Cloud users. Therefore, it is essential to re-define assets, including the assumptions describing them to better identify what needs to be secured in the Cloud and how.

In this work, by assets, we refer to a "commodity" that can be of value for a user/organization such as files, privacy settings, location, digital identities, accounts, endpoint devices etc. that support the operation of Cloud-based applications and are owned by users. Our work clearly defines the explicit link between the security of assets and users, in the absence of closely related work. We consider that if the security of an asset is compromised then the overall security of a user is compromised.

## 1.2 Proposed Solution

We propose a self-adaptive security framework that builds on decentralised market-inspired approaches and a supervised learning technique. Our framework manages the changing security requirements/goals of assets by considering the shared and on-demand nature of services and underlying resources while catering for their monetary and computational constraints. The usage of auction procedures (i.e. English auction and Posted-Offer variant auction models) enable the proposed framework to deal with the scale of the problem and the trade-offs that can arise due to the self-interested and diverse nature of the security requests coming from the assets of various users. Whereas, the usage of a supervised learning technique (i.e. Random Forest classifier [15]) allows our framework to operate in a proactive and automated fashion by detecting runtime anomalies that could be indicators of possible security threats and mitigating them prior to their manifestation. By using the learning approach it is feasible to arrive on more efficient bidding plans, informed by historical data. Instead of entering the bidders into exhaustive bidding for candidate offers, the learning helps us to identify optimal security strategies (i.e. identify and short list appropriate services and resources) for securing an asset.

As the proposed solution is grounded on market-inspired methodologies, this thesis is also concerned with examining the dependability of auctioning mechanisms. In particular, it identifies market-specific security limitations in the engineering of commonly used market-oriented Cloud mechanisms and attempts to overcome them by proposing can-

didate defensive mechanisms for warranting the secure operation of bidders, sellers and auctioning mechanisms in market-oriented Clouds. This thesis aims to provide answers to the following research questions:

1. Can asset-centric security be more cost-effective and efficient for the satisfaction of the runtime security goals of multiple assets compared to aggregated security?

2. What techniques/mechanisms can be used in the Cloud to secure multiple assets in a proactive, cost-effective and elastic manner? The selection of market-inspired techniques as the foundation of the proposed solution raise the following sub-questions:

   - Can market-inspired mechanisms be more effective in the allocation of services and resources compared to conventional non-market mechanisms in the presence of scarce resources in security constrained environments?
   - Can learning algorithms be used to arrive on more efficient bidding plans (identify and short list appropriate candidate solutions), instead of entering bidders into exhaustive bidding for candidate offers?
   - How can representative auction models influence the security and performance of the proposed system?
   - Can market-inspired mechanisms be a dependable and secure optimisation tool for securing assets in the Cloud?

These questions are established in the subsequent chapters and sections.

### 1.2.1 Motivating Example

To exemplify the need for engineering market-inspired self-adaptive security systems for the Cloud, we consider a motivating example based on Cloud storage services in a university application environment. Despite that universities are not considered to be ultra-large environments they are characterised by multitenancy and they often store immense

amounts of sensitive data/assets, such as classified research, student data, etc. which are maintained and used by various employees (e.g. lecturers, researchers, etc.). Identifying appropriate services and resources for supporting the security of these assets is a challenging task due to the elastic and multitenant nature of these environments alongside the finite, often limited, resources available for supporting security. A candidate solution to the problem is the deployment of market-inspired self-adaptive security solutions that will allow users to compete with each other for service while catering for computational and monetary costs. By doing so it will be feasible to converge to a better solution for maximising the utility of the whole environment (i.e. satisfy as many users as possible) while ensuring the provision of service to users that face an imminent threat. In the occasion where simplistic allocation mechanisms are deployed it is possible to witness the wasteful allocation of resources to users that do not require their immediate use or cause resource starvation, which can render a system ineffective or even outright unresponsive.

To demonstrate the significance of the proposed solution we consider the following motivating example. University "X" obliges employees of externally funded projects to store their assets, for a cost, in four university owned Cloud storage services that vary in terms of security features and computational resources (e.g. disk space). The university maintains limited resources which, in some occasions, are not sufficient for the concurrent satisfaction of the security needs of a large number of employees and their assets. John is one of the externally funded employees which maintains four folders/assets (i.e. Classified Research Folder, Video Folder, Student Welfare Folder and Photo Folder) that need to be stored in a secure Cloud storage. John is often very busy and regularly handles classified data, thus he requires a self-adaptive security solution that can dynamically store his assets in the Cloud in a secure manner while catering for monetary and computational costs. To achieve this, the self-adaptive solution uses the Random Forest classifier to examine the contextual data (e.g. location, employee position, etc.) and content (e.g. file type, etc.) of each asset to identify their security requirements and dynamically short list candidate solutions based on recorded historical training data samples. Each sample

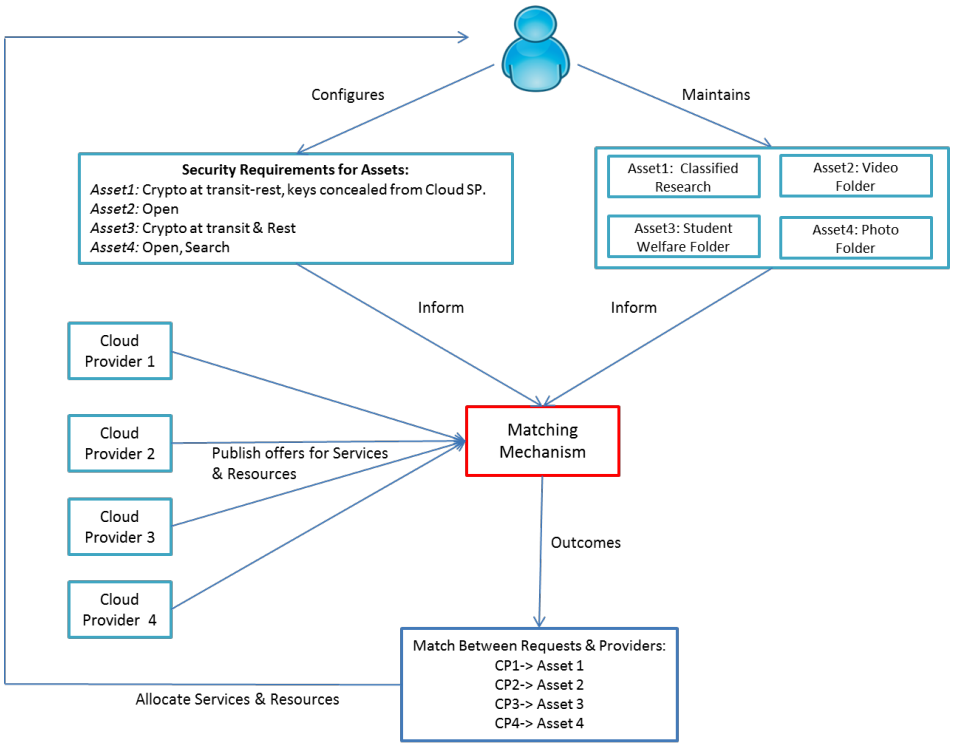


Figure 1.1: Asset-Centric allocation of storage services.

contains information concerning the security strategies that a user, such as John, followed to secure similar files in different occasions. By analysing these samples, the self-adaptive system can learn and enforce security (i.e. select appropriate storage services) in an automated and efficient manner, without entering bidders into exhaustive bidding for candidate offers. We assume that users have an insight into their own threat models and are able to evaluate their security properties. In the case where a user has insufficient information concerning the threat models and/or the security properties, he/she can use third party Cloud service providers to enforce security on their behalf.

A model security decision that the Random Forest algorithm could make for the above scenario is: *Asset1*-Classified Research Folder: Requires encryption at rest and in transit with encryption keys that are concealed from the service provider, *Asset2*-Video Folder: Requires no security, *Asset3*-Student Welfare Folder: Requires encryption in transit and at rest and *Asset4*-Photo Folder: Requires no security with search capability (Figure 1.1). Based on the identified security requirements the self-adaptive solution needs to select

appropriate Cloud storage services that can best satisfy the security requirements and constraints (e.g. disk space size, price, etc.) of each asset. As we assume that the four Cloud providers maintain scarce resources and they need to serve a large number of self-interested users and their assets, the self-adaptive system uses an auctioning mechanism to allow John and the other users to compete for service (Figure 1.2).

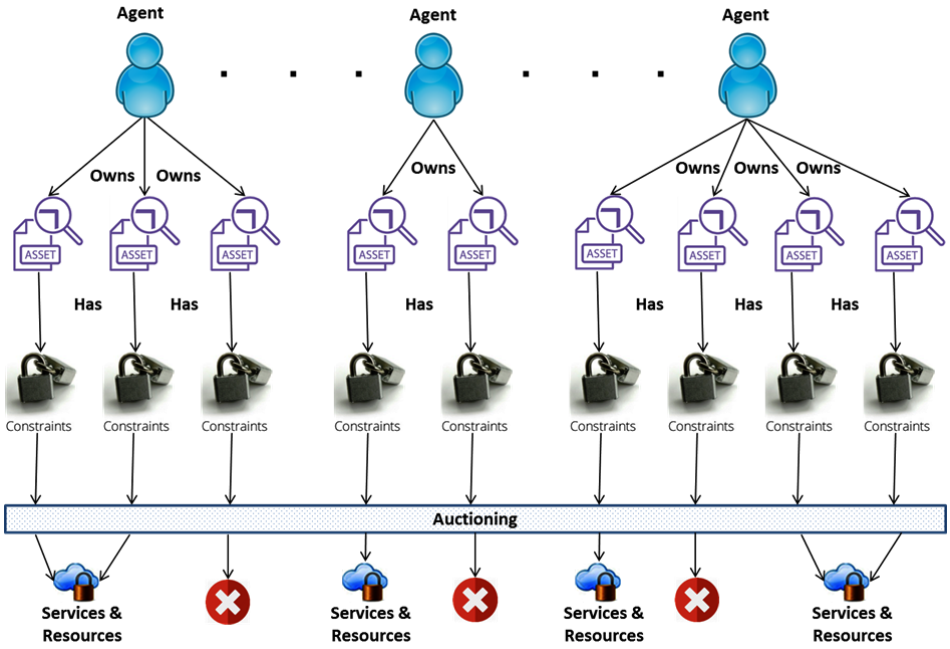


Figure 1.2: Competition between users for acquiring services and resources.

There are multiple candidate mechanisms/solutions for auctioning, with each introducing different characteristics, benefits and limitations to the problem environment. For this example, we consider a variant of the well-known Posted-Offer auction model [16]. The Posted-Offer auction model is founded on a take-it-or-leave-it basis. In this model, the SPs publicly announce the services and resources that are trading along with the prices that are willing to sell them on for a fixed trading period and it is up to the user agents/users to accept or decline the offers. In order for the proposed system to automatically determine whether a user can afford to pay a seller's requested price, as well as to identify users that face an imminent threat, we have introduced user bidding prices in the classical Posted-Offer model. The usage of bidding prices as a heuristic for ranking users based on their threat level rises from the sentiment that Cloud users

that face an imminent threat are more willing to pay higher prices compared to secured users. For each of John's assets, the self-adaptive system forms a bid encapsulating a description of its security requirements along with its computational and monetary (i.e. the highest price that John is willing to pay to secure a file) constraints. John's bids are then forwarded to a central auctioneer along with the bids of rival users for auctioning. For the purpose of this example, we assume that 1000 bids were received from various bidders and there were enough computational resources to serve only 800 bids. John submitted high bidding prices for Asset1 and Asset3 and low bidding prices for Asset2 and Asset4. As a result, Asset1 and Asset3 acquired the required Cloud storage service, where Asset3 and Asset4 were not served due to their low bidding prices and significance. Now consider that a new photo is added by John in Asset4, depicting sensitive data. The self-adaptive system identifies the runtime modification of Asset4 and re-deploys the supervised learner to determine if the security requirements of Asset4 have also changed. The goal is to form a new bid reflecting the refined security requirements and constraints of Asset4. A sample security decision from the Random Forest algorithm could be that Asset4 needs to be transferred to another Cloud storage that supports file segmentation to multiple physical machines for enhanced security.

## 1.3 Contributions of this Thesis

This thesis makes the following contributions:

1. **Literature Review on Self-adaptive Security-aware Systems:** We survey self-adaptive security methodologies. We propose an architecture-centric taxonomy drawn by the existing work in ultra-large, open and elastic environments including the Cloud. The taxonomy maps and compares current research directions in self-adaptive security systems. We reflect on the taxonomy findings and discuss limitations, research challenges and design principles in the current research and practice. We then provide recommendations concerning the future directions of

self-adaptive security systems and their effective application in elastic and open environments, such as the Cloud.

2. **Market-Inspired Methodology for Asset-centric Security:** We present a self-adaptive security system that proactively manages the runtime changes in the security goals and constraints of assets via the selection of services and resources. The system exploits decentralised agent-based market-inspired methodologies and a learning algorithm to deal with the scale of the problem as well as the need for the efficient identification of appropriate services and resources for security (even in the presence of scarce resources).

3. **Proofing Market-Inspired Methodologies from Market-Specific Threats:** Market-oriented methodologies have been widely employed by software engineers for solving dynamic allocation problems in online systems such as the Cloud [17], [18], [19], [20]. Despite the growing work in the area, to our knowledge, the existing market-oriented methodologies have not provided treatment for online market-specific threats in the context of distributed systems and the Cloud. To ensure the secure operation of these systems, it is not sufficient to protect them against generic attacks (e.g. denial of service, etc.), but to also consider possible market-specific threats that can disturb the operation of bidders, sellers and auction mechanisms. We experimentally demonstrate the deficiency of existing market-oriented Clouds in facing market-specific security threats and provide candidate, lightweight defensive mechanisms for securing market-oriented Clouds against these attacks. In addition, we compare and analyse how markets are affected in the absence and presence of the selected attacks and the proposed defensive mechanisms. The market-specific attacks considered by this work are: Shill bidding [21], Reputation attack [22], Monopoly [23] and Denial of payment attack [24].

## 1.4 Structure of the Thesis

This thesis is structured according to the aforementioned contributions.

**Chapter 2:** We perform a literature review on self-adaptive security methodologies from the context of open, ultra-large environments and examine the architectural characteristics enabling their effective application in these environments. We propose an architecture-centric taxonomy for mapping and comparing the current research directions in the field. We reflect on the taxonomic findings and discuss design principles, limitations and research challenges in the current-state-of-art and practice. We then highlight candidate future research directions contributing to the effective application of self-adaptive security systems in ultra-large, dynamic environments, such as the Cloud. Surveying existing solutions is a valuable exercise as it can assist security engineers, analysts, researchers and practitioners to better comprehend the challenges, limitations and research gaps in existence. It can also provide guidance on commonly used security mechanisms, design principles and strategies along with their strengths and pitfalls, with the objective of engineering more cost-effective, elastic and dynamic solutions.

**Chapter 3:** This chapter discusses how the proposed framework addresses the challenges presented in Chapter 1. In Chapter 3 we introduce and examine the conceptual architecture model of our framework. Among other aspects, our system analysis entails the examination of i) the entities comprising our system and their operational phases, ii) the market-oriented mechanism enabling the auctioning and allocation of the services/resources supporting the runtime security goals of assets and iv) the usage of the Random Forest algorithm for identifying candidate solutions and enforcing security in an automated manner. The applicability and effectiveness of our framework have been tested by instantiating and developing a variant of our framework, based on a simulated university application environment facilitating Cloud storage and Voice-Over-IP (VOIP) services. The use of simulation allows us to examine the applicability, effectiveness and elasticity of our system on a larger scale.

**Chapter 4:** This chapter investigates market-specific security attacks. We follow an

experimentally driven approach for exposing existing market-specific security vulnerabilities. We use a market-oriented Cloud simulation tool (i.e. CloudSim [25]) to demonstrate that the designs of existing markets are limited when facing market-specific attacks and when thwarting malicious bidders and sellers from manipulating auction mechanisms for personal gain. We then demonstrate the negative effects of these attacks on market-oriented Clouds.

**Chapter 5:** Based on the observations made, in Chapter 4, in our experiments with market-specific security attacks we develop candidate, lightweight defensive mechanisms for securing market-oriented Clouds against these attacks. To evaluate the effectiveness of the candidate solutions, we deploy them in CloudSim and analyse how the market is affected in the absence and presence of the selected attacks and the proposed defensive mechanisms. We then designate the added value of using market-specific defensive mechanisms in the Cloud.

**Chapter 6:** This chapter conducts a qualitative and reflective evaluation of the thesis with respect to our established research questions.

**Chapter 7:** We summarise the contributions and the implications of this work. We then present limitations of our work along with our thoughts on possible future research directions and their potential impact on the field.

## 1.5 Publications Emerging from this Thesis

### Conferences:

1. G. Tziakouris, R. Bahsoon, T. Chothia and R. Buyya (2016). Thwarting Market Specific Attacks In Cloud. The 9th IEEE International Conference on Cloud Computing, IEEE Cloud, San Francisco, USA, IEEE press.

2. G. Tziakouris, M. Zinonos, T. Chothia, R. Bahsoon (2016). Asset-Centric Security-Aware Service Selection. The 5th IEEE International Congress on Big Data, San

Francisco, USA, IEEE press.

3. G. Tziakouris, C. J. Mera Gomez, R. Bahsoon (2014). Securing Cloud Users at Runtime via a Market Mechanism: A Case for Federated Identity. The 16th IEEE International Conference on High Performance Computing and Communications, Paris, France, IEEE Press.

## Journals:

1. (Under review for publication) G. Tziakouris, R. Bahsoon, T. Chothia and A. Babar, "A Survey on Self-Adaptive Security for Large Scale Open Environments." ACM Computing Surveys (CSUR).

2. (Under review for publication) G. Tziakouris, C. M. Gomez, F. Ramirez and R. Bahsoon, "Economics-Inspired Self-Adaptive Security for Assets in Cloud", IEEE Transactions on Cloud Computing (TCC).

# CHAPTER 2

# TAXONOMY AND SURVEY OF SELF-ADAPTIVE SECURITY SYSTEMS FOR LARGE SCALE ENVIRONMENTS

This chapter conducts an in-depth examination of self-adaptive security solutions in the context of open, ultra-large environments, their idiosyncrasies and characteristics. We focus on the analysis of solutions that maintain architectural characteristics, such as scalability and operational transparency, which can enable the effective application of self-adaptive security systems in ultra-large and elastic environments, such as the Cloud. Large scale open environments can be viewed as Systems of Systems (SoS), which consist of several large and small distributed and complex systems [26]. The analysis of self-adaptive security systems for open environments can assist us to better comprehend the challenges, limitations and research gaps shared by modern solutions including the Cloud. It can also provide guidance on commonly used security mechanisms, design principles and strategies along with their strengths and pitfalls, with the objective of applying this knowledge in the area of Cloud computing for engineering more robust, elastic and dynamic security solutions. [2]

[2]Part of the work presented in this chapter has been submitted for publication [27].

## 2.1 Taxonomy of Self-Adaptive Security Systems

Our survey extends existing taxonomies of self-adaptive security solutions [28], [29], [30], [31], [32] by looking at them from the architectural point of view. We revisit the characteristics, design patterns and methodologies enabling these solutions to understand their fit for ultra-large environments. The proposed architecture-centric taxonomy aims to assist security engineers and researchers in better comprehending the patterns, pitfalls, gaps, strengths and limitations of contemporary implementations of self-adaptive security mechanisms when deployed in open and elastic environments, with a particular focus on Cloud. We provide guidance on the selection of design patterns based on the dynamism of the operating environment and the problem setting. Moreover, we explore how specialised application knowledge can drive different domain-specific models, approaches and patterns for autonomous software security architectures.

The remaining of this section describes the research methodology used for constructing our taxonomy, followed by the introduction and analysis of our hierarchical taxonomy.

### 2.1.1 Research Methodology

This survey can be considered a representative study that attempts to identify and examine the current state-of-the-art in self-adaptive security for open and elastic environments. The research methodology followed is grounded in the selection of indicative work that can demonstrate the current trends, challenges and gaps in the field. Even though we have not fully complied to the systematic literature review (SLR) procedure, much of the work was guided by variations for queries covering the subject of this investigation, mainly related to self-adaptive security and their applicability to open, elastic environments. Thus, we were able to alleviate the biased and restrictive nature of the search queries used in SLR studies. A detailed analysis of our research methodology is provided in the succeeding sub-subsections.

#### 2.1.1.1 Research Protocol

The research protocol followed by this study consists of i) background research; ii) the identification of appropriate research questions and objectives; iii) the identification of the targeted data sources for the collection of the surveyed work; iv) the establishment of inclusion/exclusion criteria for the selection of relevant papers; and v) the extraction and analysis of the data for constructing the taxonomy.

#### 2.1.1.2 Research Questions

Our research questions focus on the fundamental mechanisms, architectural primitives and design strategies that enable self-adaptive security systems to operate in ultra-large, elastic environments. Table 2.1 presents the research questions along with their objectives.

#### 2.1.1.3 Target Data Sources

We searched the electronic databases of high-impact conferences and journals on self-adaptive systems for selecting the relevant papers. The electronic databases used for identifying the surveyed work were the digital libraries of IEEE, ACM, ScienceDirect and Springer. These digital libraries are expected to cover a majority of the state-of-the-art work in the area. We also used Google Scholar to identify any high-impact work that was not archived by the aforementioned digital libraries.

#### 2.1.1.4 Inclusion and Exclusion Criteria

As this work is primarily concerned with the fundamental methods and architecture primitives enabling the effective application of self-adaptive security systems in elastic, ultra-large environments, we selected the papers to be reviewed based on the elasticity and dynamism of their operating environments. We mainly focused on reviewing self-adaptive systems for security that operate in decentralised and/or hierarchical configurations in

Table 2.1: Survey Research Questions and Their Objectives.

| ID | Research Questions | Objective |
|---|---|---|
| RQ1 | What dimensions of security have researchers addressed in open and ultra-large environments? | This research question aims to identify which aspects of security have been explored by researchers in open and elastic environments. |
| RQ2 | What architectural characteristics of open, elastic environments have been explored by researchers? | The objective of this research question is to identify the various architectural characteristics of elastic environments considered by researchers for promoting more effective security solutions in the context of ultra-large environments. |
| RQ3 | What are the different mechanisms enabling self-adaptive security systems to effectively operate in open, elastic environments? | This research question aims to identify various mechanisms that allow self-adaptive security solutions to manage the dynamism of ultra-large environments. |
| RQ4 | How can different architecture designs impact the effectiveness and dynamism of a security solution for ultra-large environments? | The objective of this question is to discover how different architectural designs can affect a self-adaptive security solution. |
| RQ5 | How specialised application knowledge can drive different domain-specific models, approaches and patterns for software security architectures? | This research question aims to identify how specialised application knowledge for ultra-large and elastic environments can assist in further advancing software security architectures. |
| RQ6 | What are the gaps in and limitations of effectively applying self-adaptive security solutions in open, elastic environments? | This research question aims to identify the gaps and limitations that may restrict the effective application of the existing self-adaptive security solutions in ultra-large environments. |
| RQ7 | How can we further advance the existing solutions to become more elastic and dynamic? | The objective of this question is to identify how current solutions can be advanced towards becoming more dynamic, elastic and open-ended for facing the dynamism and emerging challenges of ultra-large environments. |

the context of Cloud, Grid, Ad-hoc, mobile environments and distributed systems which are characterised by scalability, multi-tenancy, dynamism and heterogeneity. Our work does not restrict its survey to Cloud-based solutions as we believe that self-adaptive security solutions for open, ultra-large systems share various characteristics due to their similar nature. These characteristics can inform and drive our attempt for engineering more effective and dynamic solutions for the Cloud. Our survey excludes solutions that are deployed in closed, controlled and static environments. Examples of such work are the adaptive compiler techniques of Cowan et al., [33] for eliminating buffer overflow attacks, the work of Lai et al., [88] and Safe et al., [35] on adaptive steganographic techniques and the adaptive cryptographic methodologies of Dodis et al., [36]. However,

as the research conducted on self-adaptive security solutions for open environments is limited, we have relaxed our selection criteria and considered solutions that operate in semi-controlled environments, but still maintain characteristics and design patterns, such as scalability and operational transparency that can assist researchers and practitioners in developing more elastic, dynamic and open-ended security solutions. Our study only considers work that was published in the English language, in high-impact journals and computing conferences.

#### 2.1.1.5 Search and Study Selection Process

We explored four selected digital libraries, i.e., ACM, IEEE, ScienceDirect and Springer, as well as Google Scholar to identify relevant papers for revision. We were able to identify 39 research papers for review which we believe that represent a majority of the work reported on self-adaptive security systems.

### 2.1.2 Synthesis of Architecture-Centric Taxonomy

Based on our knowledge of the area, we propose a taxonomy that consists of the following fifteen dimensions: Security Goals, Control Topology, Component Composability, Component Dependency, Component Discoverability, Operational Transparency, Design Centricity, Platform Dependencies, Elasticity, Security Mechanism(s) Deployment, Adaptation Inspiration, Adaptation Awareness, Adaptation Layer, Anticipatory Support and Cost Sensitivity, which fall into two groups: the System Conceptual Model and the Adaptation Characteristics group. Similarly to the taxonomy proposed by Salehie and Tahvildari [30], our taxonomy deals with the *What* and *How* characteristics of adaptation. The system Conceptual Model group deals with the "What" aspects of security adaptation (e.g., What security qualities should I consider for adaptation? etc.). It characterizes the security goals and architecture design of self-adaptive security systems. It is feasible to relate security goals to different architectures (e.g., centralised and hier-

archical) which introduce diverse sensitivities, limitations and trade-offs in the problem environment. Whereas, the Adaptation Characteristics group is concerned with the fundamental features and characteristics of adaptation mechanisms. It deals with the "How" aspects of adaptation (e.g., How to stimulate adaptation?).

Figure 2.1 presents the proposed taxonomy and indicates, with a red dotted line, the mutually exclusive design strategies for each dimension. This will allow us as well as security software engineers to identify the possible sets of design principles that can be used together for developing more dynamic and elastic solutions. For example, a system's control topology cannot be simultaneously grounded in decentralised and centralised architectures due to their mutually exclusive nature. Security systems can be grounded in either decentralised or centralised or semi-decentralised methodologies, where all of the aforementioned architectures can be used (mutually inclusive) with hierarchical designs.

The subsequent sections provide an in-depth analysis of the dimensions comprising our taxonomy.

#### 2.1.2.1 System Conceptual Model

(i) Security Goals: This dimension describes the security objectives of a self-adaptive system. We define security goals in terms of confidentiality, integrity, availability, invasive behaviour and accountability. Our organization of the taxonomy has been guided by the taxonomy on "dependable and secure computing" [37].

- Confidentiality: Confidentiality is concerned with concealing information from unauthorized third parties.
- Integrity: Integrity is the assurance that data is an accurate and unchanged representation of the original secure information.
- Availability: Availability is concerned with ensuring that information is accessible by authorized users at all times.

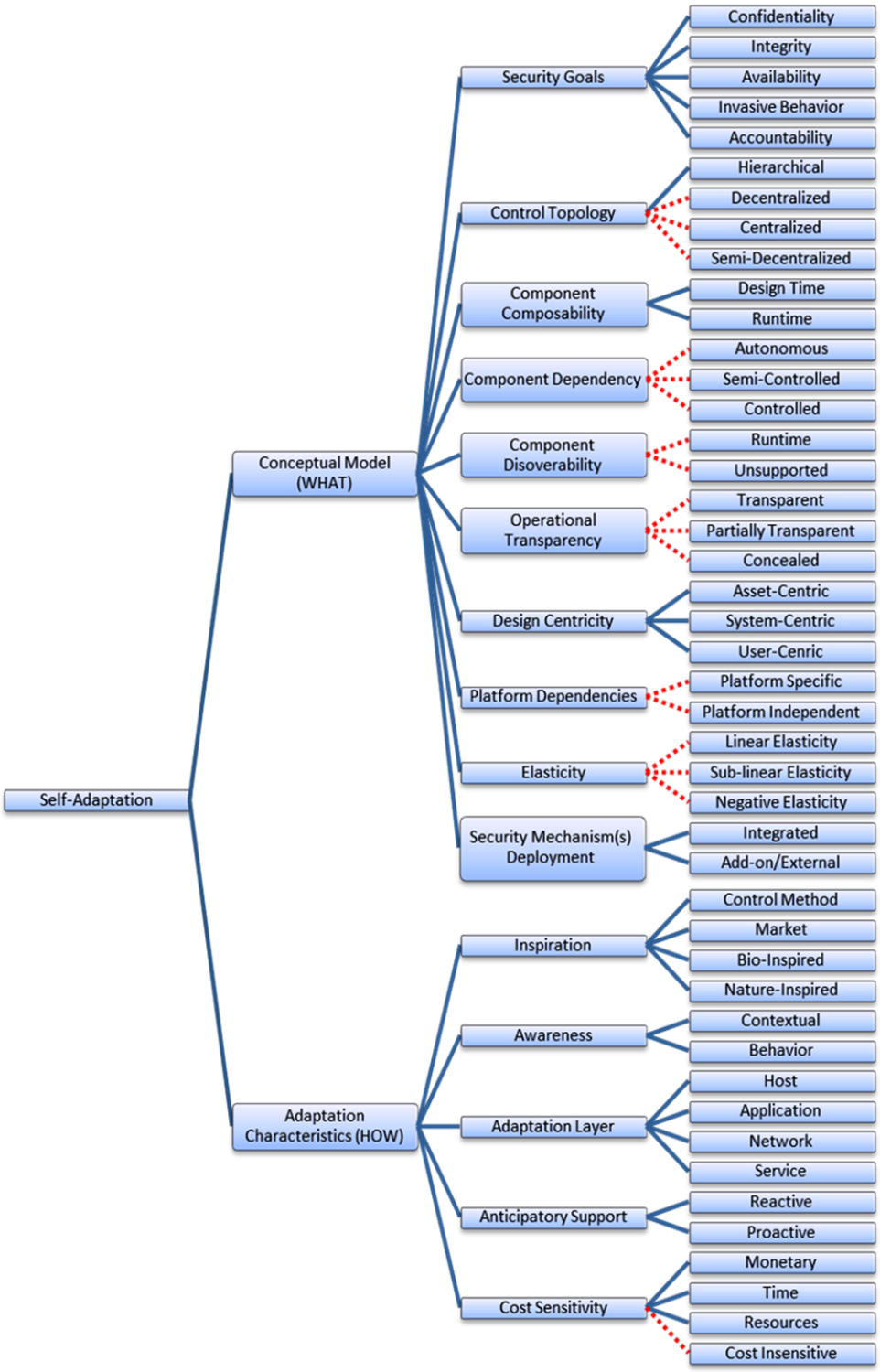


Figure 2.1: Architecture-centric taxonomy for self-adaptive security systems.

- Invasive behaviour: Invasive behaviour illustrates that a self-adaptive system is concerned with the detection and mitigation of intrusive behaviour (such as malware, DoS attacks, and code injection).

- Accountability: Accountability is concerned with the assignment of responsibility for malicious actions to the user(s) that committed them.

(ii) Control Topology: The control topology describes the architectural deployment of the components comprising a self-adaptive security solution. To describe this dimension, we use the following classes:

- Hierarchical: In hierarchical architectures, the components comprising a self-adaptive system operate in multiple layers. Adaptation in hierarchical architectures results from the operations of different layers at different times. Lower layers, often, consume short time to operate ensuring the accurate and timely adaptation of the part they are responsible for. Whereas, higher layers necessitate more time for their operations and share a more global vision. To adapt, it is required for all layers to exchange data between them and coordinate.

- Decentralised: In decentralised architectures, each host implements a complete control loop. Adaptation is achieved through the coordination of corresponding peer components from different physical hosts.

- Centralised: Centralised self-adaptive systems implement a complete adaptation loop with a central control mechanism that is responsible for the overall adaptation process. All adaptation decisions and actions are taken and deployed by a central control mechanism.

- Semi-decentralised: Semi-decentralised architectures delicate adaptation tasks to multiple distributed hosts/mechanisms in a system, which are organised by a central authority.

(iii) Component Composability: Component composability signifies if a solution is able to select multiple (a bundle) components/mechanisms to support the security requirements of a system/user. We use the following descriptors to define this dimension:

- Runtime: Illustrates that a self-adaptive system is able to compose different components at runtime for satisfying security requirements.
- Design Time: Designates that a self-adaptive solution can only use security bundles (if any) that were predefined at design time.

(iv) Component Dependency: The component dependency signifies the separation level between the components comprising a self-adaptive system. We use the following classes to define this dimension:

- Autonomous: Indicates that the components/mechanisms in a system are completely separated/isolated from each other. Though they may co-operate with each other, they have full control over their functionalities/operations.
- Semi-Controlled: Semi-Controlled components are partially controlled by a central component that orchestrates their operations; however, they still perform some core functionalities separately.
- Controlled: Controlled indicates that a self-adaptive system implements the master-slave methodology, where one or more master components fully control the functionalities/operations of slave components.

(v) Component Discoverability: The component discoverability shows if a system can discover security components/mechanisms that are situated in different locations (can be locally and/or online) at runtime for mitigating runtime threats. We use the following descriptors to describe this dimension:

- Supported: Illustrates that a system is able to discover and use components situated in different locations at runtime.

- Unsupported: Shows that a system is unable to discover new components/mechanisms. The system is restricted to use a set of pre-installed security components.

(vi) Operational Transparency: This dimension demonstrates whether a user can monitor the operations of a self-adaptive security system and how his/her data is handled by the system. We use the following classes to describe operational transparency:

- Transparent: Indicates that a system allows users to monitor its activities.
- Partially Transparent: Demonstrates that only some of the system functionalities are visible to users.
- Concealed: Signifies that a system does not allow users to monitor its operations.

(vii) Design Centricity: This dimension describes the entity/entities that a self-adaptive security system was designed to protect. We use the following classes to describe design centricity:

- User-Centric: User-centric systems revolve around the runtime security requirements of users. Based on changes in a user's security requirements, appropriate countermeasures are deployed to secure them.
- Asset-Centric: Asset-centric systems revolve around the runtime security requirements of individual assets. When runtime events occur that can threaten the security goals of assets, security countermeasures are deployed to protect them.
- System-Centric: System-centric adaptive mechanisms revolve around the runtime security of a system. Based on behaviour and/or contextual changes in a system, self-adaptive mechanisms adapt their security to protect it.

(viii) Platform Dependencies: This attribute demonstrates if a system is based on a specific platform or not. We use the following classes to describe platform dependencies:

- Platform Specific: This dimension signifies that a system was engineered based on a specific platform. To use it, certain platform requirements should be met.

- Platform Independent: Indicates that a system can be applied to any platform that can support the architecture and interface requirements of a system, without the need for specialised mechanisms.

(ix) Elasticity: This feature presents how scalable a self-adaptive security system is. We use the following classes to describe elasticity:

- Linear Elasticity: The elasticity remains constant when capacity is added to the system.
- Sub-Linear Elasticity: The elasticity factor slightly decreases when capacity is added to the system.
- Negative Elasticity: The performance of the system gets worse when capacity is added.

(x) Security Mechanism(s) Deployment: Indicates the location of the security mechanisms of a self-adaptive security system. We use the following classes to describe this dimension:

- Integrated: Signifies that the security mechanisms comprising a self-adaptive system are maintained locally.
- Add-on/External: Indicates that a system is able to select external security mechanisms situated at different physical locations (such as web services over the Internet).

#### 2.1.2.2 Adaptation Characteristics

(i) Inspiration: This dimension signifies the source of inspiration for the adaptation mechanism of each self-adaptive security system. The following classes are used to define the adaptive inspiration of self-adaptive systems:

- Control Method: Control method is the most common approach used by software engineers for self-adaptive solutions [38], [39], [40]. This technique is grounded on the automatic adjustment of system controllers at runtime to ensure the satisfaction of

security requirements when uncertain/malicious events are sensed. Classical control methods comprise a sequence of four phases: monitor, analyse, plan, and execute. The monitor phase gathers information required to inform adaptation. The monitored data is then forwarded to the analysis phase for examination. Based on the adaptation goals the analysis phase draws conclusions on which action(s) should be undertaken by the self-adaptive system. Once decisions are made the planning phase is employed to put together a series of adaptation actions to resolve unpredictability. The set of actions are then carried out in the execution phase.

- Economics Inspired: Indicates that a self-adaptive system is founded on economic methodologies, such as auctioning mechanisms [41].

- Bio-Inspired: Self-adaptive systems that are grounded on bio-inspired methodologies exploit key paradigms from biology that allow them to self-organize, self-maintain and self-heal. An example of such work is the paper of Dressler [42] on cellular metabolism for improving the efficiency of behaviour patterns of routers and firewalls.

- Nature Inspired: Nature inspired self-adaptive systems leverage mechanisms found in nature to promote security, such as how the crystal-growth process has inspired the tile architecture style [43].

(ii) Awareness: The awareness describes the attributes that a self-adaptive system monitors to inform its adaptation process. We use the following descriptors to define this dimension:

- Contextual: Self-adaptive systems that ground their adaptation on context are concerned with capturing the situational and environmental information describing the current situation of users and/or systems which can use to anticipate their immediate needs. Examples of contextual attributes are time, resource availability, location, battery and processor load. Context-driven adaptation is usually used by mobile and pervasive systems [44].

- Behaviour: Self-adaptive systems that ground their adaptation on behavioural patterns are concerned with monitoring the runtime behaviour of a user and/or a system. The runtime security of a system/user is adapted according to the behavioural patterns exhibited. Typically if the runtime actions of a system/user deviate from the expected behaviour, adaptation is triggered to ensure the satisfaction of their security goals.

(iii) Adaptation Layer: The adaptation layer describes the set of system attributes that are manipulated for performing adaptation. The attributes used to describe the adaptation layer are:

- Host: Adaptation at host level indicates the adjustment/adaptation of local system attributes such as physical resources, firmware, operating systems and visualisation aspects. An example of host level adaptation is the work of Son et al., [45] which deals with the adaptation of the behaviour of a real-time database system during transient overloads by executing transactions at a lower security level, thereby reducing resource consumption.

- Application: Adaptation at the application layer is concerned with updating policies and re-configuring application parameters at runtime. A notable work is the paper of Saxena et al., [46] which presents an autonomic framework that analyses security events and based on the results suggests a high-level security action to reconfigure a system.

- Network: Adaptation at the network level is concerned with the manipulation of communication links, networking protocols, network devices, network resources and topologies. An example of such work is the paper of Hsieh et al., [47] which performs adaptation on the network level by establishing secure communication links, broadcasting authentication between neighbouring nodes and detecting-eliminating malicious nodes in the network.

- Service: Self-adaptive systems operating on the service layer deal with the selection, composition and deployment of online services. An example of service layer adaptation is the work of Xu et al., [48] which presents an architecture that composes collaborative Cloud security services based on user runtime requirements.

(iv) Anticipatory Support: The anticipatory support shows the ability of a self-adaptive system to forecast runtime threats. To denote this feature we use the following descriptors:

- Proactive: Proactive self-adaptive solutions support the anticipation of anomalous behaviour at runtime. Such systems use learning methodologies to predict possible threats and deploy countermeasures before they are manifested. Proactive methodologies have been used by Abie et al., [49] for estimating and predicting the risk damages and future benefits for an IoT system using context-aware game theoretic models.

- Reactive: Reactive self-adaptive system do not use any forecasting mechanisms. These systems adapt once threats are manifested and detected. An example of such work is the paper of Mazur et al., [50] which uses multiple agent systems and network data to provide automated defence for malware upon their detection.

(v) Cost Sensitivity: Cost sensitivity signifies if a system is concerned with the trade-offs that can arise from adapting security. The classes used to describe cost sensitivity are:

- Monetary: Illustrates that a self-adaptive system considers the trade-off between security and the associated costs.

- Time: Demonstrates that an adaptive system is time critical and considers time as a dimension when adapting security.

- Resources: Signifies that an adaptive system considers the link between security and resource consumption.

- Cost Insensitive: Indicates that a self-adaptive system is not concerned with any of the trade-offs that can arise from adapting security.

## 2.2 Landscape of Self-Adaptive Security-Aware Research

This section categorises and presents the existing research and practice in self-adaptive security systems (SAS). We survey a representative set of work that accommodates the existing, state-of-the-art self-adaptive security solutions in large-scale open environments. We discuss the limitations, gaps, opportunities and new directions in the field. The surveyed work has been clustered based on the application areas. Thus enabling us to identify the architectural characteristics, design strategies, challenges and pitfalls shared by area-specific solutions. We have established three applications areas: Service Oriented SAS; Mobile Ad-hoc Network driven SAS; and Host-driven SAS. Service oriented self-adaptive systems operate in service repositories (e.g. Cloud) and deal with the management and adaptation of online services, computational resources, physical infrastructures and virtual machines. Mobile Ad-hoc Network driven SAS operate in mobile environments and deal with communication links, network protocol/devices and topologies. Finally, Host driven SAS operate in a localised fashion and deal with the adaptation of hardware components, firmware, operating system attributes, security policies and application properties.

Although Mobile Ad-hoc Network driven SAS and Host-driven SAS solutions do not share many architectural characteristics with service-oriented solutions, and in extent the proposed solution, the selected papers facilitate security mechanisms that are characterised by dynamism, scalability, transparency, etc. which are dimensions that can drive the design of our solution.

### 2.2.1 Service Oriented SAS

The work of Siljee et al., [38] proposes DySOA, an architecture that allows existing service oriented applications to become dynamic. This framework enables software engineers to design attributes, including security, that are concerned with the quality of service (QoS) evaluation and variable composition configuration. DySOA consists of four phases:

monitoring, analysis, evaluation and reconfiguration. During the monitoring phase, data related to the runtime requirements of QoS attributes are recorded and forwarded to the QoSCalculator which determines the current QoS. Following, the evaluation phase is deployed in which the Evaluator contrasts the quality of service information to their goals. In the case where the existing QoS cannot be satisfied, adaptation is stimulated. The reconfiguration phase determines the new application configurations based on a variation model, which provides information concerning the variability of QoS between candidate services.

Mazur et al. [50] propose a semi-autonomous defensive security mechanism for Cloud environments. The proposed system uses smart agents and network data to dynamically detect and mitigate classified and zero-day threats. The distributed intelligent agents are used at runtime to collect data related to machine language execution, services, network devices and data streams within the Cloud environment. The collected information is handled by agents who settle threats in a local fashion, wherein some occasions it is required to correlate or reason for the appropriate course of action. The brokering agents constantly alter their runtime behaviour with the assistance of game theoretic hazard evaluation and network data generated based on the input of agents. By using network data, for describing malware, the framework is able to compare pairs of data to determine if malware exists, which depends if the two pieces of data have a close match. If a threat is detected the framework uses the predefined policies to eliminate it.

The work of Tziakouris et al., [20] dynamically manages the runtime variations in the security requirements of the digital identities of users in federated Clouds. This is achieved with the allocation of suitable computational resources that support those requirements via a market inspired mechanism. The security adaptation is stimulated once the security requirements of a user are not satisfied. To initialize adaptation, a user formulates a bid that inquires the support of its runtime security goals with the allocation of additional resources from different service providers (SPs) within the federated Cloud. Once a bid is constructed, it is forwarded to the central market auctioneer which matches the bids with

available SP offers (ask) that can best satisfy the security goals of a user at the requested price. Once a match is performed, the selected SP(s) allocate the requested resources to the users to allow them to deploy computationally heavy security policies/services (e.g. an intrusion detection system that analyses network traffic for violations) for meeting their security goals and eliminate runtime threats.

Xu et al., [48] present CloudSEC, a peer-to-peer overlay-based architecture for secure service composition in the Cloud. CloudSEC uses resources from various security mechanisms scattered over a network to satisfy global runtime goals. This is achieved via the collection of data from different security mechanisms in a Distributed Hash Table. In particular, CloudSEC's service composition aims to detect and contain threats such as intrusions, distributed denial of service (DDoS) attacks, spam and malware. To achieve this, CloudSEC uses three different mechanisms: "the administration group, the collaboration groups and the peripheral entities". The administration group is the kernel of CloudSEC, which is an interface and is maintained by task coordinators. Each coordinator is responsible for a set of autonomous security agents as part of an administrative domain. Task coordinators are responsible for: i) performing security policy decisions; ii) managing collaboration tasks and iii) sharing analytical data over various domains. Following, each collaboration group composes a security service by performing a collaborative process. Collaboration groups consist of security agents that are dynamically clustered, each providing a group of global access points to various security facilities. Finally, the peripheral entity summarises service providers and users and is able to install security services and supply/consume resources via a push/pull mechanism.

The work of Li et al., [51] introduces CyberGuarder, a "visualisation security assurance architecture" that provides three types of security services: "a virtual machine security service; a virtual network security service; and a policy based trust management service". The virtual security service comprises: i) a VMM-based (Virtual Machine Management) integrity measurement methodology for network applications trusted loading, ii) "a multi-granularity network application isolation mechanism to enable OS-user isolation" and iii)

"a dynamic approach for virtual machine and network isolation for multiple network application's based on energy-efficiency and security requirements" [51]. The virtual network service allows the dynamic deployment of virtual secure services in a network application system. In such environments, secure services can be delivered as virtual machine instances and be used in a virtual network. Finally, the trust management service is used to enforce access control on network resources and play the role of a trust federation tool that selects optimal configurations for maximising the privacy and cost efficiency between various resource pools.

AdapTest [52] is a self-adaptive integrity attestation system grounded on a weighted attestation graph model for large-scale Clouds. AdapTest is able to lessen attestation overhead and reduce detection time through the dynamic selection of attested nodes. The probabilistic attestation is guided by deriving trust scores per-node and pair-wise. The proposed system is able to dynamically evaluate the trustworthiness of various services according to their previous attestation results. Attack detection is performed through the usage of replay-based consistency check, which duplicates input data and resends them as attestation data to similar service instances for consistency check. Based on the results obtained, the clique-based algorithm is deployed to discover compromised nodes with the construction of an attestation graph that represents nodes as identical services. If two nodes return a "consistent output" they are labelled as legitimate users, where if an inconsistent result is received they are classified as compromised.

Youngmin and Chung [53] propose an authorization algorithm based on an improved role-based access control technique that dynamically determines the access level to Cloud computational resources considering contextual and security information. Once a user attempts to retrieve the protected resources, the Service Provider (SP) collects the context data from both the user and the environment to perform access decisions. The SP consists of two modules: the service module, which provides users with various types of services (such as e-commerce and Digital Rights Management service) and the security module that offers the security function of the services. The algorithm uses a context

interpreter that converts collected contexts to quantitative values, which are used to evaluate the security level and the access control algorithm required. According to the security level, role, and access policy, the context engine determines and allocates the appropriate security services to a user.

Squicciarini et al., [12] propose policy execution techniques for the dynamic protection of users' sensitive resources (referred by the authors as "security-aware objects") in the Cloud. Based on the contextual characteristics of a security-aware object, the local laws and the service level agreements, the proposed system adapts and deploys security policies of varied granularity. For each security-aware object, the system deploys five key components for its protection: i) authentication and authorization tools; ii) self-enforcement policy engine; iii) security policies in executable form; iv) secure connections manager and v) protected file(s). When a security-aware object is created or is moved to a new location, the policy composition and translation process is activated to ensure that only relevant and applicable policies are used. The policy translation process is preceded by the selection of applicable rules, followed by the static ordering of applicability, which reveals the optimal available policy.

Ma and Wang [54] introduce a self-adaptive access control model for the Cloud. The model is based on feedback loops and consists of five phases: monitor, analyse, plan, execute and knowledge-base. The feedback loop starts with monitoring the access requests, access attributes, access behaviour and history records of a user which are then provided as an input to other modules. The proposed model uses an analysis module to examine the recorded access behaviour of a user, which determines whether it is required to update the knowledge-base by selecting a sample from the records. The knowledge-base contains basic access control information, including the relation degrees among access control attributes. According to access feedback data, the relation degree can self-repair and self-improve. Following, the plan module computes the relation degrees in the sample history records and updates the knowledge-base to provide decision support for access control. Lastly, the execute module is used as an interface to retrieve the new knowledge-

base and perform the access request decisions.

Yee et al., [55] promote a self-adaptive intrusion detection system for protecting web services from security threats associated with SOAP/XML/SQL. To achieve this, the proposed system deploys agents to monitor user behaviour. The recorded behaviour consists of: "the source and destination IP, service requests and responses, user ID, SOAP parameters, message size, request/response time frame, the number of messages over a certain time frame for a request/response, valid SQL commands and valid XML syntax and schema". The agents are then using data mining methods to identify violations. These violations are further analysed with the usage of fuzzy logic to reduce false positives. In the case of a detected violation, the action provider can either block, reject or terminate an activity to eliminate/mitigate the violation.

### 2.2.2 Mobile Ad-hoc Network Driven SAS

The work of Dressler [42] explores similarities between computer networking and cellular mechanisms. This work analyses how molecular biology can be promoted as a generic approach for self-organizing solutions in computer networking. Specifically, the author examines how the signalling pathways can be adaptable to information exchange in network security environments and other communication relationships.

Chigan et al., [39] address the issue of limited resource consumption in Mobile Ad-hoc Networks (MANETs), which unintentionally causes Denial-of-Service attacks. To resolve this limitation, the authors promote a framework that can design an adaptive network provisioning scheme that caters for the security and resource consumption of users. The proposed framework operates on two layers: the "offline optimal secure protocol selection module" and the "online self-adaptive security control module". By using these modules it is feasible to deploy various combinations of security protocols at runtime. The "offline optimal protocol selection module" is used to determine the optimal permutation of security protocols between various system layers, where the "online self-adaptive security control module" is responsible for alleviating the trade-off between network and secu-

rity performance in MANETs. The proposed framework quantitatively evaluates various permutations of protocols based on their security level and the computational overhead they produce, which is achieved with the usage of two "quantitative indexes: the Security Index (SI) and the Performance Index (PI)". The SI illustrates the effect of each group of security protocols to the overall security of a MANET, where PI signifies the performance of a security protocol in a network according to the quality of a specified service. By using these benchmarks the offline module is able to deploy different sets of secure protocols that provide varying security capabilities and are associated with different corresponding performance costs. To compare protocol sets the proposed system utilises two SI quantification procedures: the equal and unequal threat procedures. The equal threat procedure is used when little information exists concerning a threat. Whereas, the unequal threat procedure is used when substantial information exists for a threat. Lastly, the "online self-adaptive security control module" uses data from the "offline optimal secure protocol selection module" to dynamically adapt the secure protocols according to contextual changes.

Kong et al., [56] propose a framework for authenticating nodes and detecting security intrusions in hierarchical Ad-hoc networks with Unmanned Aerial Vehicles (UAVs). Depending on the infrastructure changes on a UAV-MBN (mobile backbone node) network, the proposed solution alters between two modes to satisfy the network and security requirements, namely: i) the infrastructure mode, in which UAVs play the role of central authentication authorities and ii) the infrastructure-less mode, which is employed once all UAVs malfunction or are eliminated. The framework consists of three layers: the "ground mobile nodes (soldiers), the ground mobile backbone (MBN) nodes, and the UAV nodes" [56]. Based on the availability of UAVs in the network, a mobile backbone node alters between the two modes of communication. When UAVs are absent, the surviving units turn to the infrastructure-less mode for security and communication. The authentication of nodes and intrusion detection in the infrastructure-less mode is localised to each ground node until a new UAV is available to switch back to the infrastructure mode.

In infrastructure mode, UAVs take the role of centralised certification authorities and perform node authentication and distributed intrusion detection.

The work of Farid et al., [57] presents an adaptive network intrusion detection system that reduces false positives and maximises the detection rate when classifying intrusions. This is achieved by using a modified Bayesian algorithm on given datasets describing sets of probabilities concerning the likelihood of being part of various clusters. Zero-day attacks are classified in a training set and assume maximum value. The weights assigned to training sets are adjusted until every training set is classified or until a certain threshold of classification accuracy is achieved. Based on the weights of the training samples produced and the large volume of network data analysed, it is feasible to enhance the effectiveness and precision of the detection mechanism.

Kurosawa et al., [58] propose a new detection method for MANET environments that is grounded in dynamically updating learning data. As a case study, the authors use the Ad-hoc On-demand Distance Vector (AODV) routing protocol. The framework uses four features of the AODV routing protocol to detect abnormalities in a network link, namely: "Number of received RERR messages, Number of sent out RERR messages, Number of dropped RREQ messages and Number of dropped RREP messages" [58]. To detect anomalous behaviour, all nodes in the MANET monitor their traffic. The framework uses the Principal Component Analysis methodology on the recorded traffic data to explore the correlation between the normal and malicious state of network activity. When the projection distance between the two vectors surpasses a certain threshold, then this traffic is labelled as an attack. On the contrary, all normal traffic is used as a learning dataset.

Abie and Balasingham [49] present a risk-based self-adaptive security system for the Internet of Things (IoT) in e-Health. The proposed system is capable of estimating and predicting system benefits and risks with the assistance of context-aware game theoretic models. The system consists of the following models: "Adaptive risk management model; Adaptive monitoring model; Analytics and predictive model; Adaptive decision-making model; and Evaluation and validation model" which operate in a repeated cycle. This cy-

cle allows the system to alter its security policies according to the obtained estimations. The adaptive monitoring model adapts the architecture through a continuous cycle of monitoring and analysis of contextual data and IoTs status information at runtime. The analytics and predictive model examines the recorded data with game theory models to evaluate and forecast possible risks and benefits. Following, the decision-making model adapts to the environment, the runtime variations of things and the potential threats. Finally, the evaluation and validation model is used to discover trade-offs between different solutions by varying assumptions on threats and requirements, which leads to the selection of better metrics.

Robertson and Laddaga [59] suggest a self-adaptive trust modelling system for networked resources. The proposed framework is founded on conditional preferences and the principle of maximum entropy [60]. The authors assume that they maintain explicit models of the computational assets that their framework reasons over including the models of executing configurations and candidate alternative configurations. The framework uses these models to identify unexpected behaviour and diagnose which parts are responsible for the unexpected behaviour.

Hsieh and colleagues present SecCBSN [47]. The system comprises of three security modules that are used for establishing secure communication links, broadcasting authentication between neighbouring nodes, and detecting and eliminating malicious network nodes. The self-organization module is initially used to adapt the clustering algorithm of LEACH. To achieve this, the cluster head plans future transmissions and monitors the periods for each of the nodes in the network. Following, member nodes transmit data to the base station through the cluster head within the predetermined transmission periods. Each cluster further divides its members to multiple monitoring subgroups which are used to monitor communication throughout the delivery phase. SecCBSN authenticates newly registered sensor nodes by using neighbouring authenticated nodes at the end of each cluster round. All authenticated nodes share pairwise keys with all their neighbouring nodes, which results to trust relations between the nodes. To achieve greater

security, SecCBSN utilises a secure transmission module that operates between member nodes and the cluster head; and between the cluster head and base station nodes in which symmetric and pairwise keys are used for the authentication of nodes and their private data. All authenticated member nodes are then forwarding their encrypted data to the cluster head. The cluster head then forwards the cumulative obtained data (encrypted with the shared key) to the base station by the end of each cluster round. In the case that a network node is compromised, SecCBSN employs the compromised node detection and elimination module in which an alarm-return protocol is deployed. All monitoring nodes are able to report malicious nodes to the base station by forwarding alarm packets. Once an alarm packet is received by the base station, the source node is evaluated based on a trust value. The evaluated nodes are then clustered into a blacklist or a whitelist based on their given values. Nodes in the blacklist are eliminated, whereas nodes in the whitelist are used as a reference for candidate member nodes.

Son and colleagues engineered an adaptive security manager for distributed database systems [45]. The security manager is responsible for authenticating clients and enabling secure communications between them while catering for resource consumption in the system. In the case of a system overload, the manager changes its behaviour by deploying computationally lighter security, thus lessening resource consumption. This work exploits a multi-level security classification methodology, in which the upper layers provide higher security and necessitate more resources than the underlying layers.

Schneck and Schwan engineered Authenticast [61], a dynamic communication protocol that offers varying security levels. Authenticast aims at satisfying the runtime authentication needs of clients while catering for the trade-offs between security and performance. More precisely, this work is concerned with the manipulation of CPU resources and the elimination of security threats. The solution uses heuristics to dynamically determine the level of authentication required in a network and the strategy for deploying the security solution. The proposed method uses a "security thermostat" as a controller for deploying adaptive authentication policies at runtime.

Zou and colleagues present an intelligent firewall architecture grounded in a fuzzy adaptive security algorithm [62]. The proposed framework uses six different modules, the: "packet capture and data mining module, static packet filter, dynamic packet monitor, address translation gateway, control module and security policy rules". A fuzzy controller is used as the input of the system, which records the characteristics of network packets. The system determines and adjusts the security level for each network packet based on the various states of each packet, hence mitigating the trade-off between performance and security.

The work of Tedesco et al., [63] proposes an intrusion detection algorithm inspired by the human immune system for discovering zero-day attacks. This is achieved by detecting packets whose contents significantly diverge from the current signature databases with the assistance of dendritic cells (DC) and T-cells. DC are a part of the immune system, which interacts with antigen to regulate the state of adaptive immune systems cells, where "T-cells are members of the adaptive immune system that use receptors that bind to antigen presented in an MHC-antigen complex on the surface of DCs and respond to the strength of the match between receptor and antigen" [63]. Once a DC depicts the needed packets, it is then the duty of T-cells to decrease their number by discovering similarities in their data structure. DCs are responsible for discovering anomalous behaviour, where T-cells are dealing with the selection of patterns among the antigen data.

Awais and colleagues propose a "bio-inspired self-defending framework" for IP multimedia subsystems [64]. Their system is used for protecting infrastructure nodes against denial of service (DoS) and distributed denial of service (DDoS) attacks. It serves as an artificial immune system that classifies anomaly detection that emerges from the analysis of network packets. To detect DoS and DDoS attacks, the proposed system uses the negative selection classification technique, which can be found in the immune system. Lymphocytes (detectors) grow in thymus and undertake the negative selection process. The lymphocytes that are able to withstand the negative selection procedure remain in the thymus (database describing normal behaviour). Following, the remained lympho-

cytes are distinguishing and separating the self (healthy packages) and non-self (malicious packages) antigens.

The work of Carney and Loe [40] proposes four techniques for the engineering of dynamic security policies for distributed trusted operating systems that separate the definition and enforcement of a policy in a server. The distributed trusted operating system comprises a micro-kernel and a set of servers. The micro-kernel coordinates and manages the communication between servers, where the servers provide different operating system services (e.g., authentication). Once a service request is made, the micro-kernel forwards the security identifiers describing the context of a subject/object to a special server, called the Security Server, to define the security policies. The four methods studied by this work are the following: "i) Reloading a new security database for the security server, ii) Expanding the state and security database of the security server to include more than one mode of operation, iii) Implementing another security server and handing off control for security computations and iv) Implementing multiple, concurrent security servers each controlling a subset of processes". The four methods have been evaluated based on five criteria: policy flexibility, functional flexibility, security, reliability and performance.

The work of Bailey et al., [65] proposes a self-adaptive authorization system for managing distributed policy-based access control and attribute-based access control authorization infrastructures. The system uses a control loop that monitors the behaviour of users and based on its assessment the authorization policy is either relaxed or strengthened to meet the new requirements at hand. The proposed system consists of five manageable assets: the attributes-credentials, "the attribute authority's credential issuing policy, the resource owner's credential validation and the access control policies". Throughout the modification of these attributes, the system is able to manage the credentials issue to users, thus increasing/decreasing their permissions. By adapting/switching between authorization policies the system is able to control the access rights for different groups of users at policy level.

Foo et al. promote an automated, proactive response mechanism for the containment

of intrusions in distributed e-commerce systems comprising of interacting services [66]. The proposed system uses an I-GRAPH to define the associations between services in terms of intrusion spread. By maintaining this graph, the system is able to proactively stop an attacker from moving between attack nodes with the deployment of countermeasures at specific nodes. This is achieved by estimating the likelihood of an intrusion spread between neighbouring services based on a propagated I-GRAPH and then determining the appropriate countermeasures for facing an attack. The decisions taken are grounded on the disruptivity of the responses to non-benign system processes, the effectiveness of previously deployed countermeasures and the assurance that the intrusion will occur. Finally, the feedback mechanism examines the outcome of the enforced countermeasures and uses that for future decisions.

Hassan and Abdellatif [67] propose a two-layer framework that adapts the security of mobile agents at runtime. The first layer performs a static adaptation with the assistance of MSAS (Management System of Agents Security) component, which is a storage for various security mechanisms. Based on the services requested by an agent, the MSAS component selects appropriate security mechanisms for the agent and determines how the mobile agent can further adapt at runtime. Following, the second layer is deployed for performing a reflexive structural adaptation. Based on the degree of confidence that a mobile agent has for the operating platform the security mechanisms are accordingly adapted.

The work of Tsigkanos et al., [68] proposes a topology-aware adaptive security system that can identify possible violations in the security requirements that emerge from the topological changes of users and their assets. Based on the occurring runtime changes, the system selects appropriate security countermeasures to prevent these violations before they are manifested. To achieve this, the proposed methodology uses the paradigm of ambient calculus to represent a real-time model of the topology, including the agents and assets comprising the environment, as well as the possible future states of the system. Whenever assets/agents re-allocate, or the structure of the physical space is changed,

the system tries to identify possible violations in their security requirements and deploys appropriate countermeasures to avoid them. This work grounds its adaptation strategy on topological changes, which can prove ineffective in dynamic environments, such as the Cloud, where the location of users and assets is often unknown.

### 2.2.3 Host Driven SAS

Sexena and colleagues present a system for autonomic security that is grounded on an adaptation loop comprising of: monitoring, analysing, and responding modules [46]. The proposed system selects different access control, authentication and cryptographic security services to reconfigure itself according to a system's runtime security requirements. To accomplish this, the system starts with the deployment of monitoring modules that observe security related events, called security context. Following, the analysing modules subscribe to the events detected by the monitoring modules and based on the type of the events, each analysing module proposes a security task for re-configuring the system. Finally, the responding module maps these security actions to implementation specific sub-systems.

Salehie et al. [69] consider assets as primary entities that dynamically evolve at runtime and need to be individually secured. At design time, the framework uses three models describing: "assets, threats and security requirements" along with risk information and utility nodes for building a fuzzy causal network. The constructed fuzzy causal network defines the elements (i.e. assets, threats, security requirements) and the links between them, which are used to determine how the runtime variations in assets can influence security. The fuzzy causal network is updated and used at runtime when assets are modified to provide a set of security options, from which the most suitable option is depicted for adaptation. The authors test their framework based on a mobile phone example, where different mobile phone devices maintain different assets with varying significance and different security policies are enforced. In spite of the asset-centric nature of the solution, the way that assets are defined and the assumptions describing them

significantly vary from assets in the Cloud. This rises from the shared and multitenant nature of the Cloud, where a large number of physical and software resources can no longer be perceived as traditional assets, but as mere resources serving the needs of users.

Shrobe and colleagues examine two information systems (i.e., PMOP and AWDRAT) and show how they can adaptively defend against internal and external attackers [70]. PMOP is used as a defence mechanism for internal threats, where AWDRAT is deployed for detecting external system threats. In particular, PMOP is used to identify if a system administrator has requested an action that is considered malicious by a system, while AWDRAT is used to determine if a system's behaviour is correct in response to a non-malicious request. Both AWDRAT and PMOP are able to discover variations from benign behaviour via the usage of self-monitoring wrappers that collect data and control application tasks; and an architectural methodology that predicts the behaviour of a system in response to internal user requests. More specifically, the authors developed a malicious behaviour detector that examines a user's application modification records and compares this information to a static model representing benign behaviour. The described model provides information concerning the type of behaviour that can be anticipated along with information on how to assess the correctness of the exhibited behaviour, which is achieved with the analysis of plant models, security models, efficient applications, heuristics and design policies.

Harmer and colleagues promote an adaptive, distributed defence system that is grounded on biological methodologies within a multi-layer architecture [71]. The proposed system is able to detect, identify and eliminate malicious code and bad network packets. The level of effectiveness is tunable through the selection of the number of antibodies, the antibody length, and the detection threshold.

Abie and colleagues propose an adaptive messaging middleware called GEMOM, which promotes self-adaptability and assurance to malicious behaviour and incorrect input at runtime [72]. GEMOM uses dynamic security and a quality of service model comprising a "continuous cycle of monitoring, assessment and evolution". More specifi-

cally, contextual information is gathered from within the system and the environment for analysis which results in the adjustment of security functions (e.g., encryption schemes).

The work of Venkatesan et al., [73] presents a threat-adaptive firewall that uses a trust state mechanism for capturing various levels of security while catering for a system's performance. More specifically, the authors promote a threat-adaptive methodology with a dynamic non-binary access control mechanism and various levels of authentication based on the runtime threats. The proposed method links a trust level with each user and accordingly deploys an ad-hoc security policy for each.

Carver et al. [74] propose an automated intrusion response system that uses a group of software agents as the first line of defence for the protection of computer systems until their administrator can take action and protect them. The proposed system uses five types of agents, namely the: interface agent, master analysis agent, response taxonomy agent, tactics agent and policy specification agent. The interface agents poses a model for each intrusion detection system, illustrating the number of previous false negatives and positives occurred. These models are used for constructing a confidence metric, which is then forwarded with the intrusion report to the "master analysis agent". Upon reception, the intrusion is classified as either a new or an existing threat. In the case of a new threat a new "analysis agent" is generated to devise a set of countermeasures, whereas if it is an already existing threat the "confidence metric and intrusion report" is forwarded to the "analysis agent" responsible for the attack. The "analysis agent" generates a specification describing the actions to be followed for resolving the attack by invoking the response taxonomy agent and policy specification agent to respectively categorize the threat and concretise the security countermeasures. Following, the analysis agent forwards the planned actions to the tactics agent for carrying out the devised countermeasures. Finally, the system records the decision of the analysis and tactic agents for administrator review.

Evesti et al. [75] present a self-adaptive security methodology for smart spaces. The proposed system is based on control loops. Initially, the monitor phase employs mon-

itoring probes to record the changes on security-related characteristics from the smart space and the applications comprising it. Once a significant amount of knowledge is gathered, the recorded data are examined to determine if the existing requirements are satisfied. If not, the planning phase generates an adaptation strategy, which is implemented during the execution phase. The authors test their system on a case study based on authentication and authorization methodologies.

Locasto et al. [76] promote FLIPS, an adaptive intrusion prevention system that thwarts binary code injection attacks at host level. To achieve this, FLIPS uses a hybrid methodology combining anomaly classification and signature matching mechanisms. The feedback used for the proposed framework is obtained by STEM, an x86 emulator. STEM is able to "discover injected code, automatically recover from an attack and forward the attack code to the anomaly and signature classifiers". To evaluate their system, the authors have deployed it to an HTTP server for discovering zero-day attacks.

The work of Garlan et al., [77] proposes an automated mechanism for detecting and recovering a system from errors with the enforcement of "externalize" adaptation. Externalise adaptation is performed by deploying components outside the system for monitoring its behaviour to determine whether the exhibited behaviour abides to the acceptable/predefined system design. The externalised mechanisms sustain and utilise system models defining an abstract, global view of the running system along with support reasoning concerning system errors/problems and repair plans. In the case of a system violation, a repair mechanism is deployed to adapt the architecture of the system and protect it.

## 2.3 Application of Proposed Taxonomy

This section applies the proposed taxonomy to the surveyed work presented in section 2.2. The findings of our taxonomic analysis are clustered based on the design centricity of the surveyed work and presented in three tables. Table 2.2 presents self-adaptive se-

curity solutions that are concerned with the runtime security of assets; Whereas, Table 2.3 and Table 2.4 present work that revolves around the runtime security of users and software systems respectively. Due to the varying nature and composition of different entities (i.e., users, assets, and systems), different security approaches/mechanisms can be used for securing them. These mechanisms can operate at different layers with varying architectures and methodologies. By grouping our findings according to the design centricity of the surveyed solutions, it is feasible to extract and identify the fundamental characteristics, idiosyncrasies and design patterns shared by security solutions of similar centricity. Thus, gaining a better understanding of the methods that need to be followed when securing specific entities in open environments such as the Cloud.

## 2.4 Taxonomic Analysis Findings and Observations

This section reports and discusses our taxonomic findings. The results have been presented in terms of the fifteen dimensions comprising our architecture-centric taxonomy. For each dimension, we present a quantitative analysis of the design principles and approaches used by security engineers when developing self-adaptive security systems, along with our understanding of their limitations and challenges. We then propose strategies for overcoming these limitations and challenges, to engineer more dependable, elastic and dynamic solutions.

Table 2.2: Taxonomic Findings for Surveyed Asset-Centric Security Solutions.

| | System Conceptual Model | | | | | | | | | Adaptation Primitives | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Reference** | **Security Goals** | **Control Topology** | **Component Composability** | **Component Discoverability** | **Component Dependency** | **Operational Transparency** | **Platform Dependancies** | **Elasticity** | **Security Mechanism(s) Deployment** | **Inspiration** | **Awareness** | **Adaptation Layer** | **Anticipatory Support** | **Cost Sensitivity** |
| [12] | Confidentiality (Authorization) Integrity (Authentication) | Centralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Control Method | Contextual | Service | Reactive | Cost Insensitive |
| [59] | Invasive behaviour | Centralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Control Method | Contextual | Network | Reactive | Cost Insensitive |
| [69] | Integrity (Authentication) Confidentiality (Encryption) Availability Accountability | Centralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Control Method | Contextual | Application Host | Reactive | Cost Insensitive |
| [68] | * | Semi-decentralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Control Method | Contextual | Network Application | Proactive | Cost Insensitive |

'*' Indicates a generic adaptive security solution with no explicit links to any specific security features (e.g. integrity, confidentiality etc.).
'N/A' Indicates that a work is not concerned with a dimension of our taxonomy.

Table 2.3: Taxonomic Findings for Surveyed User-Centric Security Solutions.

| | System Conceptual Model | | | | | | | | | Adaptation Primitives | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Reference** | **Security Goals** | **Control Topology** | **Component Composability** | **Component Discoverability** | **Component Dependency** | **Operational Transparency** | **Platform Dependancies** | **Elasticity** | **Security Mechanism(s) Deployment** | **Inspiration** | **Awareness** | **Adaptation Layer** | **Anticipatory Support** | **Cost Sensitivity** |
| [20] | Confidentiality Integrity Availability | Semi-decentralised | Design Time | Supported | Autonomous | Transparent | Platform Independent | Linear | Add-on | Economic Inspired | Contextual | Service | Reactive | Resources Monetary |
| [53] | Confidentiality (Authorization) | Centralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Control Method | Contextual | Service | Reactive | Cost Insensitive |
| [54] | Confidentiality (Authorization) | Centralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Control Method | Contextual | Service | Reactive | Cost Insensitive |
| [39] | Availability (DoS) | Centralised | Design Time | Unsupported | Semi-Controlled | Partially Transparent | Platform Independent | N/A | Add-on | Control Method | Contextual | Network | Reactive | Resources |
| [61] | Integrity (Authentication) | Centralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Control Method | Contextual | Network | Reactive | Resources |
| [46] | Confidentiality (Authorization) Integrity (Authentication) | Centralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Add-on | Control Method | Contextual | Application | Reactive | Cost Insensitive |
| [67] | * | Semi-decentralised | Runtime | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Control Method | Contextual | Service | Reactive | Cost Insensitive |

"N/A" Indicates that a work is not concerned with a dimension of our taxonomy.
"*" Indicates a generic self-adaptive security solution with no explicit links to any specific security features (e.g. integrity, confidentiality etc.).

Table 2.4: Taxonomic Findings for Surveyed System-Centric Security Solutions.

| | System Conceptual Model | | | | | | | | | Adaptation Primitives | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Reference** | **Security Goals** | **Control Topology** | **Component Composability** | **Component Discoverability** | **Component Dependency** | **Operational Transparency** | **Platform Dependencies** | **Elasticity** | **Security Mechanism(s) Deployment** | **Inspiration** | **Awareness** | **Adaptation Layer** | **Anticipatory Support** | **Cost Sensitivity** |
| [38] | - | Centralised | Runtime | Supported | Semi-Controlled | Partially Transparent | Platform Independent | Sub-linear | Integrated Add-on | Control Method | Contextual | Service | Reactive | Cost Insensitive |
| [50] | Invasive behaviour (Malware) | Semi-decentralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Control Method | Contextual | Service | Reactive | Cost Insensitive |
| [48] | Invasive behaviour (DDoS, spam, malware) | Semi-decentralised | Runtime | Supported | Autonomous | Transparent | Platform Independent | Sub-linear | Add-on | Control Method | Contextual | Service | Reactive | Cost Insensitive |
| [51] | Accountability Confidentiality (Authorization) Integrity (Authentication) | Semi-decentralised | Design Time | Unsupported | Controlled | Partially Transparent | Platform Independent | N/A | Integrated | Control Method | Contextual | Service | Reactive | Resources |
| [52] | Integrity (Trustworthiness) | Centralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Control Method | Contextual | Service | Reactive | Time |
| [42] | Invasive behaviour | Semi-decentralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Bio-inspired | Contextual | Network | Reactive | Cost Insensitive |
| [56] | Integrity (Authentication) Invasive behaviour | Hierarchical decentralised | Design Time | Unsupported | Autonomous | Concealed | Platform Independent | N/A | Integrated | Control Method | Contextual | Network | Reactive | Cost Insensitive |
| [57] | Invasive behaviour | Centralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Control Method | behaviour Contextual | Network | Proactive | Cost Insensitive |
| [58] | Invasive behaviour | decentralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Control Method | Contextual | Network | Reactive | Cost Insensitive |
| [49] | Invasive behaviour | Centralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Add-on | Control Method | Contextual | Host Network | Proactive | Cost Insensitive |
| [47] | Invasive behaviour Integrity (Authentication) | Hierarchical | Design Time | Unsupported | Controlled | Concealed | Platform Specific | N/A | Integrated | Control Method | Contextual | Network | Reactive | Cost Insensitive |
| [45] | Integrity (Authentication) Confidentiality (Encryption) | Centralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Control Method | Contextual | Network | Reactive | Resources |
| [62] | Invasive behaviour | Centralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Control Method | Contextual | Network | Reactive | Time |
| [63] | Invasive behaviour | Centralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Bio-inspired | Contextual | Network | Reactive | Cost Insensitive |
| [64] | Availability (DoS, DDos) | Semi-decentralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Bio-inspired | Contextual | Network | Reactive | Cost Insensitive |
| [40] | * | Semi-decentralised | Design Time | Supported | Semi-Controlled | Concealed | Platform Independent | N/A | Integrated | Control Method | Contextual | Application Host | Reactive | Cost Insensitive |
| [70] | Invasive behaviour | Centralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Control Method | behaviour | Application | Reactive | Cost Insensitive |
| [71] | Invasive behaviour | Hierarchical | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Bio-Inspired | Contextual | Application Network | Reactive | Cost Insensitive |
| [72] | * | Semi-decentralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Control Method | Contextual | Application | Reactive | Cost Insensitive |
| [73] | Integrity (Authentication) Confidentiality (Authorization) | Centralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Control Method | behaviour | Application | Reactive | Cost Insensitive |
| [74] | Invasive behaviour | Centralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Control Method | Contextual | Application Host | Reactive | Cost Insensitive |
| [55] | Invasive behaviour (SOAP/XML/SQL) | Centralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Control Method | behaviour | Service | Reactive | Cost Insensitive |
| [65] | Confidentiality (Authorization) | Semi-decentralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Control Method | Contextual | Network Application | Reactive | Cost Insensitive |
| [75] | Integrity (Authentication) Confidentiality (Authorization) | Semi-decentralised | Design Time | Unsupported | Semi-Controlled | Concealed | Platform Independent | N/A | Integrated | Control Method | Contextual | Application Host | Reactive | Cost Insensitive |
| [66] | Invasive behaviour (Threat containment) | Semi-decentralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Control Method | behaviour | Network | Proactive | Cost Insensitive |
| [76] | Invasive behaviour (Code Injection) | Centralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Control Method | behaviour | Host | Reactive | Cost Insensitive |
| [77] | Availability (System errors) | Centralised | Design Time | Unsupported | Controlled | Concealed | Platform Independent | N/A | Integrated | Control Method | behaviour | Host | Reactive | Cost Insensitive |

"N/A" Indicates that a surveyed work is not concerned with a dimension of our taxonomy.
"–" Indicates that an adaptive mechanism is not security explicit, however it implicitly considers and supports security.
"*" Indicates a generic adaptive security solution with no explicit links to any specific security features (e.g. integrity, confidentiality etc.).

### 2.4.1 Security goals

Tackling invasive behaviour has been identified as the most common security focus (32%) among self-adaptive security systems (Figure 2.2). The majority of the surveyed work focuses on the detection and mitigation of generic security attacks [57], [58], [49], [59], [47] with few exceptions that aim at stopping [50] and containing [48] malware binaries [66], code injection attacks [76] and SOAP/XML/SQL threats [55].

Integrity has been identified as the second most prominent security goal, holding the 24% of the surveyed work. The encountered work is primarily concerned with adaptive authentication strategies [51], [12], [56], [47] apart from the work of [52], [20] which focus on trustworthiness and integrity respectively.

Confidentiality has been also extensively addressed by security software engineers (22%). A majority of the encountered work is heavily associated with adaptive authorization mechanisms [53], [12], [54] and encryption schemes [45].

Following, availability gained limited attention (10%), showing a tendency towards omitting warranting the continuous access of data/services to users. The current work on availability restricts its focus on DoS attacks [39], [64] and system errors [77] which are only some of the threats that can harm availability.

Finally, 8% of the surveyed work is associated with generic security (security solutions

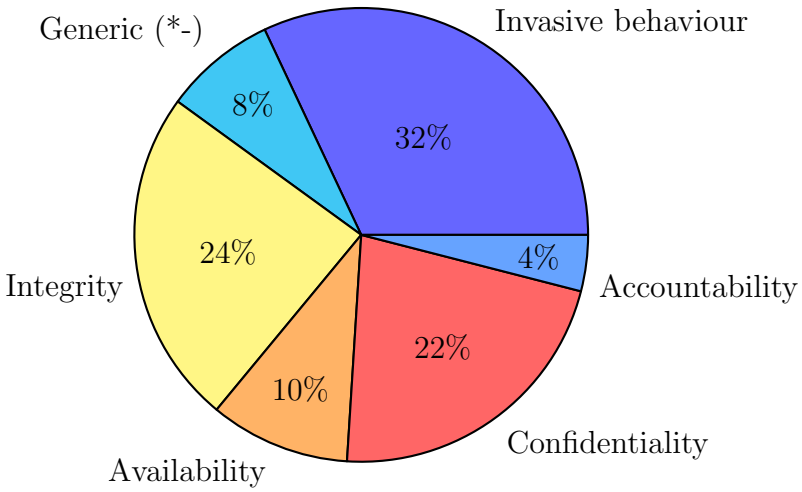


Figure 2.2: Security goals of the surveyed work.

with no explicit links to any particular security goal), where accountability is neglected by a vast majority of the existing solutions (4%).

Despite the focused attempts on goal-specific security, the existing solutions are limited as they have not been engineered with ultra-large environments in mind. Though they are highly adjustable and dynamic, they are not able to manage the varying nature and composition of the threats that might encounter in open environments. To engineer more dynamic and autonomous solutions it is necessary to realize that security cannot be achieved with the usage of limited/fixed number of pre-installed security mechanisms and countermeasures. Based on environmental changes, different security mechanisms of diverse nature must be deployed for securing users, assets and software systems. The Service Oriented Architecture (SOA) paradigm can promote a dependable solution, that will enable adaptive security systems to support diverse security goals via the discovery and selection of different security services at runtime. Despite that SOA can be used as a candidate solution to the problem, it can also introduce privacy-related threats from the side of the service providers. Therefore, security software engineers must develop methodologies that will preserve the privacy of users/systems from service providers in the occasion where they act maliciously. This can be achieved with the introduction of authorization mechanisms that will allow users/systems to enforce restrictive policies for service providers, forbidding them to perform unwanted actions.

Furthermore, security engineers should conduct a fine-grained analysis of the "adaptation boundaries" of their solutions, in terms of what are the acceptable adaptation actions that their systems can perform when operating unsupervised, which is currently overlooked by the existing work. The problem occurs in cases where "extreme" security measures are considered and executed by self-adaptive systems to protect a system/user/asset. These measures often come at a high cost as they sacrifice other features (e.g., availability). Consider the example where a user is connected to a compromised network and it is facing multiple threats with no available countermeasures. One "extreme" measure that a self-adaptive system could consider is to disconnect that user from

the network for some time to protect him, at the cost of his/her availability. Despite the cautious nature of the certain action; however, it can be perceived by a user as a non-dependable/irrational action. Therefore, software engineers should carefully determine what adaptation strategies are appropriate yet customizable by individual users before deploying them.

### 2.4.2 Control Topology

Our findings concerning the control topology (Figure 2.3) of the surveyed solutions illustrate a tendency towards centralization (57%) [38], [52], [53], [12]. We have observed that all the solutions that are grounded on centralised architectures operate in more static and controlled environments.

Following, semi-decentralised architectures have been identified as the second most preferable control topology (31%) for self-adaptive systems [64], [40], [72], [65] illustrating that a significant number of existing work considers elasticity and dynamism to an extent.

Finally, hierarchical (7%) and decentralised architectures (5%) identified as the less dominant architectures. We have noticed that hierarchical architectures are often used for Ad-hoc and mobile networks [56], [47], whereas decentralised architectures are often used for the Cloud [48], [51].

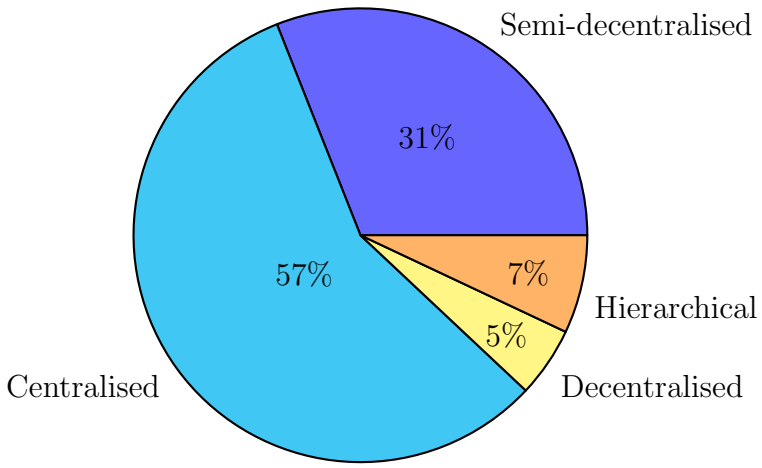


Figure 2.3: System architecture designs of the surveyed work.

The control topology preferences of contemporary solutions signify various limitations. Centralised architectures can be limited or unsuitable for open and ultra-large environments due to their unpredictable and elastic nature. Centralised architectures are ineffective as they introduce a single point of failure, processing and coordination, which damages performance and restricts users from handling their own security and assets in ways that they seem fit. Additionally, centralised architectures are not capable of the concurrent management of large numbers of heterogeneous user requests due to the high computational overhead imposed to a single processing unit. For the aforementioned reasons, security engineers should further explore decentralised and hierarchical architectures to engineer more secure and efficient self-adaptive solutions. To another extent a better, yet more complex approach is to engineer adaptable architectures to allow self-adaptive security solutions to switch between different architectures according to changes on runtime security. An example of such work is the paper of Kong et al., [56] which proposes a system for UAV-MBN networks that can switch between two infrastructures based on the availability of key nodes. A major challenge for the engineering of adaptable architectures is the transitioning between architectures, which is an extremely complex, time-consuming and costly procedure [78]. Transitioning methodologies should operate in an efficient manner while warranting the regular operation of self-adaptive systems throughout the transition.

### 2.4.3 Component Composability

Figure 2.4 illustrates that a vast majority (i.e. 92%) of the surveyed solutions do not facilitate any mechanisms for composing security mechanisms at runtime. Whereas, only 8% of the examined solutions are able to compose security mechanisms. We have observed that all the solutions capable of composing components operate in open, ultra-large environments, such as mobile networks [67] and the Cloud [38], [48].

In some occasions, using a single security mechanism might be inadequate for securing a user/system. Such occurrences can be extensively witnessed in open, elastic environ-

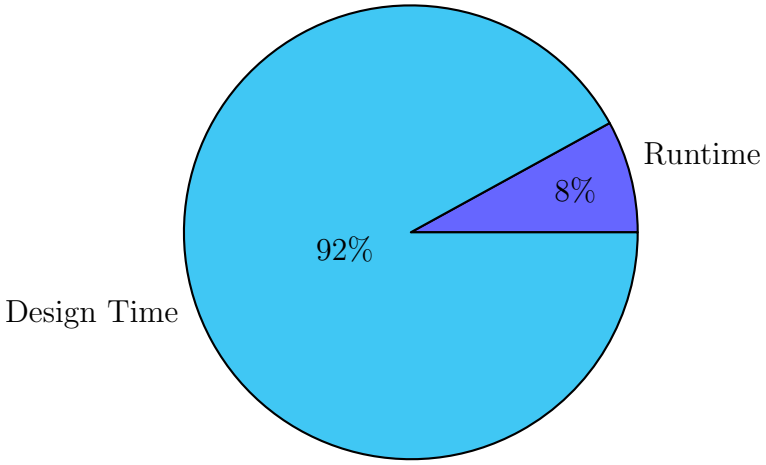


Figure 2.4: Component composability capabilities of the surveyed work.

ments, where unforeseen threats of diverse nature are frequently encountered. In these cases, it is needed to compose and orchestrate a bundle of security components to mitigate runtime threats. Therefore, security engineers should transition from the trivial deployment of sole security components to the deployment of multiple security components of diverse nature.

A key challenge for component composability is to ensure that different components do not maintain conflicting goals. In the case where trade-offs exist between security components, the operations of a system can be challenged or even break. To better illustrate this challenge, consider the example where a user wants to deploy a context-aware security solution to detect malicious network traffic. Now consider that an anomaly is detected and the self-adaptive system deploys SSL/TLS encryption to protect data from leaking. Once the SSL is deployed, the context-aware mechanism can no longer function properly due to the encrypted traffic that restricts it from accessing the data. Thus, the overall stability of a user's security is damaged. To avoid such occurrences, trade-off analysis mechanisms should be used to examine the compatibility of newly introduced security components to the operating environment. A method to achieve this is the use of symbiotic simulation [79] in which administrators can test the compatibility of new components in a simulated, risk-free environment prior to their deployment in the real

system. Significant work exists on trade-off analysis, though, it is concerned with the analysis of generic, security implicit trade-offs [80], [81], [82], [83].

### 2.4.4 Component Dependency

A majority (i.e. 81.6%) of the surveyed self-adaptive security systems enforce a high degree of control and dependency between the security components comprising them (Figure 2.5). To our surprise, some of these solutions were designed for Cloud computing, which are environments that necessitate the deployment of dynamic solutions. Despite the co-depended nature of the security mechanisms comprising the identified Cloud-based solutions, they were still able to effectively operate due to the static nature of their operations, such as integrity attestation [52], access control [53] and policy enforcement [12].

On the contrary, semi-controlled and autonomous methodologies acquired limited attention, reaching the 10.53% and 7.87% of the surveyed work respectively. The obtained results illustrate that these methodologies were employed by solutions applied in highly elastic and dynamic environments such as Ad-hoc networks [56], [39] and Service Oriented Architectures [48], [20], [38].

As ultra-large environments undergo significant runtime changes, they necessitate

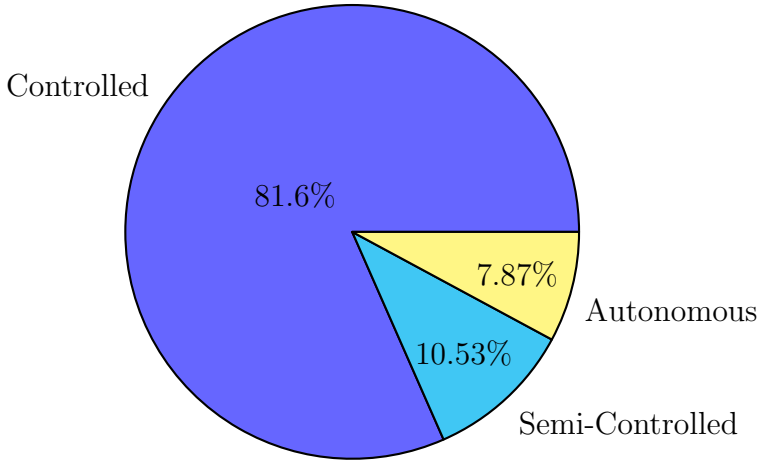


Figure 2.5: Level of dependency between the security components of the surveyed solutions.

the deployment of different security components to fulfil their changing security needs. Therefore, it is essential that security components are not highly correlated to each other or the host systems so they are easily removed and replaced. Security components should be treated as add-ons with no specific composition to allow their effective integration in adaptive security systems.

### 2.4.5 Component Discoverability

Existing self-adaptive systems (i.e., 90%) favour security that emerges from pre-installed security mechanisms that are instantiated at design phase (Figure 2.6). These solutions can prove limited as they may fail to face runtime threats of diverse nature due to the use of pre-defined countermeasures. Only 10% of the reviewed solutions were able to discover components that are situated at different locations at runtime. These solutions were mostly deployed in service oriented architectures [38], [20] and distributed environments [40], in which it is crucial for users and software systems to switch between different security mechanisms to meet their runtime goals.

Security engineers should further consider the development of suitable mechanisms for the discovery of security components/mechanisms, to ensure the secure operation of

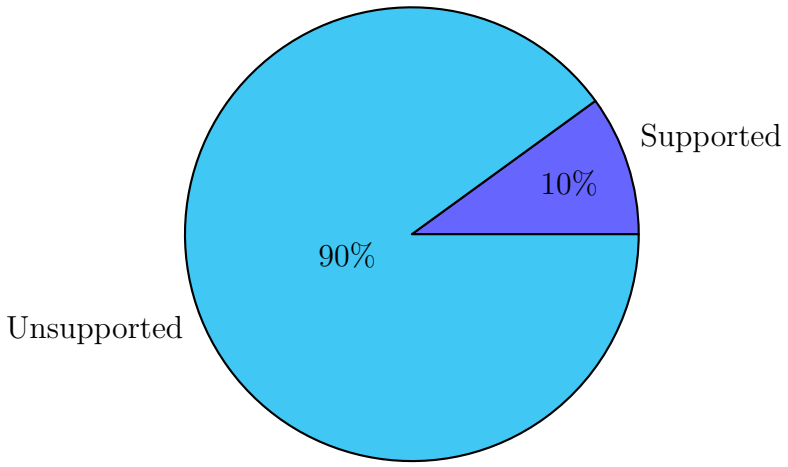


Figure 2.6: Component discoverability capabilities of the surveyed work.

their systems/users in the face of runtime threats. A way to achieve this is through the construction of a public record that will maintain information concerning the location of different security components that can be provided by various third party providers, along with appropriate interfaces that will enable self-adaptive security systems to use them. By doing this, it is possible to warrant the deployment of state-of-the-art security, as this procedure heavily relies on various security professionals to provide their security components. Despite the effectiveness of this solution, it is a challenging undertaking as these mechanisms need to be grounded in platform-independent technologies to allow various systems to utilise them without the need for special mechanisms. Furthermore, as these mechanisms need to cater for the needs of a large number of users, provisioning and distribution mechanisms should be used to ensure the fair and dependable sharing of resources between users/systems. From the legal point of view, all the entities involved in this environment should be bounded by legal agreements concerning the usage of the resources/services.

### 2.4.6 Operational Transparency

Despite the vast acknowledgement from security software engineers that "security through obscurity" is neither effective or a good practice, the surveyed work (Figure 2.7) demon-

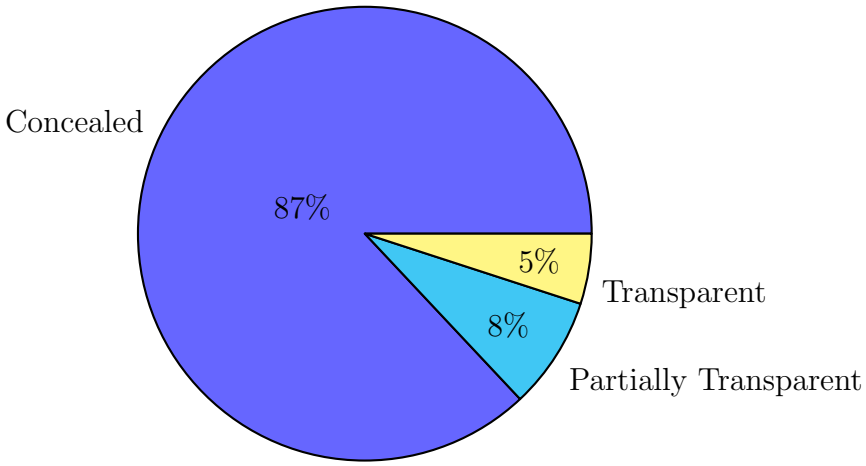


Figure 2.7: Operational transparency of the surveyed work.

strated that a vast majority of the self-adaptive security systems (i.e., 87%) still follow this paradigm. Only 8% of the surveyed solutions considered operation transparency at some extent [51], [38], where just 5% of them promoted complete transparency on their operations [48], [20].

We have observed that the solutions offering some degree of transparency were designed for service oriented environments, where users are concerned with how their data and security is handled due to their association with multiple providers and third parties. Operational transparency should be further explored as it can promote trust between users and systems [84], [85], [86]. A way to promote operational transparency is by deploying logging mechanisms to monitor, record and display the actions performed by an adaptive system, including the processes used for handling user data. Based on the recorded data, a user should be able, if required, to establish policies that can restrain a self-adaptive system from performing unwanted/unnecessary actions.

### 2.4.7 Design Centricity

74% of the reviewed solutions were designed for protecting software systems [70], [72], [71], [74] (Figure 2.8). Whereas, 18% of the solutions are concerned with the security of users [54], [61], [46], [67]. Finally, only a small number (i.e. 8%) of the encountered work is concerned with securing individual assets [69], [68].

The work conducted on asset-centric security is still in infancy. Security software engineers should further explore cost-effective, elastic mechanisms to allow self-adaptive security solutions to handle diverse and complex requests that necessitate the enforcement of different security policies for different types of assets. Added, security engineers need to re-define assets and the assumptions describing them in the Cloud to better identify what needs to be protected and how.

Furthermore, the correlation between system-centric and user-centric methodologies can be further exploited. As monitoring and controlling ultra-large environments as a whole is a difficult undertaking; it is possible to follow a holistic approach by employing

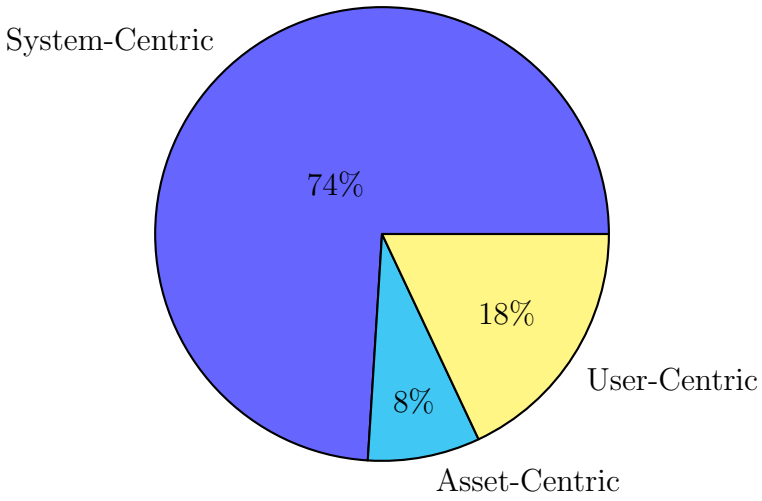


Figure 2.8: Design centricity of the surveyed work.

user-centric methodologies to monitor and control the behaviour of individual entities in a localised fashion and then orchestrate these methods to protect the overall system from malicious behaviour. By using this approach, it is possible to effectively secure a system as a big part of the computations and decisions are performed in a decentralised fashion.

### 2.4.8 Platform Dependencies

The vast majority (i.e. 97.36%) of the surveyed solutions (Figure 2.9) are platform independent, with the exception of the work of Hsieh et al., [47] which is specifically applied on the LEACH algorithm.

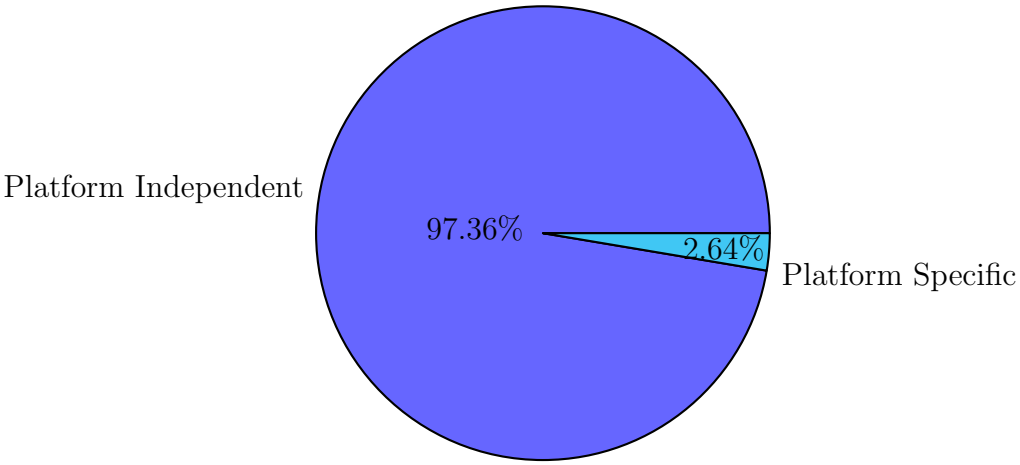


Figure 2.9: Platform dependencies of the surveyed work.

The results are encouraging as they signify that security engineers have made an extensive effort to make their solutions applicable to a vast majority of the application environments in existence by switching from archaic, platform-specific systems to platform independent solutions.

### 2.4.9 Elasticity

A significant number of the examined solutions (92%) did not consider any dimensions of elasticity (Figure 2.10). Only three Cloud-based self-adaptive security systems considered elasticity, from which two exhibited sub-linear elasticity [38], [48] and one linear elasticity [20].

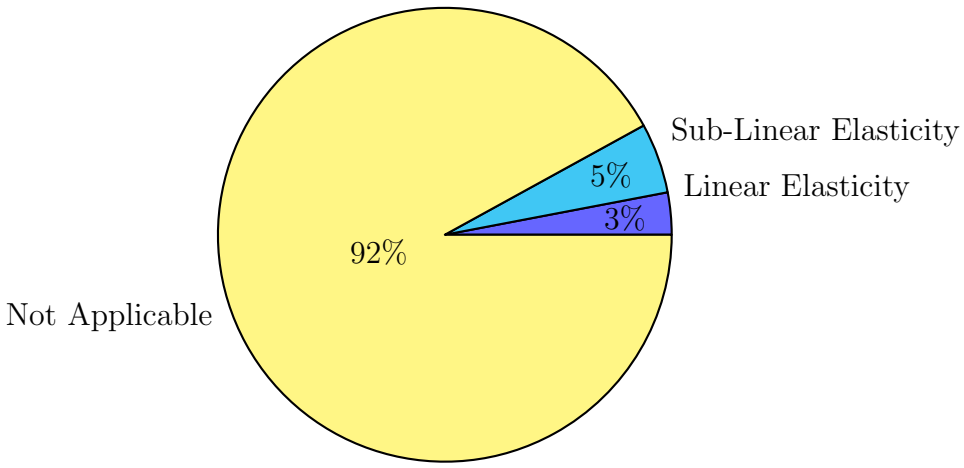


Figure 2.10: Elasticity of the surveyed work.

Despite that elasticity is one of the main characteristics of ultra-large systems, however, the existing implementations are limited in dealing with this dimension, making these solutions unsuitable for open environments. In the case where a solution cannot scale, it is possible that it can be challenged or even break when applied in ultra-large, high-demand environments. Therefore, it is necessary for self-adaptive security solutions to be assessed against various workloads to determine their elasticity, performance and suitability for ultra-large environments. Even by doing this, challenges remain as security

software engineers can determine the breaking point of their solutions but they cannot accurately speculate the varying demand and its implication on security. However, testing scalability from the security point of view is essential for anticipatory and proactive security solutions.

### 2.4.10 Security Mechanism(s) Deployment

The acquired results (Figure 2.11) signify that modern self-adaptive security solutions prefer the usage of integrated security mechanisms (i.e. 84.21%) compared to the usage of external/add-on security mechanisms. Only 13.16% of the reviewed solutions use add-on security mechanisms, where just one solution (2.63%) uses a hybrid methodology [38].

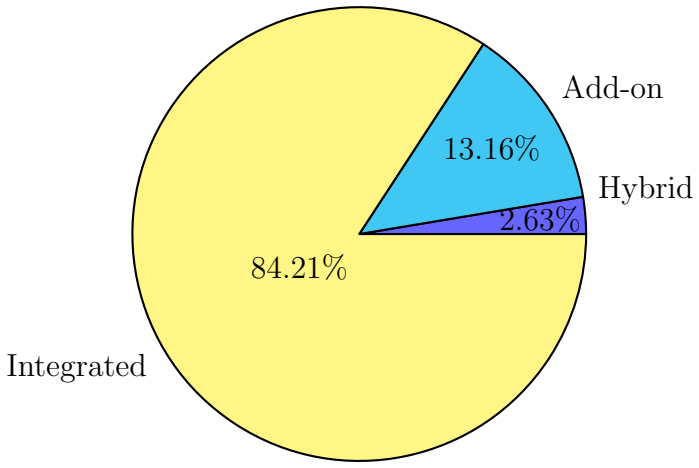


Figure 2.11: Security mechanism deployment of the surveyed work.

We have observed that all the encountered work that uses add-on security mechanisms is applied in mobile environments [49], Ad-hoc networks [39] and service oriented architectures [20], [48] which often necessitate the deployment of different security mechanisms for treating heterogeneous runtime threats. On the contrary, the surveyed solutions found to use integrated security mechanisms, are more static and do not require the use of additional security mechanisms. A few examples of such solutions are trust management frameworks [51], integrity attestation systems [52], authorization techniques [53], [54],

policy execution methods [12] and adaptive firewalls [42].

We view security as a set of varying goals, that cannot be satisfied with the use of static, pre-installed security mechanisms and countermeasures. However, a majority of the existing systems ground their solutions on pre-installed security. To overcome this limitation, contemporary solutions must deploy mechanisms that will enable them to access and deploy different security "components" that are situated at different locations via programmable interfaces. Thus, making them more elastic and dynamic, capable of tuning their security capabilities according to runtime threats.

### 2.4.11 Adaptation Inspiration

A majority (i.e. 87%) of the surveyed systems base their adaptation mechanisms on the classical paradigm of control methods (Figure 2.12), which often consist of four phases: monitor, analyse, process and execute. Whereas, 10% of the reviewed solutions ground their adaptation on bio-inspired methodologies (e.g., [42], [64]). The remaining 3% explore economics inspired methodologies for security (e.g., [20]). Finally, none of the surveyed work has exploited nature inspired methodologies for security.

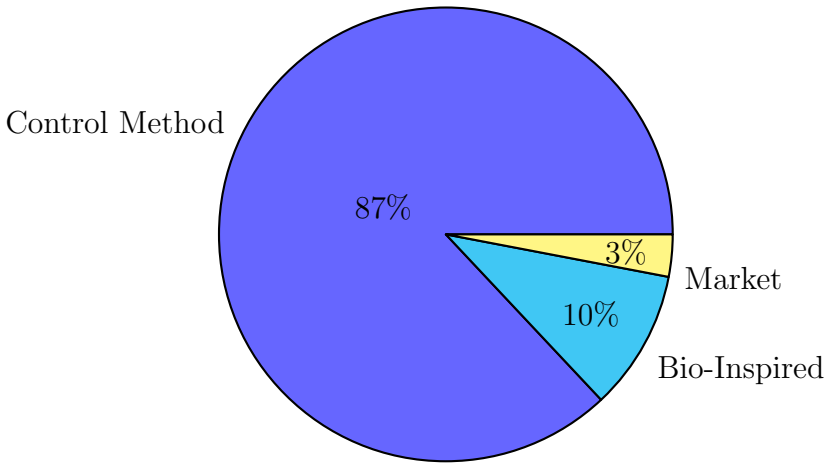


Figure 2.12: Source of inspiration for the adaptation mechanisms surveyed.

It is imperative that bio-inspired, nature-inspired and economic-inspired adaptation techniques are further explored, to advance the current solutions from the classical

paradigm of control methods to more dynamic and elastic methodologies. Software engineers can leverage economic-inspired methodologies, promising "efficient" and "light" optimisation mechanism for the continuous satisfaction of varying security requirements, as they are considered to be an efficient solution to dynamic allocation problems [87],[88], [89], [90], [91], [92]. The economic-inspired models can promote transparency in the way services and resources are traded as their operations are founded on systematic procedures; henceforth, promote trust between self-adaptive systems and users. Furthermore, the decentralised nature of these methods promote the development of more cost-effective and elastic frameworks as it: i) eliminates the single point of failure, ii) allows users to handle their security requirements and data and iii) simplifies the concurrent management of multiple user requests, as a major part of the computations and decisions are performed in a decentralised manner and iv) allows service providers and users to make their own decisions for maximising their gain.

### 2.4.12 Adaptation Awareness

82% of the surveyed systems monitor (Figure 2.13) contextual data (e.g., location) to inform their adaptation process, where the remaining 18% of the work considers the behaviour of a user and/or a system for adaptation. We have observed that the solutions that ground their adaptation on behavioural information tend to be concerned with the behaviour of software systems [70], [73] and not the behaviour of users.

Choosing between contextual and behaviour driven adaptation and determining the optimal set of attributes for informing adaptation is a key challenge [93], [94], [95]. This is a complex decision as security engineers need to ensure that their systems can thoroughly monitor the behaviour and/or context of the entities that are protecting while minimising the computational overhead that can emerge from monitoring and analysing a large number of attributes. As any attempt for selecting a pre-defined set of attributes for informing adaptation can prove ineffective in the long-run, it is possible to use mechanisms that will enable self-adaptive security systems to switch between various sets of

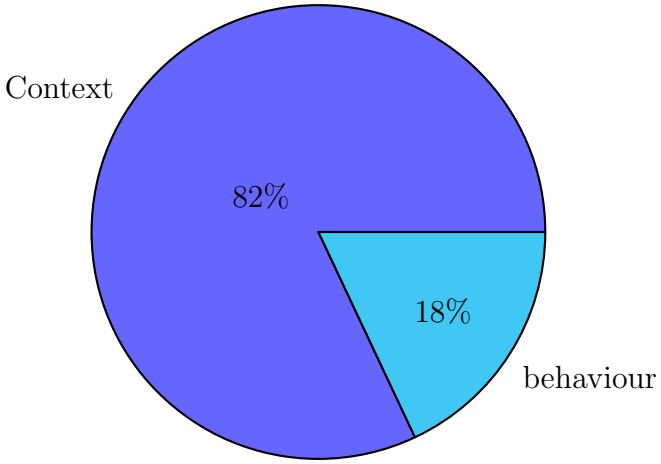


Figure 2.13: Adaptation awareness of the self-adaptive mechanisms surveyed.

monitoring attributes according to the changes in the runtime security requirements. A way to achieve this is by using symbiotic simulations [79], in which administrators can test different sets of attributes for identifying the optimal set for adaptation.

### 2.4.13 Adaptation Layer

Network layer has been identified as the dominant layer for adaptation (i.e., 37%) [39], [56], [57]. Followed by the service layer with 24% [51], [52], [53], [48]. Lastly, application [46] and host [40], [69] layers have been identified as the most neglected layers possessing 22% and 17% of the surveyed solutions respectively. The results (Figure 2.14) demonstrate that engineers have well acknowledged open and elastic environments as well as the need for adaptable security in these environments.

Security solutions to a big extent have looked at securing explicit layers which is limiting as it may be necessary to adapt/manipulate multiple layers in order to satisfy changes in security. The challenge is to secure multiple layers and to provide effective self-adaptive solutions covering these layers. This can be achieved by managing the explicit link between low-level (e.g. computational resources and hardware components) and high-level layers (e.g. online services and applications). This link is crucial as it captures the computational overhead that a higher layer could impose on underlying layers as well

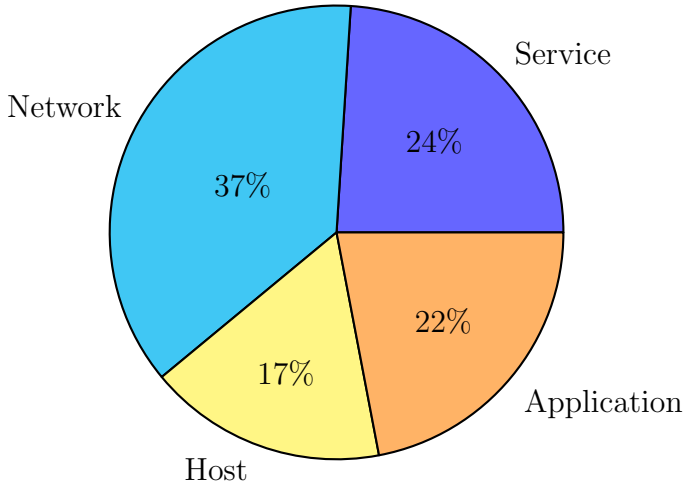


Figure 2.14: Adaptation layer of the surveyed mechanisms.

as the security that a solution could deploy.

### 2.4.14 Anticipatory support

Figure 2.15 illustrates that a large number of the reviewed systems (i.e., 92%) do not facilitate any mechanisms for forecasting their future state and the threats that might encounter at runtime (operate in a reactive manner). Only 8% of the examined solutions facilitate mechanisms for the anticipation of runtime threats. Examples of such work are reported in Foo et al., [66] which perform containment of intrusions in distributed e-commerce systems, the work of Abie et al., [49] which estimates and predicts system benefits and risks in IoT. And the work of Farid et al., [57] which promotes a proactive network intrusion detection system.

Despite the extensive usage of reactive mechanisms for satisfying functional requirements, reactive security is a difficult task to achieve. This is due to the time-critical and proactive nature of security, which does not tolerate delays. These delays rise due to the time required by reactive mechanisms to discover, select and deploy countermeasures, which can, in turn, cause the compromise of a system/user/asset. To avoid adaptation delays and engineer more efficient solutions, self-adaptive security systems should deploy proactive mechanisms to anticipate and mitigate runtime threats before they are mani-

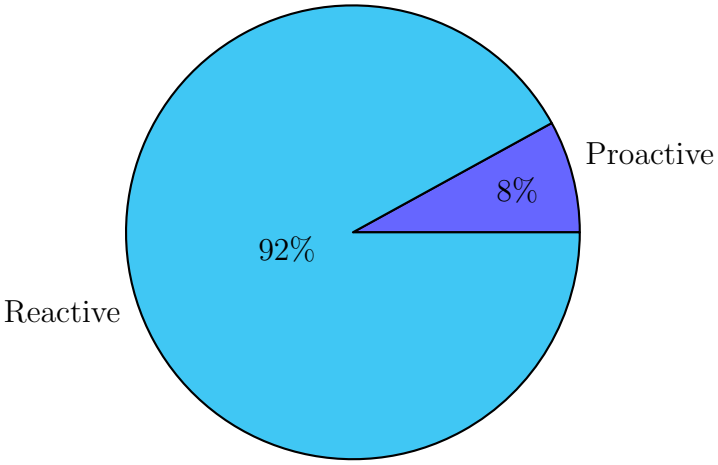


Figure 2.15: Ability of the surveyed adaptive mechanisms to anticipate threats.

fested. A key challenge that should be addressed when designing proactive methodologies is the adaptation frequency. As adaptation is expensive, it should only be triggered when critical events occur. The frequency of adaptation must be well-defined and scheduled, as any delays in the adaptation, can compromise a system's resilience. Whereas, continuous adaptation can cause operational problems due to high computational overhead which can lead to resource starvation and system errors.

### 2.4.15 Cost Sensitivity

77% of the surveyed solutions (Figure 2.16) have omitted considering the mitigation of potential trade-offs that can occur when adapting security. Just 15% of the reviewed systems are concerned with exploring the relationship between security and computational overhead, where the trade-off between security and time received less attention (5%). Finally, monetary constraints were almost overlooked by the surveyed work, reaching 3%.

Mitigating possible trade-offs in ultra-large environments is crucial, as the costs of adapting security may overtake its benefits. Examples of existing attempts for trade-off resolution is the work of Zou et al., [62] and Du et al., [52] for mitigating the trade-

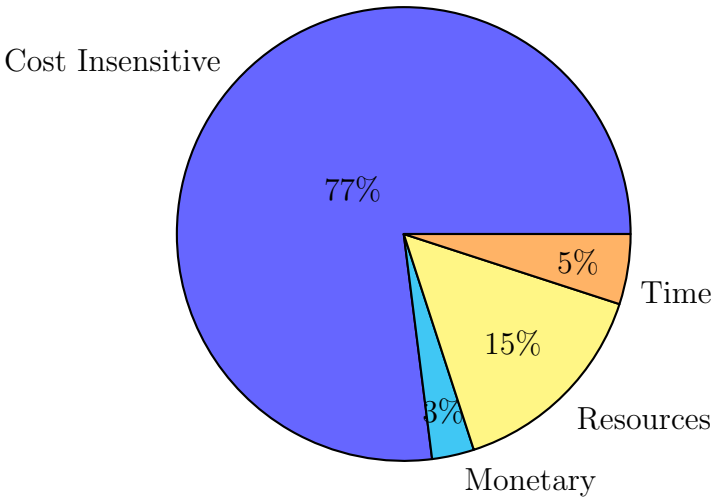


Figure 2.16: Cost sensitivity of the surveyed solutions.

off between security and time in networks and the Cloud respectively. The work of Tziakouris et al., [20] for monetary constraints in the Cloud and the work of Li et al., [51] and Chigan and Ye [39] for mitigating resource consumption in the Cloud and Ad-hoc networks respectively. Despite that some of the existing work has addressed potential trade-offs that can emerge from adapting security, they are still at infancy. Security engineers need to further examine the costs associated with security and to promote more cost-effective security solutions.

## 2.5 Summary

Our analysis of the past and ongoing research on self-adaptive solutions has revealed a notable advancement towards adaptive and autonomic security for ultra-large and open environments, including the Cloud; however, it also illustrates the gaps, limitations and challenges that need to be further explored. The analysis of our results has provided us with guidance on commonly used security mechanisms, design principles and strategies along with their strengths and pitfalls. By applying this knowledge in the Cloud it will be feasible to engineer more elastic and dynamic security solutions.

In particular, we have observed that in order to stay ahead of today's evolving threats

and allow the effective application of self-adaptive security solutions in open and elastic environments and more specifically the Cloud, a number of outstanding research issues need to be addressed as part of this thesis:

- Design Cloud-based security solutions that are grounded on fundamentally elastic architectures to promote solutions that can scale to the varying demand for service.
- Deploy learning techniques to make Cloud-based security solutions proactive for identifying and mitigating runtime threats prior to their manifestation.
- Engineer security mechanisms that will allow Cloud-based solutions to enforce different ad-hoc security policies for different assets, to advance from aggregated security to asset-centric security.
- Consider the trade-off between security and the cost of a solution.
- Further explore the link between services and underlying computational resources.

# CHAPTER 3

# MARKET-INSPIRED FRAMEWORK FOR SECURING ASSETS IN THE CLOUD

This chapter looks at market-inspired techniques as a candidate solution to the challenges posed by our research questions in Chapter 1 and our taxonomic findings in Chapter 2. As part of the solution, this chapter outlines an architecture which draws inspiration from market-inspired methodologies and learning algorithms. The proposed solution makes an analogy between markets and their usage for securing assets at runtime in the Cloud. We then provide a conceptual architecture model for our framework comprising the entities and operational phases of our solution. Consequently, we assert the applicability and effectiveness of our framework by instantiating and developing a variant of our system based on a simulated university application environment, facilitating Cloud storage and Voice-Over-IP (VOIP) services. [2]

[2]Part of the work presented in this chapter has been published in [20], [98] and submitted for publication in [99].

## 3.1 Self-Securing Assets In the Cloud by Combining Market and Learning

### 3.1.1 The Use of Market in the Self-Securing Framework

Market-oriented methodologies have been widely employed by software engineers in the Cloud for solving dynamic allocation problems. Examples of such work are the paper of Jiayin et al., [17] on preemptable resource scheduling; the work of Shin and Akkan [18] on resource management in IaaS; the paper of Lai and Chang [88] on low-latency high-efficiency resource allocation mechanisms; and the research of AuYoung et al., [91], Stoica et al., [96] and Waldspurger and Weihl [97] for managing excessive demand for service.

The extensive usage of market methodologies in the Cloud can be mainly attributed to their potential to support decentralisation in architecture and decision making. This is known to promote more dependable self-adaptive security systems. Infusing the architecture with market approaches can i) eliminate the single point of failure, ii) enable users to handle their own security requirements and data and iii) simplify the concurrent management of multiple security requests as a major part of the computations and decisions are performed in a decentralised manner. Moreover, market solutions allow both users and providers to make their own decisions for maximising their utility and regulate the supply and demand of services and resources at market equilibrium. In the presence of limited resources, auction mechanisms can promote the effective allocation of services and resources by prioritising security requests based on their criticality (reflected in the bidding prices); hence, ensuring their provision to users that face an imminent threat. We assume that users have a good understanding of their security requirements and data to select appropriate bidding prices for their assets. In the case that a user is not able to identify suitable prices for its assets it can automate the process by allowing the machine learner to generate a bidding price based on the historical bidding data of other users. The use of market enables users to express their preferences concerning the type

of resources required and the time they want to acquire them, hence allowing a system to manage excess demand by spreading it out over time [91]. Furthermore, market models can promote transparency in the way services and resources are traded and mapped to security requirements/goals as their operations are founded on a systematic procedure, henceforth encouraging trust between SPs and users.

Despite the growing work in the area of market-oriented Cloud computing, a majority of the existing methodologies have not explicitly address any dimensions of security. Our work exploits market-inspired methodologies and demonstrates their fit for the effective delivery of security in the Cloud. Given the multitenant and shared nature of Cloud systems along with their ultra-large scale, any security solutions for these environments should be grounded on fundamentally scalable architectures to ensure the continuous satisfaction of the varying security requirements of multiple assets.

Our work perceives the Cloud as the 5 $^{th}$ utility [100]; an ultra-large marketplace with shared, on-demand services and resources that are traded in the same manner as traditional utilities. These services and resources can serve the changing security goals of assets in dynamic and shared environments. Therefore, assets can be secured with the discovery and selection of appropriate services and underlying resources, with the assistance of market mechanisms, that can best satisfy their changing security goals and constraints.

In particular, a goal-oriented methodology can be followed. Each security goal (e.g. text file security) can be further decomposed into a set of security sub-goals, such as anonymity, integrity, availability, etc. which can then be mapped to suitable services and resources that can best support and satisfy an asset's security sub-goals and goals respectively (Figure 3.1). It is possible for us to use any goal-oriented methodology in our framework but as our work is not concerned with contributing to the goal-oriented requirements engineering literature, we have adopted the notion of goals from the work of Lamsweerde [101]. The work of [101] defines goals as the objectives that a system must achieve. These goals refer to intended characteristics that are required by a system.

Goals can be formed at various abstraction levels, ranging from high-level, strategic concerns (e.g usage of high-level security to protect a document) to low-level, technical requirements (e.g. delete a classified document after two wrong password entries).

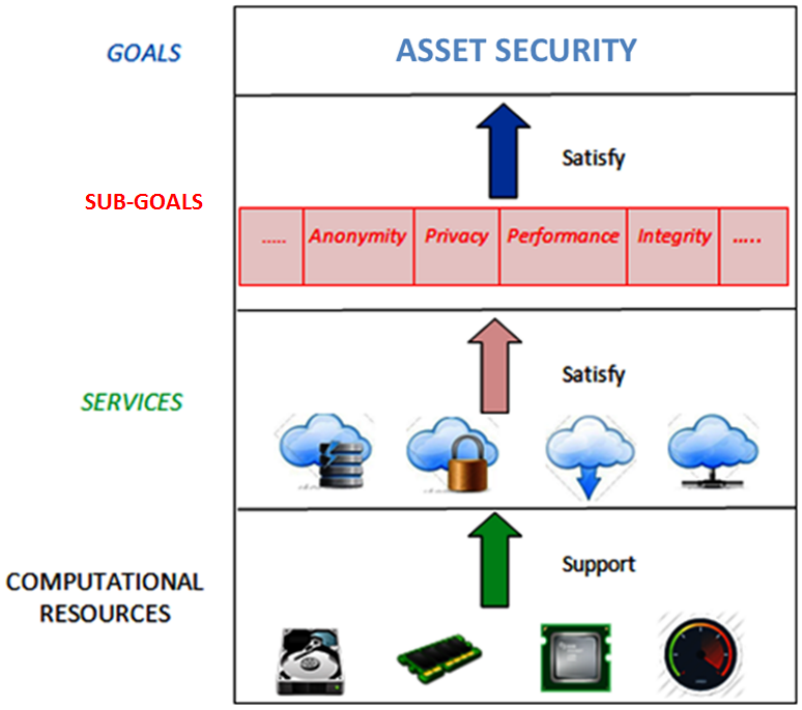


Figure 3.1: The link between goals, sub-goals, services and resources.

### 3.1.2 The Use of Learning in the Self-Securing Framework

The ultra-large and dynamic nature of market-oriented Clouds in conjunction with the vast search space of homogeneous candidate security solutions make the selection of dominant services and resources a difficult and expensive (in terms of time and resources) undertaking. One way to make markets more effective, hence allowing bidders to quickly identify and select suitable security solutions, without re-defining auctioning algorithms, is to use machine learning techniques to arrive at more efficient bidding plans, informed by historical data. Instead of entering bidders into exhaustive search for the identification of candidate security solutions, per asset, the learning can assist in the formulation of ad-hoc security strategies via the analysis of historical bidding records. Moreover, learning

can significantly decrease the bidding frequency of a user, when applicable, by identifying occasions where adaptation is obsolete based on the outcomes of past incidents, thus lowering the computational overhead associated with adaptation.

Although the learning approach can be an effective optimisation tool for market-oriented Clouds, security engineers have omitted considering them for security. The existing work is concerned with the deployment of learning approaches in simplistic electronic markets with no explicit links to security. Example of such work is the paper of Zhang et al. [102], which uses Markov random fields to model the payment transactions on eBay for classifying users as honest or fraudsters. The work of Balcan et al. [103] for reducing the profits of bidders arising from the selection of dishonest bidding approaches in revenue-maximising incentive-compatible mechanisms with the use of sample complexity techniques. The paper of Hummel and McAfee [104] on improving the performance of advertisers by incorporating active learning into a machine learning system for online auctions.

To optimize the proposed self-adaptive security solution we have used the Random Forest classifier [15] to automate security and alleviate the additional resource overhead imposed by repetitive bidding for the identification of candidate security solutions. The deployed learner examines past bidding strategies originating from recorded runtime threats to effectively identify appropriate countermeasures (i.e. services and resources) while catering for computational costs.

In spite of the benefits arising from the use of learning approaches in online markets, disadvantages and challenges exist. In terms of costs, learning approaches can significantly increase computational overhead. This is due to the need for recording and continuously updating user historical data, as well as training and deploying learners for the analysis of recorded data. Moreover, it is not always possible to completely automate security via the usage of learning algorithms due to various reasons, such as insufficient or inconsistent training data. In these occasions, it is necessary for users to switch to semi-autonomic or user-controlled adaptation and provide manual input for the counter-

measures they wish to deploy.

## 3.2 Conceptual Market-Inspired Architecture

This work considers markets to be regulated environments where several buyers and sellers can join / leave the market at any time for purchasing and trading services and resources respectively. The market can operate with different auction mechanisms according to the runtime requirements of bidders / sellers and the operating environment itself (e.g. available resources, number of buyers, etc.). Different auctioning mechanisms can introduce different benefits and limitations to the operating environment. We assume that market auctioneers are able to monitor the behaviour of sellers and buyers in their markets, including available resources, bidding prices, etc. to identify malicious behaviour that can damage the interests of legitimate bidders, sellers and the market itself.

The remaining of this section provides a conceptual architecture model describing the proposed agent-based market-inspired security framework including the entities and operational phases comprising it. We demonstrate, at a high-level, how agent-based architectures and market-inspired methodologies can co-exist for effectively securing multiple assets in the Cloud.

### 3.2.1 System Entities Description

The choice of market-inspired approaches has motivated the need for leveraging agent-based modelling as a solution. Our solution exploits agents to model and automate the behaviour of the entities comprising our framework. An agent is a computer software system that can perform autonomous actions, including decisions, for satisfying its design objectives. A multi-agent system comprises a number of different agents that interact with each other. In order for agents to successfully interact, they need to be able to cooperate, coordinate and negotiate. Cooperation is the process where multiple agents

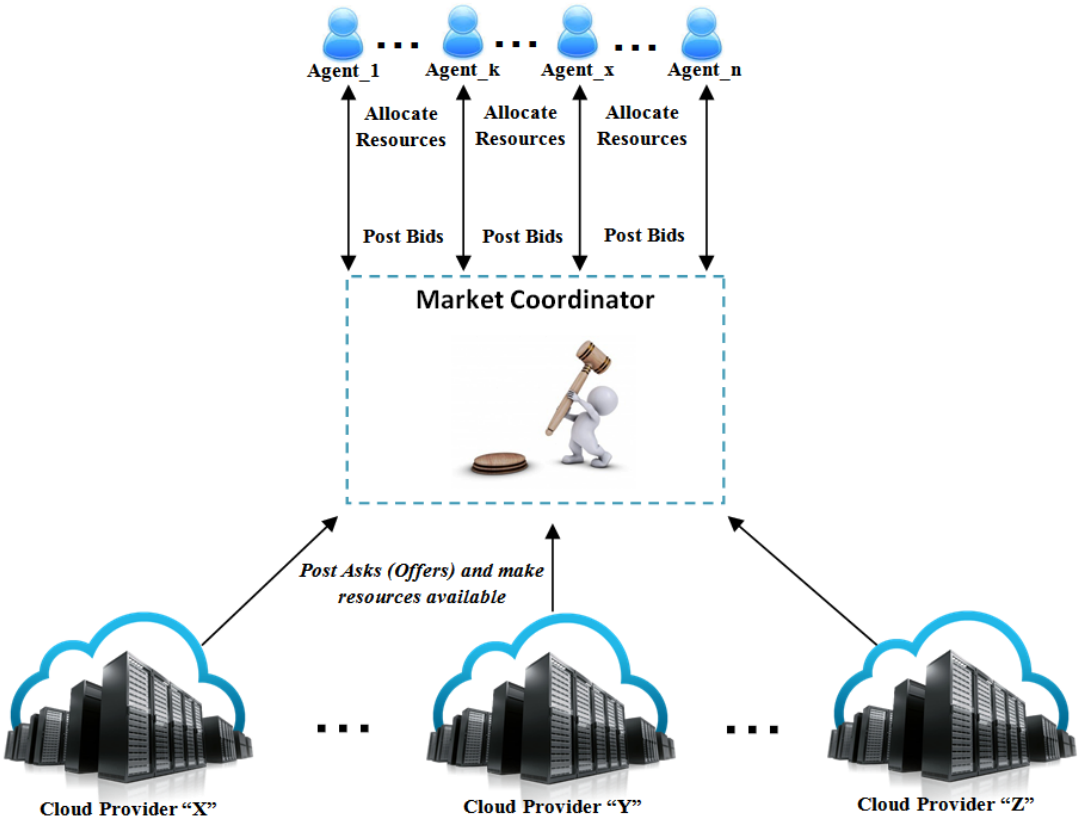


Figure 3.2: The system entities.

operate together to accomplish a common goal [105]. Due to the dynamic and complex operations (e.g. discovery of services, bargaining for services, composition of services, etc.) of the proposed entities we have used agents to automate and simplify their runtime behaviour. The use of agent-based modelling provides us with the complete control over the operations of the proposed market entities, thus allowing us to dynamically adjust their runtime behaviour and interaction patterns for examining various hypothesis and scenarios. Furthermore, the use of agents enables us to test the applicability and effectiveness of our approach on a larger scale, which is an essential aspect of the Cloud. Our conceptual architecture consists of the following agents (Figure 3.2).

#### 3.2.1.1 User Agents

User agents are self-interested, autonomous entities that represent users in the Cloud. A user agent is responsible for monitoring changes in the runtime security goals and

sub-goals of assets and triggering adaptation. In particular, user agents record events that can cause changes in the security of the assets that are in charge of along with relevant contextual and behavioural data. By maintaining data records for each user it is feasible to construct training datasets for informing the learning approach (i.e. Random Forest algorithm) and replicate a user's behaviour to enforce security in an automated and proactive manner. When a runtime event occurs that signifies known malicious behaviour, the adaptation evaluator, facilitated by each agent, examines if the existing services and resources can satisfy the runtime security goals and underlying sub-goal(s) of an asset based on pre-defined adaptation rules. If the assessment illustrates that are inadequate, the agent triggers service-resource adaptation.

To adapt, the Random Forest algorithm examines the recorded historical data of a user to discover correlations between previously encountered threats, including their mitigation strategies, and the existing situation. For the purposes of our prototype system, we have used synthetic training datasets generated with normal distribution. By doing so it was feasible to examine the effectiveness of our solution against a wide number of scenarios while considering occasions that are not often encountered in market-oriented Clouds. The results from the Random Forest algorithm are then used to construct a security specification, called a *bid*, which reflects the refined security goals and sub-goals. The bid is then analysed by the constraint evaluator to ensure that the specified adaptation constraints, such as monetary and computational costs, are not violated by the adaptation plan. In particular, bids consist of the following attributes: i) the name of the asset that needs to be secured, ii) the security sub-goals that require support, iii) the type of the service(s) required, iv) the highest price that a user is willing to pay to secure his/her asset and v) the level of security provision required. Once a bid is formed, it is forwarded to the market coordinator (auctioneer) for auctioning.

#### 3.2.1.2 Cloud Service Providers (SP)

Cloud SPs are vendors that trade their services and resources in the market. They are responsible for publicly announcing their offers (called *asks*) to the market coordinator. The asks comprise a specification of the services and resources that are trading along with the price that they want to sell them on. In the case of a match between an ask and a bid, the SP allocates the required service(s) and resources to the winner user agent in the form of a virtual machine (VM) instance. The allocated instance is formed according to the runtime security goals and sub-goals of an asset. Thus, each VM facilitates different configurations of services (varying in: service type, level of security provision, security features) and resources (varying in: number of processors, memory, bandwidth and storage space) that may be provided by different SPs.

#### 3.2.1.3 Market Coordinator

The market coordinator acts as an auctioneer and market regulator. It implements trading rounds for a market during which it accepts bids and asks from user agents and SPs respectively and performs a match between them via the usage of auctioning procedures. The coordinator is also responsible for overseeing the behaviour of user agents and Cloud SPs, including the allocation of VM instance(s), to ensure the effective operation of the market and protect the interests of all transacting parties.

### 3.2.2 System Operation Phases

To deal with the runtime changes in security and the dynamic nature of the Cloud we have grounded our solution on the widely acknowledged control method, MAPE-K [106]. MAPE-K has become the de facto architecture for developing self-adaptive solutions due to the introduction of the learning/knowledge component which enables software engineers to analyse previous data and based on them to reason concerning future decisions. The MAPE-K control loop comprises a sequence of four computations,

namely: Monitor, Analyse, Plan and Execute which operate over a Knowledge-base.

The wide use of the MAPE-K architecture by existing self-adaptive systems [38], [39], [40] promoted different implementations, including variations where two or more phases are merged/integrated. Although all the phases of the MAPE-K architecture are equally significant, our work gives emphasis to the analysis and planning phases. In which lies our novel contribution for the discovery and allocation of secure services and resources via the usage of learning algorithms and market mechanisms. For the monitor and execution phases, we follow a simplistic approach to model the necessary operations. The simplistic nature of the proposed monitoring procedure encourages us to integrate it with the analysis phase. Whereas, the complex and multifaceted nature of our planning phase necessitate its separation to two distinct phases, namely the bid formulation and bid auctioning phases. Below we provide an abstract description of the operation phases of our system and their mapping to our conceptual architecture presented in Figure 3.3.

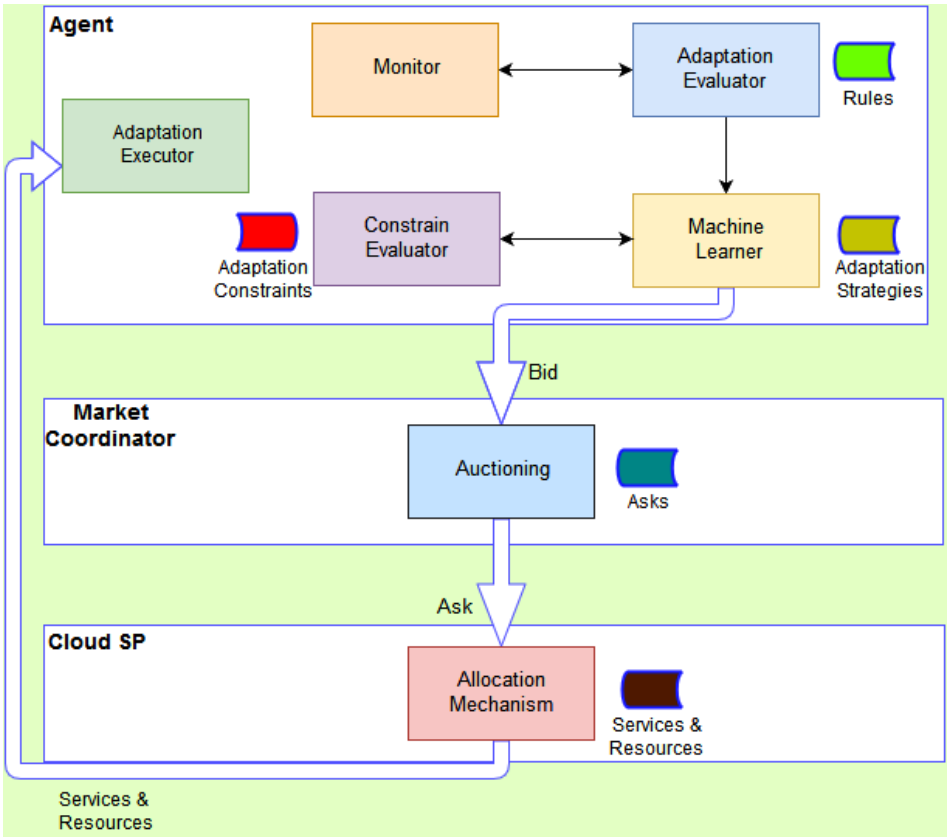


Figure 3.3: The conceptual architecture.

#### 3.2.2.1 Monitor & Analysis Phase

**Monitor**

Work on monitoring already exists (e.g. [83], [107]). Although it is possible to use monitoring from literature, we instead follow a simplistic monitoring approach to monitor the behaviour of bidders and their assets. In particular, our implementation monitors changes in the context (e.g. user location, available resources, etc.) and content (e.g. file modification, etc.) of assets. The recorded information is fed into a watch-dog process called adaptation evaluator that runs as a monitor to depict changes that can affect the security goals and sub-goals of assets. The obtained data is recorded in the knowledge-base to inform the adaptation process. The monitoring attributes informing the adaptation are system specific and should be altered based on the application environment.

**Analysis**

Similarly to the monitoring phase, the analysis phase has received significant attention. Some notable work is the paper of Souag et al. [108] on the analysis of security requirements based on domain ontologies. The work of Li et al. [109] on modelling security patterns as contextual goal models; and KAOS [110], a meta-model for capturing initial requirements for the requirements acquisition process. Despite the existence of substantial work on analysis, we have chosen to develop an ad-hoc monitoring approach due to the various idiosyncrasies and challenges arising from the use of market mechanisms and learning algorithms for security, in the absence of closely related work. During the analysis phase, the adaptation evaluator examines the data recorded from sensors to determine whether the existing services and underlying resources can satisfy the runtime security goals of an asset. The adaptation evaluator contains a pre-defined set of rules describing the occasions where adaptation is necessary. In the case where the existing resources/services are deemed insufficient adaptation is triggered. We adopt a goal-oriented approach to map security goals to their sub-goals. For example, the goal "Online Account

Security" can be further decomposed into authentication, anonymity and authorization sub-goals. Critical changes to the dynamics of one or more interrelated sub-goals can trigger the need for adaptation. As adaptation is expensive, the frequency of adaptation can be determined by considering the extent to which the security goals and sub-goals diverge from the tolerance level.

#### 3.2.2.2 Plan Phase

Although a large number of self-adaptive security systems have used the MAPE architecture, the planning phase has received limited attention. A notable work on planning is the paper of Durfee and Lesser [111] proposing a partial global planning where multiple distributed agents communicate with each other during the calculation of their local plans for making effective decision plans. In the context of our work, the planning phase is a complex, multi-step procedure performed at two different layers. Firstly, it is necessary for each user agent to plan its adaptation strategy locally (i.e. form a bid) and then conduct synchronous planning (i.e. auctioning) with other user agents on a global, system-wide scale for identifying appropriate services and resources. These services / resources can be provisioned by various providers within the Cloud and can be subject to availability.

**Bid Formulation**

Adaptation initiates with the construction of a plan/workflow for facing unforeseen, runtime security threats in a localised manner. More specifically, a bid specification is formulated by each user agent to inquire the discovery, selection and allocation of appropriate services and resources from different SPs in a Cloud. The bid is constructed with the assistance of the Random Forest classifier which identifies a suitable adaptation strategy for facing a runtime threat based on previously encountered threats and their mitigation strategies which are recorded in the knowledge-base. To ensure that the adaptation strategy will abide to the constraints defined by a user agent, if any, the machine

learner, facilitated by user agents, uses a constraint evaluator that holds and warrants that the adaptation constraints are respected throughout adaptation. The constructed bid is then forwarded to the market coordinator for auctioning for the identification of appropriate services and resources for security.

#### Bid Auctioning

The adaptation continues at a global/system-level, where the bids received by the market coordinator are entered in multiple auctions with available ask(s) to discover appropriate services and resources that can best satisfy the security goals, sub-goals and constraints of each asset. Depending on the method used for auctioning (e.g. Posted-Offer auction, English auction, etc.) it is possible to witness different results, in terms of service/resource availability, computational overhead and time delays. Based on the application environment, different ad-hoc market mechanisms must be deployed to witness the required results.

#### 3.2.2.3 Execution Phase

The execution phase is a system-specific procedure grounded on the composition, goals and idiosyncrasies of a self-adaptive security system [51], [48], [52]. Therefore, there is no work known to us that is explicitly concerned with the execution phase of the MAPE architecture. In the absence of related work, we follow a simplistic approach to model the execution phase. In the event of a match between a bid and an ask, the user agent pays the SP(s) for their services/resources via the coordinator. Following payment, the SP(s) allocate their services and resources to the winner user agent. Upon the deployment of the received services and resources, enforced by the adaptation executor, the training dataset (i.e. knowledge-base) of a user agent is accordingly updated to maintain information concerning the latest threats and their defensive adaptation strategies.

## 3.3 Variant Design And Implementation

This section examines the applicability of the proposed framework by instantiating it from the aforementioned high-level architecture and realising a variant of the system based on Cloud storage and Voice-Over-IP (VOIP) services in a university application environment. It is possible for us to develop the proposed framework in different ways. However, we have chosen the aforementioned services and application environment, due to the wide usage of storage and VOIP services by a large number of users in conjunction with our familiarity with university environments. Universities often maintain immense amounts of sensitive data/assets, such as classified research, student data, etc. which are used by various users. Securing assets in these environments is a challenging undertaking due to the large number of users and assets that need to be secured, alongside the often limited resources available for supporting security. The remaining of this section provides an in-depth analysis of our variant design and implementation.

### 3.3.1 Adaptation Triggering

During the monitor phase, sensors are employed at the user agent side to detect and record changes in the security goals and sub-goals of their assets (e.g., anonymity, integrity, etc.). Once a runtime threat is detected, the recorded data is fed to the adaptation evaluator, during the analysis phase, to determine if adaptation is necessary.

Two questions are raised from the monitor and analysis phase: Which system attributes to record? and When should adaptation be stimulated? According to Salehie et al. [30] the adaptation methodology varies for each self-adaptive system due to six causes: "i) the different attributes of adaptable software and the dependency between them, ii) the temporal aspects of adaptation, iii) the system attributes that can be altered through adaptation and what needs to be altered in each situation? iv) the goals of a system, v) the level of automation and vi) what are the most appropriate actions to take for each given condition?" Since those parameters are unknown to us and determining the optimal

attributes for informing adaptation is outside the scope of this thesis, we do not provide a fine-grained analysis of which system attributes to record and the threshold tolerance for adaptation. Instead, based on our experience with machine learning, we base our adaptation methodology on the following attributes:

- Location: This dimension is concerned with the locality of a university employee. The values used to describe this attribute are: secured location, unknown location, and hostile location.

- Position: Signifies the working position of a university employee. Different work positions necessitate different levels of security. The values used for describing this dimension are: researchers, lecturers and administrators. We assume that researchers and administrators maintain large amounts of sensitive data, whereas lectures maintain limited or no sensitive data.

- Device: Describes the type of the device used by a university employee for performing a task. The values used for describing this dimension are: university device (secured device within the premises of a university) and personal device (unsecured device).

When the "state" of one or more of the aforementioned attributes (*(p isChanged) OR (l isChanged) OR (s isChanged) OR (d isChanged)*) changes user agents trigger adaptation (*TriggerAdaptation(T)*) to proactively secure an asset. To simulate the events causing the runtime changes on the above attributes, we have assigned a timer to each agent which alters their state/values at random time intervals. An algorithmic description of the adaptation triggering process is illustrated in Algorithm 1.

### 3.3.2 Forming Bids

Once the adaptation evaluator triggers adaptation, the user agent employs the classification algorithm to refine and map the security sub-goals of an asset to suitable services and

**Algorithm 1** Adaptation triggering

**Let:** p denote the working position of an employee, l denote the location of an employee, d denote the device type and t denote a Boolean value showing whether adaptation is needed.

```
while (UserAgent isRunning) do
  if (p isChanged) OR (l isChanged) OR (d isChanged) then
    t:=true
    TriggerAdaptation(t)
  end if
end while
```

resources. These services/resources are believed to satisfy the runtime security goals and sub-goals of an asset in the most efficient way. In particular, to adapt the user agents deploy the Random Forest algorithm to analyse the historical data of users and accordingly devise a bid specification for auctioning. Random Forest refers to a supervised learning technique founded on bootstrap aggregation and the random selection of features. Given a dataset $D$ of size $N((x1, y1), .., (xN, yN))$, the algorithm constructs $B$ tree classifiers by selecting uniformly at random $Z$ bootstrap samples of size $N' <= N$ from $D$. At each node of each tree $b$, where $b \in B$ , the algorithm selects a random set of variables $m$ from the variables $p$ ($m \subseteq p$) and splits the nodes in the tree to two children nodes. Once the ensemble trees $T_{b1}^{B}$ are constructed, each tree votes for the dominant class with equal weight $\frac{1}{B}\sum_{1}^{B} Pb(c|x)$, where $c$ is the probability of each outcome in a single tree and $x$ each test point. The class with the most votes is the algorithm's derived decision / bid (*SecDecision:=AVG(SUM(Pb(c | x))*).

The formed bids are then examined by the constraint evaluator to ensure that the selected adaptation strategies do not violate any monetary or computational constraints. Each bid comprises the following attributes: i) the name of the asset that needs to be secured; ii) the security sub-goals that require support; iii) the highest price that a user is willing to pay to secure an asset; iv) the type of service required (VoIP and/or data storage services); and v) the level of security provision required. Agents can specify the level of security provision for each service via the usage of a predefined list of security

descriptors. For VoIP services the following descriptors are used: encryption in transit (encrypts data while transferred online, e.g. TLS/SSL); proxy support (usage of proxy servers for anonymity); and media encryption (provides authentication and authorization, e.g. Secure Real-time Transport Protocol), whereas storage services are described by encryption at rest (encryption of stored data); encryption in transit; different encryption keys per file; password protected files; and segmentation of files to different physical machines. Based on the security goals and sub-goals of an asset, each user agent can use different combinations of security descriptors to request different levels of security. To simplify the process of constructing bids, in our proof-of-concept system, we have restricted our security descriptors to boolean / binary values which can either request or eliminate a security function. The selected attributes are system specific and are selected for serving the explicit needs of our envisioned market architecture / proof-of-concept. Therefore, the selected binary system should be modified / altered based on the explicit requirements and goals of an application system / market. An example of an alternative set of descriptors for the encryption at rest feature is the usage of the values: AES, Triple-DES and Blowfish encryption algorithms.

To determine how the choice of an auction can affect the trade-off between security and cost of a solution, we have engineered and instantiate our system with two dominant market mechanisms, namely the English auction and a variant of the Posted-offer auction models. Consequently, we use two different methods to calculate the bidding prices when forming bids.

To approximate the behaviour of a real-life user that is trying to establish suitable bidding prices in our Posted-Offer variant model, we allow user agents to view historical data of bids of interest (*histbids:=ExtractFixBiddingPrices(historicData)*). User agents determine their bidding prices (highest price that a user is willing to bid/pay in an auction) by calculating the average value of all the recorded bid prices and then subtract or add to that value a percentage $v$ ($\frac{\sum_{i=1}^{n} BidPrices}{NumOfBids} \pm \sum_{i=1}^{n} BidPrices \times v$). $v$ is a flexible parameter which can change according to the needs of each market. For the purposes

of our experiments, $v$ is generated with normal distribution between 1% and 5%. The methodology used for calculating bidding prices aims to simulate real-life economic markets with small price fluctuations, where the supply and demand can reach economic equilibrium. Alternative values can be selected for $v$, which will outcome in the generation of different bidding price ranges. The proposed bidding method has well considered the occurrence of bidding price anomalies that are often witnessed in traditional markets in terms of excessively high prices or low prices (compared to the average market bidding price). The submission of high bidding prices, from a number of bidders, in our market can only have a marginal impact as it can only ensure that these bidders will acquire the required services and resources, while slightly increasing the average bidding price of bidders in the long-run (only applicable if the majority of bidders in the market submit excessively high prices which is not often witnessed). On the contrary, if bidders submit low prices, they will not be able to acquire service, thus exposing the security of their assets to potential runtime threats. However, this is similar to what occurs in contemporary traditional markets where bidders that are not able to compete with rival bidders for goods do not receive them. The selected pricing methodology must be perceived as an attempt for enabling the market to quickly converge to equilibrium, as well as a mean for quantifying the significance of assets. The proposed pricing / bidding methodology can be adjusted according to the goals of a market and the operating environment itself. Once the price is calculated the user agent encapsulates it along with the other required attributes in a bid.

Similarly, the English auction procedure follows almost identical steps to the Posted-Offer variant model for calculating bidding prices. In the English auction, each user agent calculates its highest bidding price by considering the closing prices of completed auctions (*histClosebids:=ExtractClosingAuctionPrices(historicData)* ), in contrast to the fixed bidding prices used in the Posted-Offer variant model ($\frac{\sum_{i=1}^{n} ClosingPrices}{NumfOfAuctions} \pm \sum_{i=1}^{n} ClosingPrices \times v$). An algorithmic description of the formulation of bids is presented in Algorithm 2.

**Algorithm 2** Bid formulation

**Let:** t denote the Boolean output from Algorithm 1, b denote a tree, B denote the ensemble trees, Z denote the samples used to create a tree, N denote the total number of data samples, m denote the variables selected from a dataset, p denote the total number of variables in a dataset, c denote the probability of each outcome in a tree, x denote each test point/node in a tree and v denote a flexible percentile increment value.

**if** t isTrue **then**
  **for all** $b \in B$ **do**
    Select Z samples of N size from Dataset D
    **repeat**
      create Tree $b$
      randomly select m variables from p
      Pick best split-point among m // $p$ ($m \subseteq p$)
      Split node to two children nodes
    **until** node $N_{min}$ is reached in Tree $b$
    EnsembleTrees.add($b$)
  **end for**
  SecDecision:=AVG(SUM(P$b$(c | x))
  historicData:=retrieveHistoryOfBids()
  **if** AuctionType isPostedOffer **then**
    BidPrices:=ExtractFixBiddingPrices(historicData)
    calcBidPrice:=AVG(BidPrices)$\pm$AVG(BidPrices)$\times v$
    fB:=(calcBidPrice, secDecision)
  **else if** AuctionType is EnglishAuction **then**
    ClosingPrices:=ExtractClosingAuctionPrices(historicData)
    calcBidPrice:=AVG(ClosingPrices)$\pm$AVG(ClosingPrices)$\times v$
    fB:=(calcBidPrice, secDecision)
  **end if**
**end if**

### 3.3.3 Forming Asks

SP's on their side form their offers/asks which they forward to the market coordinator for auctioning. Each SP is assigned a timer (assignRandomTimerToSP()) which at random time intervals triggers the process of constructing and submitting asks. SPs use two methods to calculate their selling prices, namely the Posted-Offer and English auction procedures. In both procedures, the SPs determine the selling price of their services and resources based on the historical data of submitted asks. SPs estimate their selling prices by calculating the average value of previously submitted ask

prices and then subtract or add a percentage $s$ on that value, depending on the profit margin that a SP wants to make. $s$ is a flexible parameter which can change according to the needs of the simulation user. For our experiments, $s$ was generated with normal distribution between the values 1% and 10%. For the Posted-Offer procedure the values used for calculating the ask prices are the fixed prices paid by users to SPs ($\frac{\sum_{i=1}^{n} fixedPrices}{NumOfFixedPrices} \pm \sum_{i=1}^{n} fixedPrices \times s$), where for the English auction is the closing auction prices of interest ($\frac{\sum_{i=1}^{n} ClosingPrices}{NumOfClosingPrices} \pm \sum_{i=1}^{n} ClosingPrices \times s$). Similarly to the method used for calculating the bidding prices of user agents, the procedure used for calculating ask prices aims at converging to economic equilibrium. Once a selling price is calculated a SP encapsulates the price along with a specification of the services and resources that is trading in an ask (*fA:=(cost,features,resources,securityLevel)*). The algorithmic steps describing the creation of asks are presented in Algorithm 3.

**Algorithm 3** Ask formulation

**Let:** fB denote a formed Bid from Algorithm 2 and s denote a flexible parameter taking values between 1%-10%.

```
assignRandomTimerToSP()
while SP isRunning do
  if RandomTimer isTriggered then
    histData:=retrieveHistoryOfAsks()
    if AuctionType isPostedOffer then
      fixedPrice:=extractAuctionFixedPrices(histData)
      cost:=AVG(fixedPrice)±AVG(fixedPrice)×s
      fA:=(cost,features,resources,securityLevel)
    else if AuctionType isEnglishAuction then
      closingPrices:=extractAuctionClosingPrices(histData)
      cost:=AVG(closingPrices)±AVG(closingPrices)×s
      fA:=(cost,features,resources,securityLevel)
    end if
    forwardToCoordinator(fA)
  end if
end while
```

### 3.3.4 Auctioning

The proposed solution uses the English auction and a variant of the Posted-Offer model to examine how different auctioning mechanisms can influence the security, performance and applicability of our solution. The choice of the English and Posted-Offer auction mechanisms aims at promoting transparency in the bidding and allocation process. We assume that we operate in a cooperative context, where there is no benefit of sealing bids. Although, we understand that sealed bid auctions (bidders simultaneously submit sealed bids to the auctioneer, so that no bidder knows how much the other auction participants have bided) are more strategy proof than open-sealed auctions the choice of our auctioning mechanisms was made for promoting fairness, transparency and collaboration which are essential when enforcing security in ultra-large and heterogeneous environments. We are aware that unsealed bid auctioning mechanisms can be manipulated for personal gain, thus this work considers potential market-specific threats (Chapter 4) and proposes defensive methodologies (Chapter 5) for mitigating them.

#### 3.3.4.1 Posted-Offer Variant Auction

The Posted-offer auction is founded on a take-it-or-leave-it basis and conducted in two stages. In the first stage, the sellers privately select a price for their services/resources for a certain trading period along with the maximum number of units they can offer at that price. Once all sellers make their offers available their prices are revealed to buyers and rival sellers. The trading period follows, where buyers are selected in a descending price order (instead of being randomly selected as performed in the traditional Posted-Offer model [16]) and given the opportunity to purchase service on a take-it-or-leave-it basis. By introducing user bidding prices in the Posted-Offer model, we enable our solution to determine whether a user can afford to pay a seller's requested price, hence automating the selection process. As well as, to use bidding prices as a heuristic for ranking/selecting users that face an imminent threat and necessitate immediate security provision for their

assets. This rises from the sentiment that users that face an imminent threat are willing to pay higher prices compared to users that are secured. The auctioning round continues until all buyers acquire service, or until all offered service instances have been purchased.

#### 3.3.4.2 English Auction

The English auction model [112] is an open ascending auction, in which the bidding price starts at a low price and then rises incrementally so that progressively higher bids are solicited until the auction is closed or no higher bids are received.

#### 3.3.4.3 Auctioning Procedure

Once a bid is received, the coordinator discovers appropriate asks that can satisfy the security goals and sub-goals of an asset and registers that user in these auction(s) to compete with rival bidders for acquiring them. Depending on the method used for calculating the bid/ask prices there is a respective auctioning procedure (i.e. Posted-Offer or English auction).

In the case where the Posted-Offer methodology was used, the coordinator discovers SPs that can support the runtime security goals and sub-goals of an asset by comparing the service specification in an ask with the bid specification. In particular, the coordinator compares the: type of service, the level of security provision (consisting of the aforementioned security descriptors for each service) and the price to determine the suitability of a service for a user agent (*askList:=findAsksWhere(bidSecFeature satisfiedBy askSecFeature & bidSecLevel satisfiedBy askSecLevel & bidPrice <= askPrice)*). Upon elimination of all unsuitable asks, the coordinator sorts the remaining asks in ascending price order to ensure optimal prices and efficient mapping between bids and asks. Following, the algorithm examines whether the specified computational constraints, if any, of an asset can be satisfied by an available ask. In the case where an agent's computational constraints are met a match is performed (*ReqResources > askList[i].ProvResource*) whereas if a service maintains inadequate resources then the algorithm examines the next available ask (*agen-*

*tResources* < *askList[i].reqResource*). The objective is to migrate user agents to another Cloud infrastructure that maintains sufficient resources for sustaining the deployment of the required service(s).

Alternatively, in the occasion where the English auction is used, the coordinator discovers all the ongoing auctions that satisfy the type of service, the security level and bidding price and sets a bid on behalf of the user agent. The bidding price reflects the current highest price in an auction plus a bid increment value $p$ (Table 3.1). The bid increment price is the minimum amount by which a user agent's bid can be raised to become the highest bidder. The increment value can be determined based on the highest bid in an auction. The values used by our framework are similar to eBay's proxy auction increment values [113]. These values are case specific and they can be altered by user agents according to their runtime needs and the market prices.

In the occasion where a rival user agent tries to outbid the winning user agent, the out-bidded agent automatically increases its biding price to remain the highest bidder, whilst ensuring that the highest price specified in its bid is not violated (*agent isOutbided* & $z < h$). The winning auction, in which a match occurs, is the one that an agent has set a bid and upon completion of the auction round has remained the highest bidder (*auction isCompleted & agent isHighestBidder*). If a match occurs and the user agent has set a bid to more than one ongoing auctions that trade similar services/resources these bids are discarded (*deleteBidsInOngoningAuctions()*). Submitting multiple bids to more than one similar auction is permitted to increase the likelihood of a match to occur. As

Table 3.1: Bid Increment Values.

| **Current Price** | **Bid Increment** |
|---|---|
| \$0.01 - \$0.99 | \$0.06 |
| \$1.00 - \$4.99 | \$0.25 |
| \$5.00 - \$14.99 | \$0.60 |
| \$15.00 - \$59.99 | \$1.30 |
| \$60.00 - \$149.99 | \$3.00 |
| \$150.00 - \$299.99 | \$5.00 |
| \$300.00 - \$599.99 | \$14.00 |
| \$600.00 - \$1,499.99 | \$25.00 |

the proposed system is concerned with the time critical task of runtime security it needs to efficiently match bids with asks in short time periods to overcome runtime threats. Failing to do so can result in the compromise of the security of assets. An algorithmic description of the auctioning procedure is presented in Algorithm 4.

**Algorithm 4** Auctioning Procedure

**Let:** fBs denote the formed Bids from Algorithm 2, fAs denote the formed Asks from Algorithm 3, p denote a bid increment value, h denote the current highest price in an auction and z denote the highest price that a user can pay in an auction.

```
if fBs areReceived & auctionType isPostedOffer then
  askList:=findAsksWhere(bidSecFeature satisfiedBy askSecFeature & bidSecLevel
  satisfiedBy askSecLevel & bidPrice <= askPrice)
  Sort.ascendingPriceOrder(askList)
  for i=1st to N_th ask in askList do
    if ReqResources > askList[i].ProvResource then
      notifyAgentForMatch(askList[i])
      BreakLoop {match found, exit}
    else if agentResources < askList[i].reqResource then
      askList[i++] {check next ask in list}
    end if
  end for
else if fBs areReceived & AuctionType isEnglishAuction then
  askList:=findAuctionsWhere(bidSecFeature satisfiedBy askSecFeature & bidSe-
  cLevel satisfiedBy askSecLevel & bidPrice <= auctionPrice)
  for j=1st to Nth ask in askList do
    if ReqResources > askList[j].ProvResource then
      askList[j].setBid(h+p)
    end if
  end for
  while auctions areNotCompleted do
    if agent isOutbided & z < h then
      askList[j].reBid(h+p)
    end if
  end while
  if auction isCompleted & agent isHighestBidder then
    notifyAgentForMatch(askList[j])
    deleteBidsInOngoningAuctions()
  end if
end if
```

### 3.3.5 Allocation of Services and Resources

We have followed a simplistic methodology to deal with the runtime allocation of services and resources, as is outside the scope of our research. Our allocation mechanism is guided by a goal oriented modelling approach which differentiates between different types of services and associates them with different primitives (security features) and underlying computational resources. As we operate in a simulated environment it is not possible to provide explicit resource requirements for each service, therefore we instead associate each service with a range of resources.

Once a match occurs the execution phase is initiated, during which the coordinator notifies the winner SP and user agent to commence the trade. The agent is requested to forward the payment for the won service(s) and resources to the SP. Due to the security-driven nature of the proposed system, secure and transparent currency / token mechanisms are needed for performing the payment. Apart from the use of traditional fiat e-systems, cryptocurrencies can be a good solution for enhancing the security of bidders in the market due to their pseudo-anonymous nature, which will allow them to purchase services and resources without exposing their personal information / identity. Although, this is an important part of an economics-driven system, the development of a payment mechanism is also outside the scope of this work.

The transaction is recorded by the coordinator to ensure that no party will lie concerning the validity of the payment and allocation. In the case where the auctioning was performed based on the English auction, the agent needs to pay the price of the second highest bid plus a defined bid increment (Table 3.1), where if the Posted-Offer auction was used the fixed price set by a SP is paid. Upon reception of the payment, the SP constructs a VM that encapsulates the requested service instance(s) and resources and allocates it to the winner user agent. The objective is to migrate user agents to another VM that maintains sufficient resources for sustaining the deployment of the required service(s). Following allocation, the adaptation executor of each user agent deploys the required services and underlying resources to secure an asset. The coordinator is paid

for its auctioning services by adding a small commission fee for every successful match which is equally split between the winner user agent and SP. The algorithmic steps for the allocation procedure are presented in Algorithm 5.

**Algorithm 5** Allocation of services and resources

**Let:** fM denote the match found between a bid and an ask from Algorithm 4, p denote the auction price and m denote a market fee.

```
if fM isTrue then
  notifyWinners(SP, Agent)
  agent.Pay(p + m/2)
  if payment isPerformed then
    VM:=CreateVM(Services, Resources)
    SP.Allocate(VM)to(WinnerUserAgent)
  end if
end if
```

## 3.4 Test and Evaluation

To examine the applicability of our method, as well as to provide answers to the research questions presented in Chapter 1, we have developed a Java prototype simulation system that implements our framework based on the aforementioned variant design [1].

We have identified simulation as the dominant methodology for evaluating our framework as the development of real test-beds limit the experiment to the scale of the test-bed and make the reproduction of results a difficult undertaking. Additionally, the creation of real test-beds introduces low-level tasks which are time and money consuming. Furthermore, real life markets do not allow the manipulation of key attributes from developers, which restricts the experiment and its outcomes. On the contrary, the use of simulation tools, allow the evaluation of hypothesis in an environment where one can reproduce experiments. It allows users to test their services/resources in a repeatable and controllable environment free of cost. At the provider side, simulation environments allow the evaluation of different kinds of resource leasing scenarios under varying loads and pricing

[1] Simulation tool can be found at: github.com/GiannisT/UniMarketSimulation

distributions. Lastly, simulation tools enable the homogeneous quantification of results as they are generic, architecture imperative solutions, where real life markets have their own composition and deployment requirements [25].

The engineered solution manages the runtime security goals and sub-goals of the assets of multiple university employees via the selection and allocation of services and underlying resources. For illustration purposes, we restrict the available services in the market to VoIP and Cloud-based storage services. We assume that similar services can facilitate different security features and come at different price. VoIP services vary in terms of encryption in transit, media encryption and anonymity mechanisms, where storage services vary in terms of encryption at rest, encryption in transit, encryption keys per file, authentication schemes and cryptographic data splitting techniques (encrypt and split a file to different physical machines). To support these services our system is configured to support the provision of CPU, memory and disk space resources. Despite the small number of different types of services and resources supported by our simulation, it can be extended to support a wider variety of services/resources based on the needs of the users and the market itself. All the experiments were conducted with the assistance of a notebook running on Intel Core i5-3210M CPU @ 2.50 GHz, 8 GB RAM and JDK version 1.8.0_91.

The engineered simulation allowed us to answer the following research questions from Chapter 1:

1. Can asset-centric security be more cost-effective and efficient for the satisfaction of the runtime security goals of multiple assets compared to aggregated security?

2. Can market-inspired mechanisms be more effective in the allocation of services and resources compared to conventional non-market mechanisms in the presence of scarce resources in security constrained environments?

3. Can learning algorithms be used to arrive on more efficient bidding plans (identify and short list appropriate candidate solutions), instead of entering bidders into an

exhaustive bidding for candidate offers?

4. How can representative auction models influence the security and performance of the proposed system?

### 3.4.1 Aggregate vs Asset-Centric Security

Existing self-adaptive systems are not concerned with the runtime security of individual assets; instead, they treat security as an aggregated quality, which imposes high costs and unmet security goals. To demonstrate the above assertion and exemplify the significance of asset-centric security we have examined if the selection of a single storage service can satisfy the security goals of all of the assets/files maintained by a user agent. To conduct this experiment we have configured our simulation to operate with both asset-centric and aggregated security to draw conclusions concerning the effectiveness of each approach. In particular, the purpose of this experiment is to identify the number of assets with: satisfied security goals, excessive security provision and insufficient security provision. In the occasion where an asset acquires insufficient services and resources, it can lead to the compromise of its security, wherein the presence of excessive security the associated monetary and computational costs can be unnecessarily high.

To conduct this experiment we have used 50 files/assets. For each asset, we have constructed a bid (according to the method reported in subsection 3.3.2) describing its security requirements. All the values used for forming the bids were generated with normal distribution to simulate the diverse nature of the security requirements of different assets. We have then compared the constructed bids with the security specification/ask of a real life, well secured Cloud storage provider, i.e. Cubby [114] (supports encryption at rest and in transit, different encryption keys per file and password protected files), to determine whether aggregate security can satisfy the security goals of all 50 assets. The acquired results demonstrated that only 27 files met their exact security goals, where 15 obtained excessive security and 8 insufficient security due to the absence of crypto-

graphic data splitting techniques. On the other hand, the experiment conducted with the proposed asset-centric security framework exhibited the satisfaction of the security goals of all 50 assets. In terms of reserved disk space, our framework required 1 GB to accommodate all the assets, whereas the experiment conducted with Cubby reserved 5 GB of disk space, as part of its smallest storage plan.

To warrant the consistency of our results we have executed this experiment a number of times (i.e. 20). The acquired results reveal that the use of asset-centric approaches for security can be effective for the satisfaction of the security goals and constraints of multiple assets while minimising the usage of the underlying computational resources supporting security. This can be attributed to the deployment of ad-hoc security solutions that are tailored to the explicit security requirements and constraints of each asset. On the contrary, the use of aggregated security has proven limited for the protection of multiple, heterogeneous assets. This rises due to the enforcement of security solutions that are grounded on the premise that one-solution-fits-all, which overlooks the explicit security requirements and constraints of each individual asset.

### 3.4.2 Market vs Non-Market Allocation Mechanisms

As ultra-large systems need to continuously satisfy the changing security requirements and constraints of a large number of users and their assets with (often) limited resources, it is essential to use effective allocation mechanisms that can prioritise security requests based on their significance. By doing so, self-adaptive systems can allow users that face an imminent threat to acquire service first. This experiment demonstrates that market mechanisms can be an effective solution to the above challenge, compared to existing non-market mechanisms. This can be attributed to the fact that users in online markets can express their preferences concerning the type of resources required, the time they want to acquire them and the significance of their requests (reflected in their bidding prices), thus enabling self-adaptive security systems to manage excess demand [91].

To conduct this experiment we have simulated a Cloud storage provider (i.e. Cubby)

and 100 user agents (university employees), each owning 10 assets (total of 1000 assets). The simulation tool was configured to operate with limited resources that could only suffice for securing 700 assets. For each asset we have generated a bid specification by selecting attribute values with normal distribution and labelled them according to their significance: 500 as of "high significance", 100 as of "medium significance", 110 as of "low significance" and 290 as of "future storage". We have then executed our simulation with both a market (i.e Posted-Offer variant model) and a first-come-first-served non-market allocation mechanism and observed their effectiveness in prioritising security requests based on the number of assets served from each significance group. For the experiment conducted with the non-market mechanism, the security requests were submitted in a random order to approximate real-life conditions. The obtained results show that only 352 "high significance" requests, 80 "medium significance" requests, 88 "low significance" requests and 180 "future storage" requests were served. Indicating the wasteful allocation of resources as a large number of "high significance" requests were not served, where a big number of "future storage" requests were accommodated.

On the contrary, our experiment with the market mechanism indicates a more effective allocation of services and resources, allowing: 453 "high significance" requests, 90 requests of "medium significance", 85 "low significance" requests and 72 requests for "future storage" to acquire service. The results signify a vast increase in served "high significance" requests, as well as a massive drop in the number of accommodated "future storage" requests (Figure 3.4). This can be attributed to the absence of simplistic allocation mechanisms that are grounded on a "first-come-first-served" basis.

The small number of 47 "high significance" requests not accommodated by our Posted-Offer variant mechanism can be attributed to their low bidding prices. The bidding price of each asset was generated based on the significance group it belonged in. "High significance" bidding prices were generated with normal distribution between \$30-\$17, where "medium significance" bidding prices between \$23-\$12, "low significance" bids between \$19-\$9 and bids for "future storage" between \$15-\$3. The rationale behind

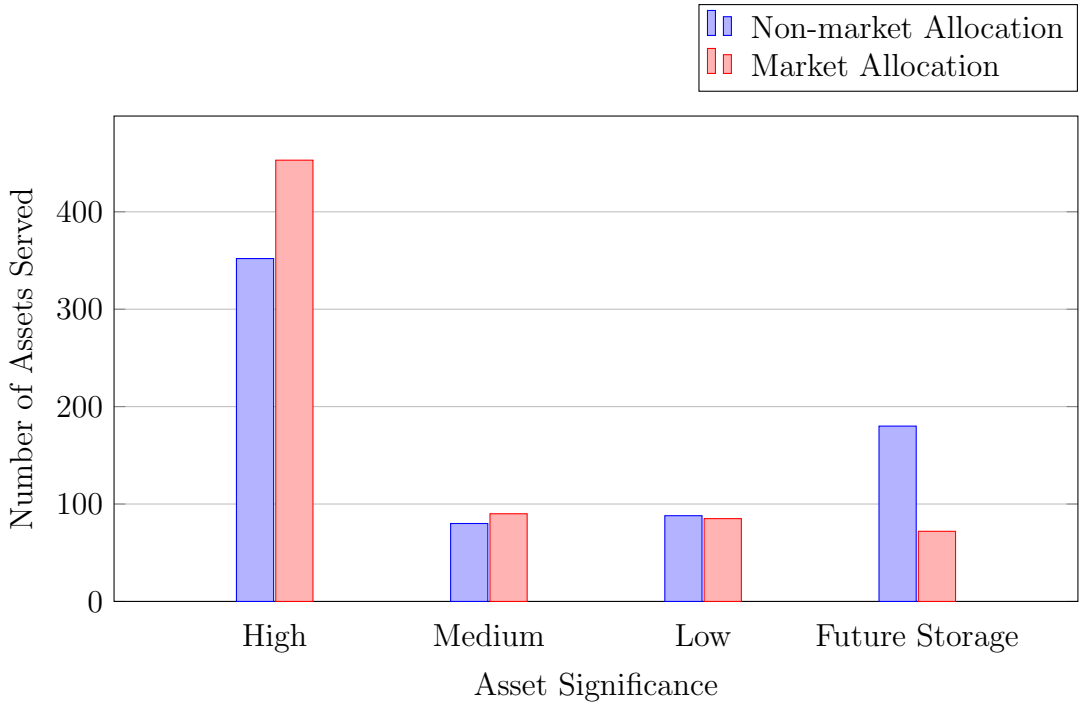


Figure 3.4: Number of assets satisfied from each "significance group" with market and non-market mechanisms.

selecting these prices is grounded on the sentiment that users that face an imminent threat are often willing to pay higher prices to secure their assets compared to users that are secured.

The obtained results (Figure 3.4) have attested our hypothesis that markets can be an effective mechanism for the allocation of services and resources in elastic environments with limited resources. We have demonstrated that market mechanisms can prioritise and satisfy the security requests of the assets originating from users that face an imminent threat while managing excess demand. To ensure the correctness of the acquired results we have executed this experiment a number of times (i.e. 20) which verified our initial results.

### 3.4.3 Machine Learning Algorithms for Security

To identify the most suitable machine learning algorithm for our solution and determine whether learning techniques can assist us in reaching more effective bidding plans for

security, we have examined the supervised learners of WEKA data mining tool [115] and monitored their effectiveness on multiple scenarios involving Cloud-based storage and VOIP services in a university application environment. In particular, this experiment allowed us to determine to what extent machine learning algorithms can automate security and whether learning can be used for optimising the bidding process to arrive in more efficient bidding plans.

To evaluate these algorithms we have generated a synthetic training dataset of 5000 samples to inform our supervised learners. Each sample comprises the position, location, device and information sensitivity of a university employee's asset. The values used for describing these attributes were selected with normal distribution from the values presented in subsection 3.3.1. We then generated a test dataset of another 5000 samples describing additional scenarios to examine the effectiveness of each algorithm. The values used for constructing the attributes of each sample were generated with normal distribution. The aim of this experiment is for each learning algorithm to classify the test samples according to the level of security provision that they require, namely: secure, fairly secure and unsecure.

Our results identified BayesNet as the dominant learning algorithm, achieving the correct classification of 91.17% of our test samples. Despite the high classification rate witnessed by the BayesNet algorithm, we have selected Random Forest as the classifier for our solution (identified as the $4^{\text{th}}$ best performing algorithm). The reason for doing so is that the Random Forest algorithm is grounded on the principles of bagging or boosting which allows it to effectively handle high dimensional spaces as well as large numbers of training samples. This makes Random Forest ideal for the efficient processing of large sizes of security information arising from multiple users and their assets in ultra-large, heterogeneous environments such as the Cloud. Table 3.2 demonstrates our comparison between the selected classifiers in terms of testing and cross-validation rates. The test results demonstrate the detection accuracy of the selected algorithms when classifying the generated test samples, whereas the cross-validation provides the average accuracy (i.e.

Table 3.2: Percentage of Correctly Classified Samples Per Classifier.

| **Classifier** | **Test** | **Cross-validation** | **Average** |
|---|---|---|---|
| BayesNet | 92% | 90.34% | 91.17% |
| NaiveBayes | 91.6% | 90.3% | 90.95% |
| NaiveBayesMultinomialText | 46.8% | 45.32% | 46.06% |
| NaiveBayesUpdatable | 91.6% | 90.3% | 90.95% |
| Logistic | 90% | 90.4% | 90.2% |
| MultilayerPerceptron | 89.8% | 90.38% | 90.09% |
| SimpleLogistic | 91.4% | 90.14% | 90.77% |
| SMO | 90.6% | 90.4% | 90.5% |
| Ibk | 86.6% | 86.76% | 86.68% |
| Kstar | 14.6% | 39.56% | 27.08% |
| DecisionTable | 90.4% | 90.52% | 90.46% |
| Jrip | 90% | 89.88% | 89.94% |
| OneR | 91.6% | 90.58% | 91.09% |
| PART | 90.6% | 90.86% | 90.73% |
| ZeroR | 46.8% | 45.32% | 46.06% |
| DecisionStump | 53.8% | 51.06% | 52.43% |
| HoeffdingTree | 91.2% | 90.3% | 90.75% |
| J48 | 89.4% | 91.38% | 90.39% |
| LMT | 90.6% | 90.42% | 90.51% |
| **Random Forest** | 90.8% | 90.34% | 90.87% |
| RandomTree | 89.6% | 85.46% | 87.53% |
| REPTree | 89.4% | 90.9% | 90.15% |

precision) of each classifier based on multiple runs. The acquired results have validated the experiments conducted in the work of Caruana and Niculescu-Mizil [116], in which a comparison between a large number of classification techniques on various datasets has been performed and identified Random Forest as one of the best classification techniques with a high detection accuracy and low computational overhead.

In spite of the high classification rate witnessed by the Random Forest algorithm, its accuracy can vary according to the volume and quality of the data informing its training process. To test the accuracy of the Random Forest algorithm in the presence of limited training data, we have used a varying number of training samples and examined the percentage of the correctly classified test samples. We have observed that even in the presence of 200 training samples, the Random Forest algorithm was still able to correctly

Table 3.3: Percentage of Correctly Classified Samples by Random Forest.

| Number of Samples | Cross-validation | Test |
|---|---|---|
| 200 | 88.05% | 88.6% |
| 500 | 91.61% | 90.4% |
| 1000 | 92.10% | 90.6% |
| 2500 | 91.60% | 90.8% |
| 3750 | 90.21% | 91.0% |

classify 88.6% of our test samples (Table 3.3).

We demonstrated that the Random Forest algorithm can be an effective mechanism for identifying appropriate services and resources for security, once a small yet significant amount of data is collected for training the algorithm. Therefore, we argue that learning approaches can be used for informing auctioning mechanisms for the efficient discovery of services and resources, instead of conducting an exhaustive search for candidate solutions. In the occasion where insufficient and/or inconsistent data is used for informing adaptation, it is possible that incorrect conclusions are drawn and false security measures are deployed. Therefore, in the existence of insufficient or inconsistent data users need to manually provide input concerning the security countermeasures that they wish to deploy.

### 3.4.4 Performance

This experiment aims to examine the performance and computational overhead of our framework. It is also concerned with how different auctioning models can influence the security and performance of the proposed system. In particular, we compare two auctioning algorithms, namely the English auction and Posted-Offer variant model and speculate concerning their performance and fit for purpose. This experiment was executed multiple times (i.e. 20) to ensure that the reported results are a true representation of the behaviour of our system.

To perform this experiment we have simulated 2 SPs, generating a total of 1500 asks,

and 100 user agents, generating a total of 1000 bids. The fixed, small number of SPs and user agents used for this experiment can be attributed to the homogeneous nature of their operations. Examining the performance of our framework with a varying number of sellers and bidders is unnecessary due to the proportional increase of the computational overhead to the number of bidders and sellers in the market. In approximately 40 seconds our simulation system was able to replicate a Cloud environment, construct all system entities described in section 3.2.1, simulate runtime threats (trigger adaptation), formulate bids with the assistance of the Random Forest algorithm, perform market auctions and allocate services and resources to user agents. No network latency or noise was simulated in our approach; hence, the time recorded for the simulation is not an accurate representation of the execution time in real life, since it can vary based on the network latency, noise, data rate, collisions and traffic in a network.

In terms of computational overhead, our approach demanded between 15 MB and 108 MB of RAM and up to 40% of our CPU resources (Intel Core i5-3210M CPU @ 2.50GHz) to operate with the given configuration. To identify the additional computational strain imposed from continuously satisfying the security requirements and constraints of individual assets, we have configured our simulation to treat security as an aggregated quality with the allocation of a "one-fits-all" service to each bidder. We have then compared the acquired results with the computational overhead exhibited in our experiment with the proposed asset-centric approach. The obtained results illustrate that aggregated security can significantly lower computational overhead, 15 MB - 89 MB of RAM (Figure 3.5) and up to 25% of our CPU resources (Figure 3.6), compared to our asset-centric security framework.

Furthermore, to assert the performance and effectiveness of our framework with different auctioning algorithms we have executed our simulation 20 additional times using the Posted-Offer and English auction models and compared their performance and fit for purpose. Throughout these tests, we have observed significant differences between the two algorithms. English auction exhibited a more complex, time-consuming procedure that

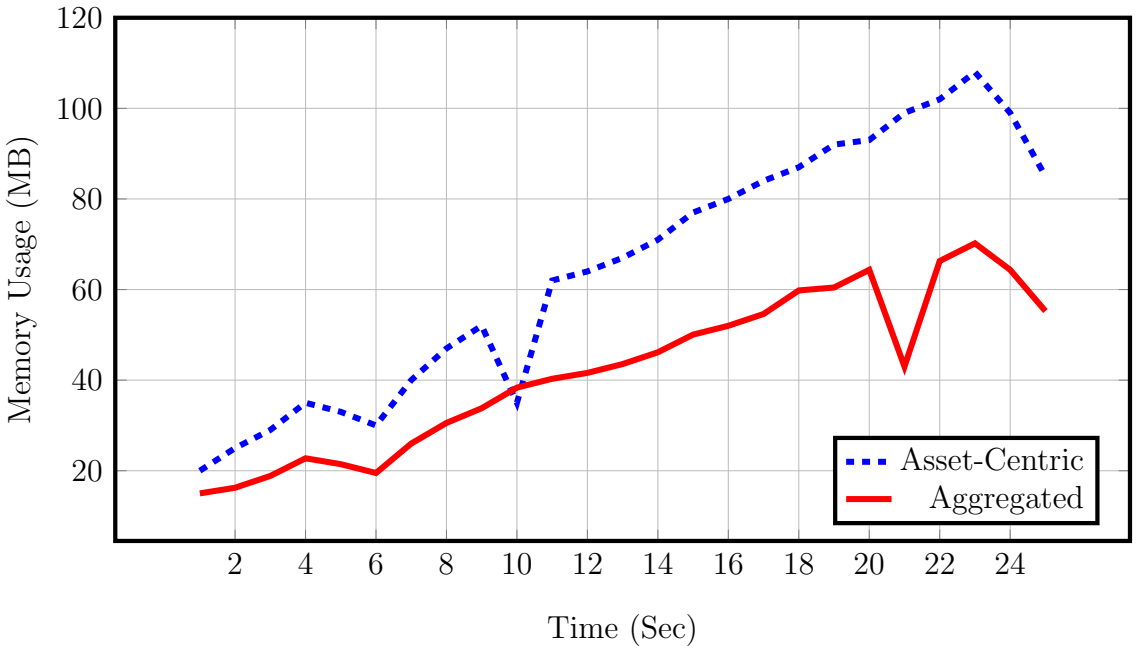


Figure 3.5: Memory usage with aggregated and asset-centric security.

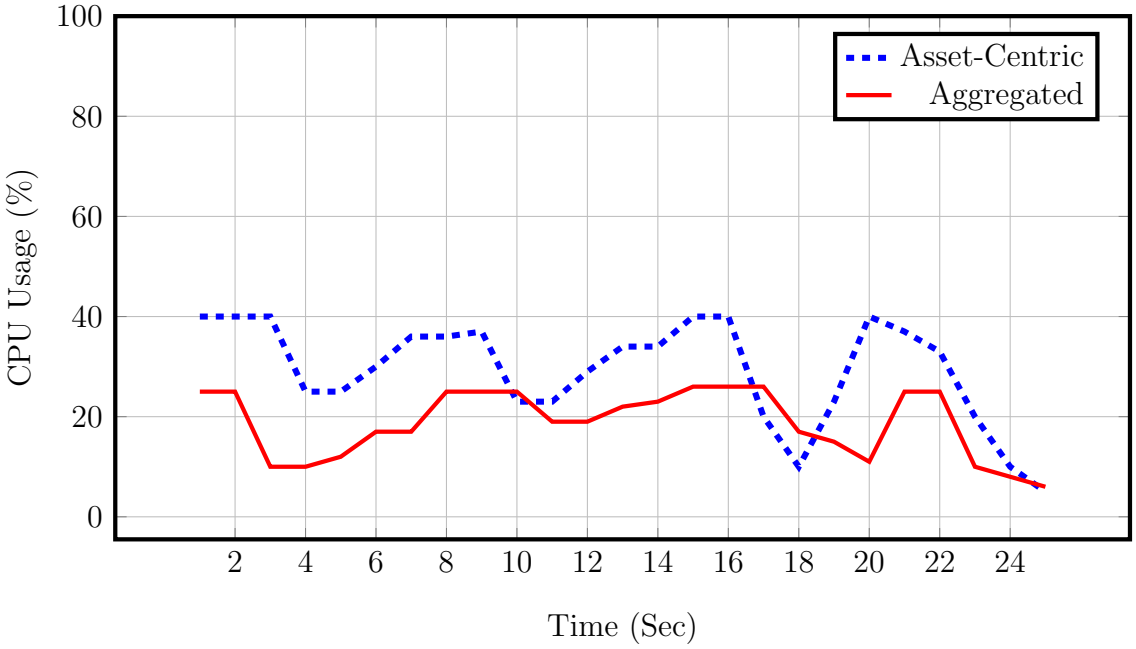


Figure 3.6: CPU usage with aggregated and asset-centric security.

necessitated an average of 789.8 milliseconds to establish an auction match in contrast to the 577.8 milliseconds required by our Posted-Offer variant algorithm. Although the time difference exhibited by the two models may seem negligible, it can play a major role in the case where a large number of assets face an imminent threat. As security is time-

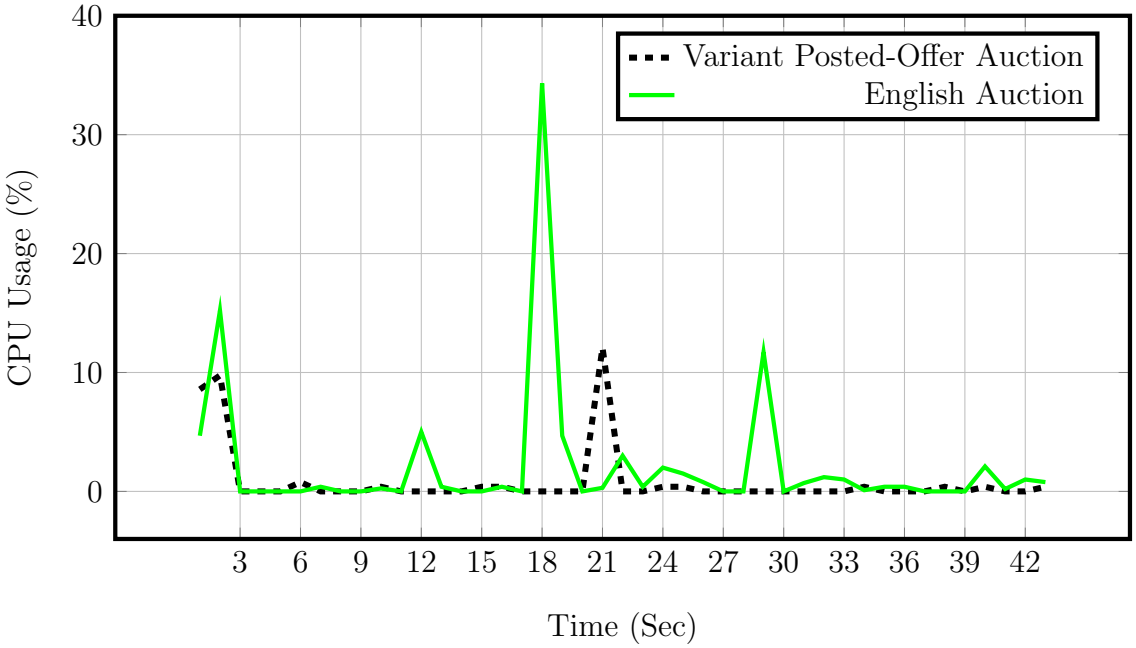


Figure 3.7: English auction and Posted-Offer algorithm CPU usage.

critical, the security of assets needs to be efficiently adapted to avoid possible time-delays that may cause their compromise. Added, the English auction algorithm displayed signs of high computational overhead, reaching at some occasions 22% more than the CPU overhead generated by our Posted-Offer variant model (Figure 3.7). On the contrary, the Posted-Offer variant model has illustrated a lower computational overhead due to the absence of complex negotiations between SPs and agents. Therefore, the Posted-Offer model can be more suitable for occasions where user agents face an imminent threat and time is a critical aspect of adapting security. On the contrary, the English auction is more suitable for occasions where user agents are not constrained by time, their security is not threatened and are willing to sacrifice time to discover cheap and/or scarce services and resources.

Despite the usage of only two auctioning algorithms, our framework can be used with a variety of auctioning algorithms. Based on the application environment and the security objectives of a system and its users, different auctioning models can be used. Each auctioning algorithm can introduce different characteristics, benefits and limitations to the problem environment. The selection of the English auction and the Posted-Offer auction

models for the purposes of this work can be attributed to their vast acknowledgement and usage in the fields of economics and computing, in conjunction with their varying nature which allow us to examine the proposed framework with two distinct auctioning approaches.

## 3.5 Summary

In this chapter, we have looked at market-inspired techniques as a candidate solution to the research questions in Chapter 1. As part of the solution, this chapter outlines an architecture grounded on agent-based modelling and market-inspired approaches infused with learning.

We have tested the applicability and effectiveness of our framework by instantiating and developing a variant of the proposed system based on a simulated university application environment, facilitating Cloud storage and VOIP services. In particular, this chapter provided answers to the succeeding research questions from Chapter 1:

1. Can asset-centric security be more cost-effective and efficient for the satisfaction of the runtime security goals of multiple assets compared to aggregated security?

2. Can market-inspired mechanisms be more effective in the allocation of services and resources compared to non-market mechanisms in the presence of scarce resources in security constrained environments?

3. Can learning algorithms be used to arrive on more efficient bidding plans (identify and short list appropriate candidate solutions), instead of entering bidders into an exhaustive bidding for candidate offers?

4. How can representative auction models influence the security and performance of the proposed system?

We have demonstrated that asset-centric security is more effective than aggregate security in terms of costs and security goal satisfaction. We have also shown that market

models are an effective and scalable solution for securing multiple assets in environments with limited resources as they enable users to compete for provision in a decentralised manner. Moreover, we have compared the performance of the English and Posted-Offer auction models and showed that the latter should be employed in occasions where time is a critical aspect of adapting security due to its simplistic nature. On the contrary, the English auction should be employed when users are secured and need to acquire cheap and/or scarce services due to the bargaining that takes place between agents and SPs. Finally, we have illustrated that learning approaches can assist users in short listing candidate solutions and acquire service in an efficient and dynamic manner without embarking on an exhaustive search for candidate solutions.

In the next chapter, Chapter 4, we examine the dependability of market-inspired methodologies for security. In particular, we investigate market-specific attacks and their effects on existing market-oriented Cloud architectures.

# CHAPTER 4

# INVESTIGATING MARKET-SPECIFIC THREATS

Market-oriented methodologies have been widely employed by software engineers for solving dynamic allocation problems [18], [19], [20] in online systems such as the Cloud. Despite the growing work in the area, the majority of the existing methodologies such as SHARP [87], Tycoon [88], Bellagio [91], Shirako [90], Aneka [117] and Gridbus [118] are neither security oriented nor they provide treatment for market specific threats. The existing work on market-oriented Cloud security has primarily focused on improving the anonymity [119], [120], [121], confidentiality [122], [123] and integrity [124] characteristics of these systems; however it has less considered market-specific threats, which can affect the operation of bidders, sellers and auction mechanisms. To ensure the secure operation of these systems, it is not sufficient to protect them against generic attacks (e.g. denial of service, etc.), but to also consider possible market-specific threats.

In this chapter, we investigate market-specific attacks in the Cloud and follow an experimental methodology to demonstrate the deficiency of existing market-oriented Clouds in facing these threats. We then examine how these attacks can manipulate the operations of bidders, sellers and auctioning mechanisms for personal gain. In particular, we focus on the analysis of four market-specific threats, namely: Shill bidding attack, Reputation attack, Monopoly attack and Denial of payment attack, and their effects on existing market-oriented Clouds. [2]

[2]Part of the work presented in this chapter has been published in [125].

## 4.1 Background Information and Related Work

This section introduces the four selected market-specific attacks. It then presents related work on the analysis and counteraction of market-specific threats in online markets and differentiates them from our work.

### 4.1.1 Shill Bidding

Shill bidding is the deliberate forgery and submission of bids in an auction to escalate the auction price and defraud legitimate bidders. By submitting shill bids, a seller can inflate the bidding price and sell goods at a higher price [126]. The effectiveness of shill bidding can vary based on the auctioning algorithm used in a market. For example, the submission of shill bids in English auctions can rapidly escalate auction prices, as each bidder can set a new highest price for services and resources. In contrast, the submission of shill bids in sealed bid auctions can only set the reserve price for the shill bid value (highest bidding price) prior to the start of an auction, which renders shill bidding ineffective.

Although online auctioning houses forbid shill bidding, there has not been any recorded attempt concerning the analysis of shill bidding in the context of the Cloud. The existing work restricts its application to simplistic e-commerce markets that trade single items and are grounded on the English auction model. Despite the significance of the existing work, market-oriented Clouds often necessitate the use of combinatorial auctions to allocate bundles of services and resources to bidders, which is not addressed by the existing literature. Furthermore, the majority of the existing work assumes that the operating environment is semi-trusted with bidders that operate in a symmetric fashion, which does not hold for shared, dynamic and elastic environments, such as the Cloud. Some notable work on the analysis and mitigation of shill bidding attacks are the following:

The work of Bhargava et al. [127] proposes a shill bidding counteracting algorithm for risk neutral online English auctions. The proposed algorithm uses an equilibrium strategy that maximises a bidder's utility, holding the bidding strategies of all other bidders fixed.

The authors assume that agents bid according to a symmetric strategy that maximises their utility as it results into Bayes Nash equilibrium.

Kauffman et al. [128] analyse bidding data from eBay coin auctions to gain insight and counter reverse price shill bidding. To identify shill bidding the authors examine the following attributes: the ratio of the number of auctions with questionable bids compared to the total number of auctions held by a seller; the experience level of a seller; a seller's reputation; the starting bid in an auction; duration of an auction; and the coin value.

The work of Threvathan et al. [129] presents an algorithm that detects shill bidding in online English auctions. The algorithm observes bidding patterns over a series of auctions, providing each bidder with a score indicating the likelihood of their potential involvement in shill behaviour. The assigned score is determined by analysing the following bidding characteristics: seller received bids; bidding frequency; number of won auctions; time of bid submission; and bidding price.

Wang et al. [126] examine the effects of shill bidding attacks on English auction markets and how auctioneers can deterring fraud. More specifically, this work promotes a Shill-deterrent Fee Schedule (SDFS) algorithm which allows auctioneers to charge sellers an intermediation fee for discouraging them from submitting shill bids. The intermediation fee is calculated based on: i) "the listing fee, which is a function of the seller's reverse", ii) "the commission fee, which is a function of the commission rate and the difference between the final sale price and the reserve price" and iii) "the commission rate, which is mathematically determined to ensure the non-profitability of shill bidding" [126]. The authors argue that the selection of appropriate commission rates can eliminate additional gains for sellers.

### 4.1.2 Reputation Attack

Reputation attack is the defamation of sellers by rival sellers or buyers. Attackers intentionally fabricate false bad feedback and submit them to sellers to harm their status, profit and clientle. False bad feedback damages the reputation of sellers and the infor-

mation available to bidders, so meaning that markets can no longer function properly. Reputation attacks can be equally effective in all types of auctions, given that a market uses reputation status for sellers.

Despite the frequent occurrence of reputation attacks in online systems [130], [131] very little has been done for their analysis, including the effects of reputation attacks in market-oriented Clouds. Some noteworthy attempts on the analysis and mitigation of reputation attacks have been conducted by the following work.

FIRE reputation system [132] computes a trust metric for each user in a market by classifying trust information into direct experience, witness information, role-based rules and third party referrals. It then filters out and penalizes inaccurate opinions. FIRE uses an inaccuracy tolerance threshold to specify the maximal permitted differences between the actual performance and witness rating. To operate, FIRE requires data from multiple sources, which in the Cloud is a complex undertaking due to the absence of third party referrals and user refusal to provide data.

TRAVOS [133] is a reputation system that uses Bayesian probability to compute the trust of an agent by analysing the past experience between two agents. To remove unfair opinions TRAVOS estimates the accuracy of reputation advice based on the number of valid and invalid advice submitted by an agent in the past. TRAVOS assumes that sellers act consistently which may not be the case in the Cloud.

Beta Reputation System (BRS) [134] estimates the reputation of sellers by using the beta probability density function. It combines the ratings of a seller being provided with multiple advisors by accumulating the number of good and bad ratings. To handle unfair opinions, BRS filters out ratings that are not in the majority. BRS can operate only when the majority of the ratings are fair, which cannot be guaranteed in the Cloud due to its heterogeneous and dynamic nature.

### 4.1.3 Monopoly Attack

In economics, a monopolistic market describes a market structure with a lone seller that trades a unique product [135]. In such a market, the seller faces no competition as he is the only seller. Monopolies can distort investment incentives and damage market profit. In our context, monopoly attack is perceived as any attempt, originating from a Cloud seller, that aims to buy up the vast majority of the resources in a market for escalating their prices.

Despite the significance of monopolies and their acknowledgement by economists, there is no known to us work that addresses monopolies in the context of the Cloud or work that considers monopoly from the perspective of a malicious, deliberate attack. The existing work is rather concerned with the theoretical analysis of specific monopolistic markets, including their effects on the incentives of bidders and sellers [136], [137]. This can be primarily attributed to the reason that traditional monopolistic markets often facilitate a single dominant seller with no major competitors. On the contrary, in the context of market-oriented Clouds it is possible for a malicious seller to join a market at any point and attempt to corner it by leasing physical infrastructure and services from rival sellers. To achieve this, the seller needs to maintain sufficient funds to purchase / lease the majority of the resources in a market. Therefore, it is significant to view monopoly as a candidate threat for market-oriented Clouds and examine its effects on these systems.

### 4.1.4 Denial of Payment Attack

Denial of payment attacks describes the process of a buyer/seller that creates fake bidding accounts to bid and win resources from sellers and then intentionally deny payment to harm their profit. Despite the significance of the denial of payment attacks, they have not received any attention from economists or computer scientists as existing traditional markets operate with the assumption that a vast majority of the buyers submit legitimate

requests for service. This assumption does not hold for contemporary market-oriented Clouds as it is possible for an attacker to create multiple fake bidding accounts and perform collaborative denial of payment attacks.

In the case where a collaborative denial of payment attack is performed by the minority of bidders in a market, its effects can prove negligible. Whereas, if a significant number of bidders collude, a market can be severely affected. Therefore, it is essential that economists and security software engineers examine neglected market-specific threats, such as the denial of payment attack to determine their possible effects on contemporary market-oriented Clouds.

## 4.2 Experimental Analysis of Market-Specific Attacks in the Cloud

This section examines our hypothesis that market-oriented Clouds are unsecured against market-specific attacks and in particular towards the four aforementioned attacks. The main contribution of this section is to provide an answer to our research question in Chapter 1: Can market-inspired mechanisms provide a dependable and secure mechanism for securing assets in the Cloud? More specifically, we follow an experimentally driven methodology to examine the following queries: i) How is the utility of bidders and sellers affected by the deployment of the selected market-specific attacks? ii) How are existing market-oriented Clouds and their auctioning algorithms affected by these attacks? and iii) Can online markets that are vulnerable to market-specific attacks be utilized as dependable optimisation mechanisms?

To examine our hypothesis we have used the CloudSim [25] simulation framework and its CloudAuction market component as our test-bed [1]. For the purpose of our experiments, we assume that we operate in a regulated market-oriented Cloud where the market auctioneer is able to monitor the behaviour of vendors and buyers in the market

[1]The source code of the developed attacks can be found in: *github.com/GiannisT/MarketAttacks*

to identify malicious behaviour and protect all transacting parties.

The remaining of this section presents our experimental setup, including the usage of the CloudSim simulation framework and the CloudAuction component for performing our experiments. It then evaluates the effects of the four selected market-specific attacks on market-oriented Clouds.

### 4.2.1 Experimental Setup

#### CloudSim Framework and CloudAuction Component

CloudSim is a framework for modelling and simulating Cloud computing infrastructures and services. It enables the simulation of i) large-scale Cloud environments on a single physical computing node, ii) service brokers, iii) service provisioning, iv) allocation policies, v) network connections among the simulated system elements and vi) a federated Cloud environment that inter-networks resources from private and public domains. CloudSim facilitates a visualisation engine that aids in the creation and management of multiple, co-hosted virtualized services.

The CloudAuction component [25] is an extension of the Cloudsim framework that enables the system to handle auction-based services. CloudAuction simulates a market-oriented Cloud in which a varying number of SPs (sellers) trade their resources (RAM, bandwidth and CPU MIPS) to user agents / buyers with the assistance of a combinatorial double auction mechanism.

Combinatorial auctions [138], refer to auctions of multiple goods, as opposed to single auctions. In combinatorial double auctions there are $X$ items $x_1, ...., x_n$, $m$ bidders and $k$ sellers. Bidder $i$ has (true) reservation value $P_i$ per unit for a bundle of items $S_i \subseteq \{x_1, \cdots, x_n\}$ , and submits a bid $B_i$ that demands up to $\mathrm{D}_i$ units of the bundle $S_i$, such as $B_i\{P_i, D_i, S_i\}$. On the other hand, each seller $j$ forms and submits an ask $Aj$ that has $Cj$ as the unit cost and offers to sell up to $S_j$ units of $x_j$ at a unit price of $F_j$, such as $A_j\{S_j, x_j, F_j\}$. In each auction round, all bids and asks are simultaneously submitted

to the auctioneer for auctioning. The auctioneer sorts bids in a descending price order $B_1 \geq B_2 \geq ... \geq B_n$ and asks in an ascending price order $A_1 \geq A_2 \geq ... \geq A_n$, where a subset $h$ of them are used for auctioning. $h$ is the largest index such that $B_m \geq A_k$. The subset of the selected bids ($q \subseteq B_n$) and asks ($z \subseteq A_n$) are in the price range of $[\max(A_h, B_h+1), \min(B_h, A_h+1)]$ which results in an equilibrium price as both demand and supply is $h$. To match the selected asks and bids the auctioning mechanism first ensures that bids are matched with asks up to their maximum demand ($D_i < S_j$) and that asks are matched with bids up to their maximum supply ($S_j < D_i$). Finally, a settlement price is calculated by deriving the average of the selected bid and ask prices and then the average of the two (Equation 4.1). All sellers who asked less than the settlement price sell and all bidders who bid more than that price buy resources at the settlement price.

$$AVG(\frac{\sum_{i=1}^{q} D_i \times P_i}{q} + \frac{\sum_{j=1}^{z} S_j \times F_j}{z}) \tag{4.1}$$

For the purposes of our experiments, in this chapter, the items comprising our security bundles are restricted to the resources supported by the CloudSim tool, namely RAM, bandwidth and CPU MIPS. As the main goal of this chapter is to determine the effects of market-specific attacks on market-oriented Clouds, the types of services and resources traded in the market are irrelevant to our experiments. The security bundles are constructed by clustering together a set of the aforementioned resources with normal distribution to simulate a realistic environment where different bidders necessitate different bundles. The price of each bundle is generated with normal distribution between \$15-\$50, where the price of each resource is accordingly selected (with normal distribution), such as the aggregated price of all the resources in a bundle sum up to its total price. The selection of the specific pricing methodology only aims to serve as a metric for quantifying the economic effects of the attacks in the market. In the case that the selected prices are replaced with analogous (higher or lower) prices the outcome of the experiments will reflect similar trends.

As Cloud users often necessitate the composition and allocation of diverse security services and resources that are supplied by different sellers in a market, it is essential to further explore the use of combinatorial auctions. Combinatorial auctions can alleviate the added computational strain imposed by single item auctions for forming and submitting multiple bids to acquire different services and resources. As identified in subsection 2.1.2.1 one of the main characteristics that self-adaptive security solutions must maintain is the ability to identify and compose various bundles of security components at runtime. Combinatorial auctions can be a candidate solution to the problem, enabling the effective delivery of bundles of security services and resources. Despite that the previous chapters have focused on the use of single item auctions (to adhere to the existing economics-inspired computing literature and simplify the complexity of our proof-of-work), this chapter wants to explore the applicability of combinatorial auctions for security and how they are affected by market-specific attacks. By deploying our market-specific attacks in combinatorial auctions it is possible to determine how highly complex auctioning algorithms can be affected, compared to simplistic single item auctions which have received vast attention in the past. As mentioned in Chapter 3, the auctioning mechanism of each market should not be treated as a static component but rather as add-on which can be easily replaced based on the runtime requirements and constraints of a market.

**Why CloudSim Simulation Framework?**

The engineering of real test-beds limit the experiment to the scale of the test-bed and make the reproduction of results difficult. Furthermore, the creation of real test-beds introduces low-level tasks, such as setting up basic hardware and software, which is expensive and time-consuming. Additionally, attributes such as allocation and provisioning algorithms are beyond the control of developers in real life markets, which restricts the experiment and its outcomes. A suitable alternative is the use of simulation tools, which enable the evaluation of hypothesis in an environment where one can reproduce tests. Furthermore, the development of simulation frameworks allows users to test their ser-

vices and resources in a repeatable and controllable environment free of cost. At the provider side, simulation environments allow the evaluation of different resource leasing scenarios under varying loads and pricing distributions [25]. Furthermore, simulation tools allow for the homogeneous quantification of results as they are architecture imperative, where real life market-oriented Clouds have their own composition and deployment requirements.

The selection of the CloudSim tool as our test-bed can be attributed to its plethora of recognition and extensive use from academia and industry [139].

### 4.2.2 Experimental Design and Results

This subsection describes the use of CloudAuction for testing our hypothesis. We report on the effects of each market specific attack and solution on the operation of sellers, bidders and the combinatorial double auction mechanism. The market parameters (e.g. number of user agent, detection thresholds, etc.) used for our experiments are tailored according to the idiosyncrasies and characteristics of each attack. We may note that these parameters are flexible thresholds, which can be adjusted by the auctioneer according to the needs of the market.

#### 4.2.2.1 Shill Bidding

**Shill Bidding Attack In CloudAuction**

To test the CloudAuction market to shill bidders we have simulated 3 SPs and a varying number of user agents (between 5-220). In particular, the aim of this experiment is to observe how the increase in the number of shill bidders can affect the average accumulated profit margin of sellers in the market. The bidding prices submitted by user agents were generated with a normal distribution between \$10-\$50. The absence of similar experiments in the Cloud, along with the sole need for quantifying the cumulative profit of SPs have driven us to select the aforementioned indicative values. At the beginning

of each auction round, a number of bidders (currently set to 10%) are randomly selected and "forced" to act as shill bidders by submitting forged bids. As we operate in a market that uses the combinatorial double auction model, a large number of bidders (i.e. 10%) need to submit high bidding prices for the settlement price (Equation 4.1) to significantly increase. In the event where a small number of shill bidders are deployed in the market, a shill attack can remain undiscovered or have a marginal impact on the expected gain of malicious sellers. The price $P_i$ submitted in the bids of shillers is calculated by increasing the average bidding prices, $b$, submitted by legitimate users in the market by a percentage $v$ (10%-20%). Value $v$ represents the tendency of shill bidders that make unnecessarily large price increments to quickly drive up selling prices. However, as the submission of bids resulting in large bidding increments increase the likelihood of detection. Our shill bidding prices only aim to moderately increase selling prices, for legitimate bidders to retain their interest in auctions and for shill bidders to evade detection. To compute the bidding price for each shill bidder we perform the following calculation: $P_i = (b \times v) + b$.

**Comparative Results**

Our comparative results illustrate the variations observed in the average accumulated profit of SPs in the presence and absence of shill bidders in the CloudAuction market. Our initial tests were performed in the absence of shill bidders, witnessing a profit margin that was proportionally increased to the numbers of bidders in the market (Figure 4.1, dashed blue line).

We then tested CloudAuction in the presence of shill bidders, witnessing a higher average market price with a disproportional, continuous growing trend. The average accumulated market price in the absence of shill attackers was \$242.36, wherein their presence it was increased to \$285.91 (Figure 4.1, red line). The upsurge in the price demonstrated that even if shill bidders submit bids with moderate price increments, legitimate bidders will still not be able to acquire resources at optimal prices. To warrant the accuracy of the acquired results, we have performed multiple runs of this experiment

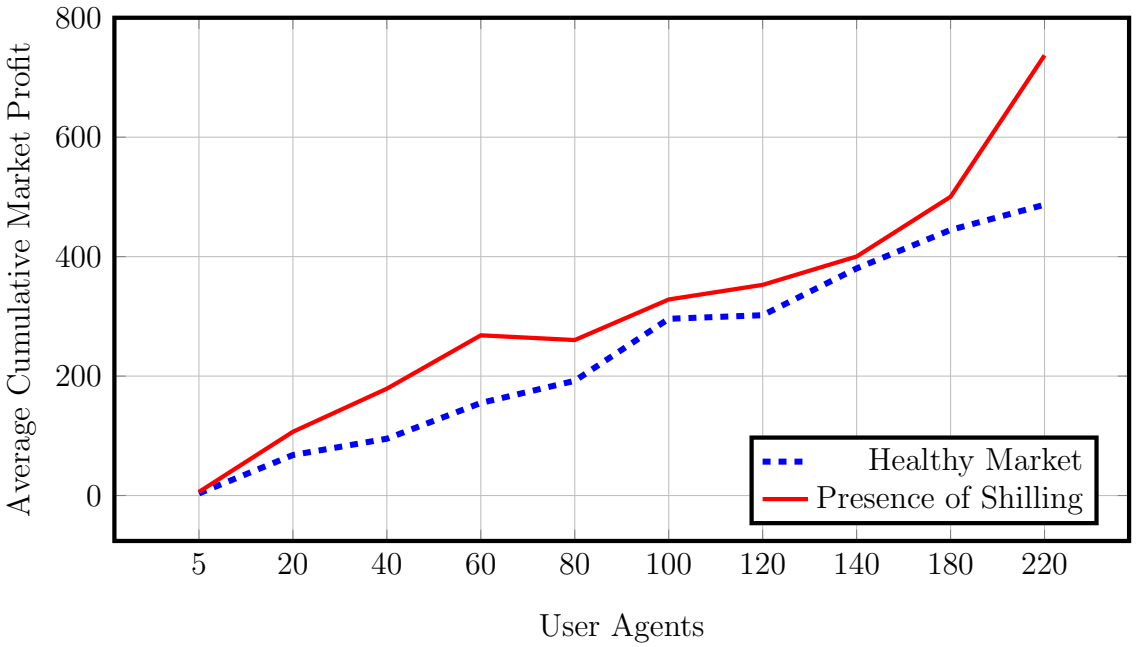


Figure 4.1: The average market price in the absence and presence of shillers.

which verified our initial results.

Although the combinatorial double auction mechanism is not highly susceptible to shill bidding attacks due to the way that its selling/settlement price is calculated, which necessitates a large number of shill bidders to collude, it was still feasible to overcome this obstacle with the creation of multiple fake bidding accounts. Demonstrating that existing market-oriented Clouds are susceptible to shill bidding attacks.

#### 4.2.2.2 Reputation Attack

**Reputation Attack In CloudAuction**

To perform reputation attacks in CloudAuction we have introduced a reputation status for each seller and allowed bidders to i) submit feedback to sellers according to their end service satisfaction and ii) to specify in their bids the reputation of the sellers they would like to obtain service from. Depending on the nature of the market and the services/resources that it trades, different feedback information can be used to enable users to express their opinions. For the purpose of this experiment we assume that all

feedback is grounded on five causes: i) seller fails to meet resource demands (e.g. allocated fewer resources, etc.), ii) seller fails to allocate resources, iii) seller fails to satisfy the QoS requirements of a bidder, iv) payment errors and v) hardware/software errors that restrict a bidder from acquiring the won resources.

To test the CloudAuction market to reputation attacks we have simulated 3 SPs and 25 user agents. The small number of sellers and buyers used in this experiment simplifies our need for the quantification of the profit of sellers. The usage of a larger/varying number of bidders and sellers is unnecessary due to the homogeneous nature of their runtime behaviour which will promote analogous results. At the end of each auction round, we randomly select one bidder that won, paid and received resources from a seller to submit false bad feedback to that seller.

```
*****BidderDatacenter: Datacenter_1
Broker ID          Debt
17                352.54
19                905.24
20                419.52
21                573.57
22                1027.92
7                 924.02
24                542.81
9                 937.69
25                032.44
11                095.56
15                808.89
**********************************
*****BidderDatacenter: Datacenter_2
Broker ID          Debt
16                864.63
17                207.01
18                649.32
23                653.54
**********************************
*****BidderDatacenter: Datacenter_3
Broker ID          Debt
16                421.3
17                420.44
18                034.15
21                731.3
6                 214
23                256.11
8                 064.73
10                216.55
11                161.08
13                287.25
14                163.49
```

Figure 4.2: The profit of sellers in the absence of false bad feedback.

#### Comparative Results

To determine how sellers are affected in the absence and presence of false bad feedback, we have initially configured CloudAuction to operate with no false bad feedback and then introduced bad feedback to a seller (i.e., SP3). Bidders were programmed to avoid sellers with bad reputation to simulate the reluctance of real world buyers to be associated with bad reputation sellers.

We have witnessed a vast variation in the profit margin of the seller with bad reputation. In the absence of false bad feedback, SP3 was able to obtain 11 bids, whereas in the presence of false bad feedback the number of bids (3 bids) and its profit dramatically decreased. The debt column in Figure 4.2 and Figure 4.3 presents the money received by the simulated SPs in the absence and presence of false bad feedback respectively. Similarly to the shill bidding attack, the bidding prices submitted by user agents in this experiment were generated with a normal distribution between \$10-\$50.

The obtained results demonstrate that the submission of false bad feedback can drive candidate buyers away from sellers and thus damage their profit. Due to the highly competitive nature of contemporary market-oriented Clouds, it is essential to identify and eliminate false bad feedback at runtime to ensure that sellers are not defaming rival sellers for personal gain.

### 4.2.2.3 Monopoly Attack

#### Monopoly Attack In CloudAuction

To conduct our monopoly experiments we have simulated 3 SPs and 25 user agents. At the beginning of each simulation round, we randomly select one seller that forwards multiple requests for leasing resources from rival sellers until it exceeds the ownership of the 50% of the overall resources in a market [141]. This aims to simulate the malicious behaviour of a seller that tries to buy up resources and corner a market.

```
*****BidderDatacenter: Datacenter_1
Broker ID          Debt
6                 115.44
7                 947.9
8                 568
10                292.84
12                698.64
14                1341.64
16                272.02
17                479.58
18                696.94
19                956.42
20                270.04
21                194.58
22                922.82
23                295.91
24                999.93
25                962.46
**********************************
*****BidderDatacenter: Datacenter_2
Broker ID          Debt
18                409.8
6                 515.18
7                 889.66
23                030.35
8                 011.56
9                 110.53
25                680.72
11                418.66
13                887.33
15                226.39
**********************************
*****BidderDatacenter: Datacenter_3
Broker ID          Debt
6                 716.92
11                310.02
13                964.28
```

Figure 4.3: The profit of sellers in the existence of false bad feedback.

### Comparative Results

Similarly to our tests for the reputation and shill bidding attacks, we have performed multiple runs for this experiment to ensure the correctness of our results. The acquired results demonstrate how the average price of each traded resource is affected in the presence and absence of a monopoly attack in CloudAuction. We have observed that monopoly can significantly increase the average price of resources. Therefore, damaging the profit of rival sellers, as well as distorting their investment interest as they are only able to acquire resources at higher prices. More specifically, we have observed that in the absence of monopoly attacks the average price for leasing RAM (per 1 MByte), storage (per 1 GByte) and bandwidth (per 1 Mbit/s) was \$4.8, \$4.8 and \$5.2 respectively, whereas when we introduced a monopoly attack the prices escalated to \$6.2, \$6.1 and \$5.5 (Figure 4.4).

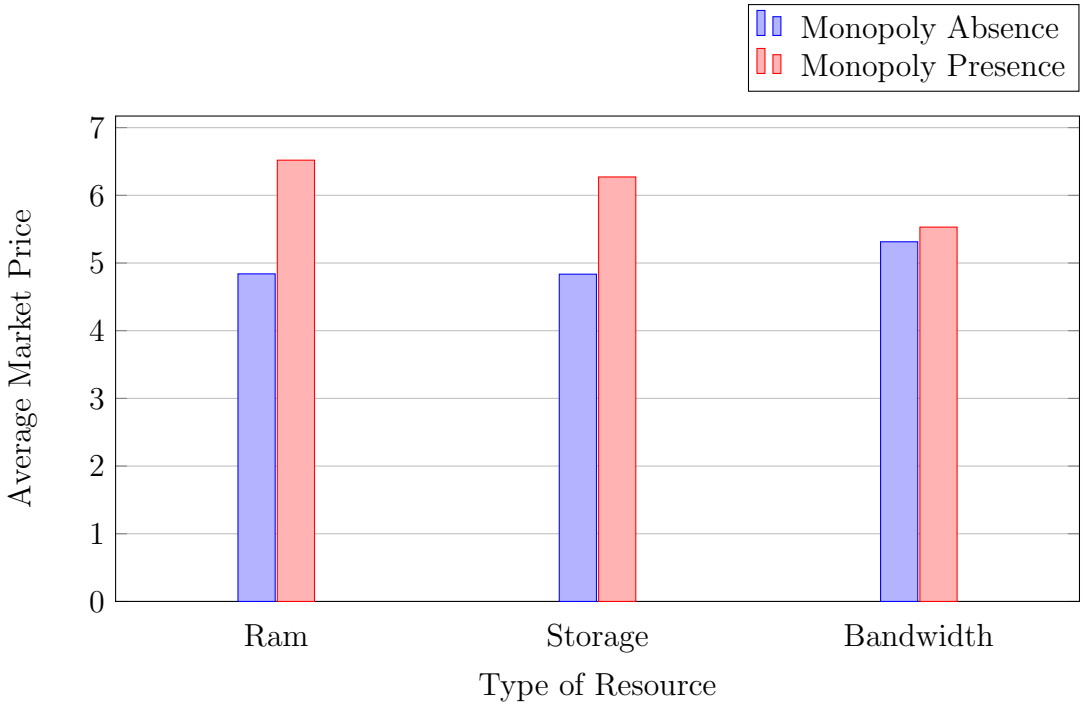


Figure 4.4: The average price of resources in the existence and absence of monopoly.

The values used for pricing the traded resources were generated with normal distribution between $4 and $6. The selected values serve as a mere metric unit for comparing the average resource prices in the absence and presence of a monopoly attack and should by no means perceived as a realistic representation of the prices of computational resources in the Cloud. The price difference exhibited between the bandwidth and the other resources in the presence and absence of a monopolistic attack can be attributed to the generation of bidding prices with normal distribution. The small bandwidth price difference in the presence and absence of monopoly does not invalidate / contradict the rest of our results as it still demonstrates that the attack has increased the price of bandwidth.

In this experiment, we have illustrated that security software engineers have not provided any treatment for the malicious acquisition of monopolistic power by service providers/sellers in market-oriented Clouds. In particular, we have shown that market-oriented Clouds are susceptible to malicious behaviour aiming to "forcibly" acquire the majority of the resources in a market for escalating their prices.

#### 4.2.2.4 Denial of Payment Attack

### Denial of Payment Attack In CloudAuction

To perform denial of payment attacks, in CloudAuction, we have simulated 3 SPs and 25 user agents. At the beginning of each simulation run, we randomly select a number (currently set to the 15%) of ongoing auctions and halt their payments. This aims to simulate the behaviour of bidders that refuse payment to sellers. The large number of bidders selected for the purpose of this attack (i.e. 15%) aims to exemplify its effects on the profit of sellers and the overall revenue of the market.

### Comparative Results

Our findings illustrate how the cumulative profit of sellers is affected in the presence and absence of denial of payment attacks. The obtained results show that the profit of sellers is severely affected by bidders that intentionally refuse to pay for won resources as their resources remain bounded, where no money is received. More specifically, in the

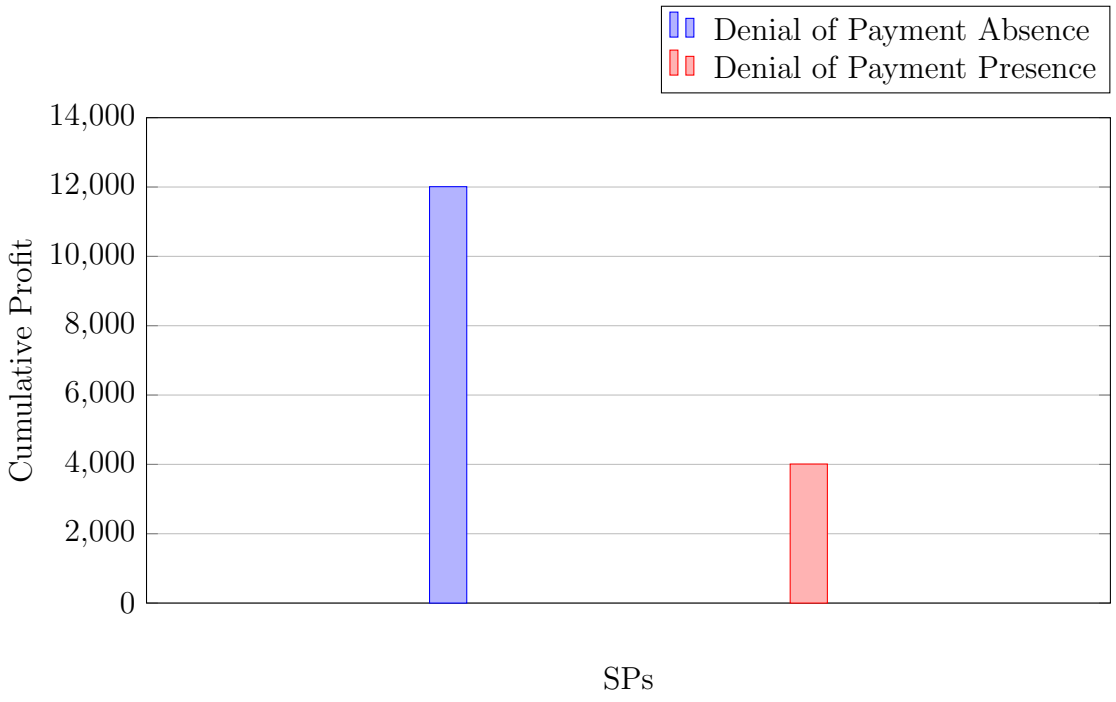


Figure 4.5: The profit of SPs in the presence and absence of denial of payment attacks.

absence of intentional denial of payments the cumulative profit of sellers reached $12010, wherein their existence the market revenue dropped to $4010 (Figure 4.5). Similarly to our previous experiments, the bidding prices submitted by user agents were generated with normal distribution between $10-$50. We have performed multiple runs of this experiment to verify the correctness of our results.

Treating denial of payment attacks in the context of the Cloud is essential as attackers can create multiple fake bidding accounts with the objective to flood a market with forged bids, thus increasing the damage caused to the profit of sellers.

## 4.3 Summary

In this chapter, we have hypothesised that market-oriented Clouds are threat-unaware and unable to deal with market-specific attacks such as shill bidding and monopoly. We have used CloudAuction, a market-oriented extension of CloudSim simulation framework, as our test-bed to verify our hypothesis.

Our results have confirmed that existing markets are vulnerable towards market-specific attacks and ascertain on how such threats can affect sellers, bidders and underlying auctioning mechanisms (i.e. combinatorial double auction) in these environments. Grounded on the findings of this chapter, the following chapter uses observations of the experiments to promote and engineer candidate, lightweight and scalable defensive mechanisms for securing market-oriented Clouds against the four examined market-specific attacks.

CHAPTER 5

# THWARTING MARKET-SPECIFIC ATTACKS IN THE CLOUD

In Chapter 4 we have shown that existing markets are vulnerable towards market-specific attacks and ascertain how such attacks can affect sellers, bidders and auctioning mechanisms. In this chapter, we use observations of the experiments on market-specific attacks to develop defensive mechanisms for securing market-oriented Clouds against these attacks. We then report on the added value of introducing our defensive mechanisms for securing market-oriented Clouds by comparing how these systems are affected in the absence and presence of the selected attacks and the proposed solutions. [2]

## 5.1 Motivation

Even though advanced strategy-proof (e.g. Vickrey–Clarke– Groves [140]) auction mechanisms exist for warranting that the established auctioning protocols are respected, we are not aware of any systematic or ad-hoc attempt in deploying them in the context of the Cloud. This may be attributed to the following: firstly, strategy-proof solutions can be complex and/or of limited scalability when applied in dynamic and fundamentally elastic environments like the Cloud. This can render them ineffective, where the perceived benefits are likely to be overtaken by the cost and overhead of their application.

[2]Part of the work presented in this chapter has been published in [125].

Secondly, these proofs have been to a big extent theoretical in nature and concerned with the fundamentals of auctions. Consequently, their assumptions and solutions may be challenged or even break when applied in scalable and unpredictable environments. The employment of strategy-proof auction mechanisms in the Cloud might negatively impact Service Level Agreement compliance in many ways, for example, i) the time required to adapt security will increase exponentially as the number of violation alerts increases, ii) given the dynamism of the system, mitigation decisions are likely stale by the time they are deployed, and iii) the auction controller may become overloaded and thus fail, making a Cloud system out-rightly unavailable.

Henceforth, market-inspired security solutions shall be fundamentally lightweight and scalable; they may need to operate with assumptions suited for the Cloud covering areas related to distribution and/or federation management. These solutions should not be concerned with the fundamental characteristics of underlying auctioning mechanisms, but instead operate as lightweight plug & play add-ons that can scale in the presence of a large number of users. Henceforth, this chapter proposes lightweight, add-on defensive mechanisms [1] for each of the four market-specific attacks presented in Chapter 4 (i.e. Shill Bidding, Reputation attack, Monopoly and Denial of payment attacks). To experimentally demonstrate the applicability and effectiveness of the proposed defensive mechanisms we have deployed them in the CloudAuction market. Signifying the added-value of market-specific defence mechanisms in market-oriented Clouds. In particular, we compare how a market can be affected in the absence and presence of the selected attacks and the proposed solutions.

## 5.2 Experimental Design and Results

This section describes the use of CloudAuction to test our candidate solutions. We report on the effects of each market-specific attack and solution on the operation of sellers,

[1] The source code of the developed attacks and solutions can be found in: *github.com/GiannisT/MarketAttacks*

bidders and the combinatorial double auction mechanism. All the experiments were performed using the experimental setup discussed in section 4.2. The market parameters (e.g. number of user agent, detection thresholds, etc.) selected for our experiments were tailored according to the idiosyncrasies and characteristics of each attack/solution. The selected parameters allowed our defensive mechanisms to attain the highest detection and lowest false positive and false negative rates possible. We may note that these parameters are flexible thresholds, which can be adjusted by the auctioneer according to the needs of the market. Unfortunately, it is impossible for us to identify and establish a set of optimal values for the detection thresholds of our defensive mechanisms. This is due to the diverse composition of markets and the environments they operate in, which render these parameters system and case specific. One possible method to overcome this problem is to examine the sensitivity of our defensive mechanisms by altering the thresholds until the required results are witnessed.

### 5.2.1 Shill Bidding

#### 5.2.1.1 Defensive Mechanism

The proposed shill bidding defensive mechanism is founded on the real time analysis of bidding records. Each bidding record archives the following attributes for each bidder: i) price difference between a buyer's bid and the next highest bid in an auction, ii) bidder's feedback (e.g. positive, negative), iii) number of lost auctions, iv) total number of auctions participated, v) number of times that a bidder overbid himself/herself while winning an auction, vi) number of bids submitted before and after the halftime of an auction, vii) the number of bids submitted to each seller and viii) the overall number of bids submitted to the market. At the end of each auction round all bidding records are analysed. The record analysis entails the comparison of the nine archived attributes with given thresholds which result to a shill value for each bidder. In the case that a comparison is true, 1 is returned and added to the "shill value", where if it is false 0 is

returned instead. The shill value illustrates the number of malicious conditions met by a bidder in a single iteration of our algorithm. The higher the score of a shill value, the higher the likelihood of a bidder being malicious.

The proposed algorithm performs the following tests to determine whether a bidder is malicious. First, it examines the feedback of a bidder. If no or negative feedback is found the algorithm increases the likelihood of a bidder being malicious (*feedback is bad OR neutral*) as shill bidders usually do not receive feedback. Following, it examines the ratio between the lost auctions of a bidder and the total number of auctions that he/she participated in. As the sole goal of shill bidders is to drive up prices and then allow legitimate bidders to win, shill bidders maintain an unusually high number of lost auctions. Therefore, we assume that if a bidder has lost more than $a = 70\%$ of the auctions he/she participated in *(LostAuct* $>$ *NumOfAuctPartic*$\times a$*)* he/she is probably malicious. We consider 70% to be a fitting value for $a$ as it is high enough to avoid the misclassification of legitimate bidders and low enough to identify shill bidders. In the case that a legitimate bidder has lost more than 70% of its auctions, he/she is still at no risk of being misclassified as this is just one of the nine dimensions examined for calculating the shill value.

Our algorithm then examines whether a bidder has overbid himself/herself while winning an auction. Shill bidders tend to overbid themselves even when they are winning an auction, as their primary goal is to constantly increase auction prices. The proposed defensive mechanism is programmed to tolerate bidders that overbid themselves once ($c = 2$) to avoid misclassifying legitimate bidders that accidentally overbid themselves (*OverbidWinSelf* $\geq c$). Next, the proposed algorithm compares the bidding frequency of a bidder during the first and second half of an auction. Shill bidders are more eager to bid during the first half of an auction in order to give legitimate bidders sufficient time to place their bids and win (*BidNumFirHalf* $>$ *BidNumSecHalf*).

Following, our defensive mechanism examines the percentage of bids submitted by a user to each seller. In particular, our algorithm examines whether a bidder has submitted

a majority of its bids (i.e. $d = 60\%$) to a single seller ($BidsToSeller > TotalBids \times d$). Shill bidders often submit all of their bids to a specific seller to increase its revenue, where legitimate users submit their bids to various vendors. Therefore, the selected value for $d$ is high enough to avoid misclassifying legitimate bidders and low enough to identify shill bidders. Even if a legitimate bidder focuses all its bids to a specific seller he/she will not be penalised as this is one of the dimensions examined to calculate the shill value. Finally, the proposed solution evaluates the price difference between the bidding price submitted by the bidder in question and the next highest bid in an auction. It then compares the identified price difference with the average price difference exhibited in similar auctions in the market ($SubmitBidPrice\text{-}OverbidPrice > AverageMarketPriceDiff$). This allows us to determine if the prices submitted by a bidder are unnecessarily high, which signifies behaviour often witnessed by shill bidders.

Once the shill value of a bidder is calculated, it is compared with a threshold (i.e. $t \geq 5$) to determine whether the bidder is malicious. The value selected for $t$ is high enough to avoid the misclassification of legitimate bidders, that exhibit signs of shill-like behaviour (meet some of the malicious conditions in our algorithm), but still effective for the identification of shill bidders in the market. If a bidder is found malicious, it is flagged as a possible shill bidder and is given a warning. If the exhibited malicious behaviour is repeated from the flagged bidder a number of times (current set to $q = 2$), the bidder needs to pay a fine $f$ to the market and its account is discarded. The algorithm has been configured to tolerate the first occurrence of shill bidding from each bidder in order to decrease the cases of false positives. The imposed fine $f$ aims to discourage shill bidding by exceeding the expected gain of a shill bidder and the seller it represents. The imposed fine is obtained by subtracting the average market price of resources in similar, shill-free auctions from the average price of resource in the presence of shill bidders and then increase the outcome by $z = 10\%$ ($f = (AVG(ShillResourcePrice) - AVG(NonShillResourcePrice)) \times z$). The algorithmic steps of the proposed defensive mechanism are presented in Algorithm 6.

**Algorithm 6** The shill bidding defensive mechanism pseudocode

**Let:** $a$ denote a flexible parameter currently set to 70 %, $c$ denote a flexible parameter currently set to 2, $d$ denote a flexible parameter currently set to 60 %, $t$ denote a flexible parameter currently set to 5, $q$ denote a flexible parameter currently set to 2, $f$ denote a fine payment, $z$ denote a flexible parameter currently set to 10 % and shillValue denote a value illustrating if a user is malicious

```
for i=1 to N_th Bidder do
  shillValue:=0
  if feedback = bad OR neutral then
    shillValue:= shillValue + 1
  end if
  if LostAuct > NumOfAuctPartic * a then
    shillValue:= shillValue + 1
  end if
  if NumOfBidsSubmittedToAuction > BidsInAuction * b then
    shillValue:= shillValue + 1
  end if
  if OverbidWinSelf ≥ c then
    shillValue:= shillValue + 1
  end if
  if BidNumFirHalf > BidNumSecHalf then
    shillValue:= shillValue + 1
  end if
  if BidsToSeller > TotalBids * d then
    shillValue:= shillValue + 1
  end if
  if SubmitBidPrice-OverbidPrice > AverageMarketPriceDiff then
    shillValue:= shillValue + 1
  end if
  if ShillValue ≥ t then
    FlagAsPossibleShillBidder()
    WarnBidder()
  end if
  if TimesFlagged ≥ q then
    f:= (AVG(ShillResourcePrice) - AVG(NonShillResourcePrice))×z
    PayFine(f)
    DeleteBidderAccount()
  end if
end for
```

#### 5.2.1.2 Results and Analysis

The proposed defensive mechanism was able to detect 83% of the shill bidders simulated in the CloudAuction market. To conduct this experiment we have used the experimental

setup used in subsection 4.2.2.1 for our shill bidding attack experiment. To demonstrate the added value of the proposed defensive mechanism we have compared the average accumulated profit of SPs (in CloudAuction) in a healthy market; in the presence of shill bidders; and in the presence of shill bidders and our solution. In particular, we have compared the results presented in Chapter 4 showing the average cumulative market price in the presence (Figure 5.1, red line) and absence (Figure 5.1, blue dotted line) of shill bidders with the average cumulative market price in the presence of shill bidders and our solution. The usage of our defensive mechanism in CloudAuction (Figure 5.1, green line), in the presence of shill bidders, has illustrated a significant drop in the average profit margin of SPs and similar price fluctuations to the experiment conducted in the absence of shill bidders. More specifically, the average accumulated market price in the absence of shill attackers was $242.36, wherein their presence it was increased to $313.91. With the deployment of our defensive mechanism, the price declined to $239.36, demonstrating a similar average cumulative price to a healthy market.

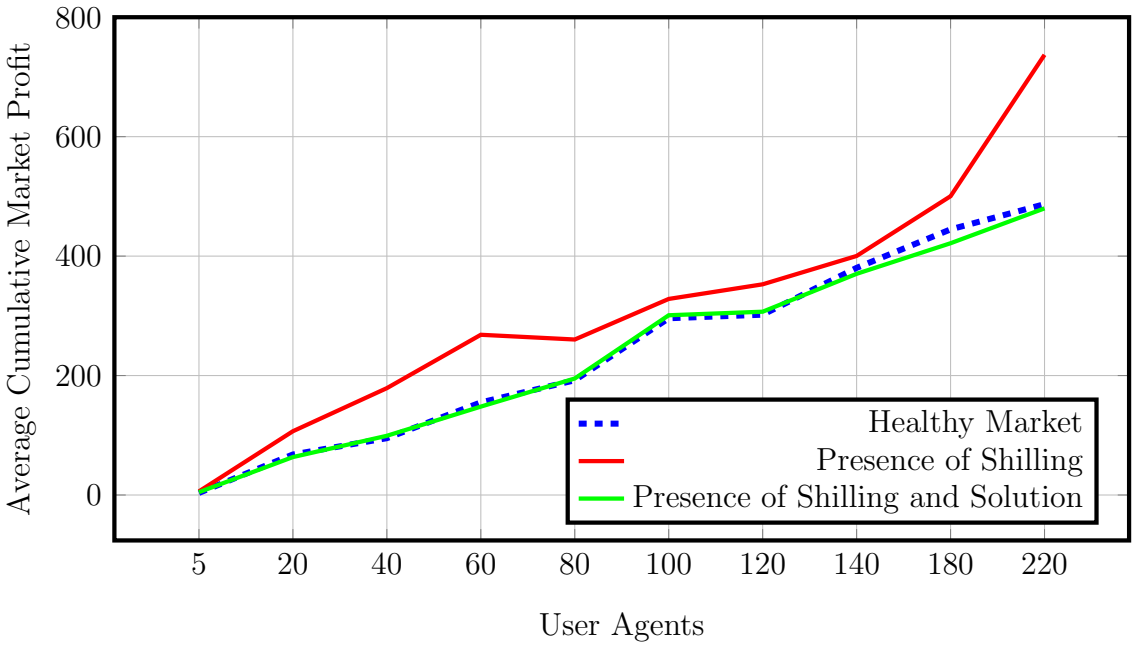


Figure 5.1: The average market price in the absence and presence of shill attackers and our solution.

Finally, to examine how sensitive is our defensive mechanism to the detection thresh-

old $t$ we have lowered $t$ from 5 to 3. The obtained results signify an increase in the detection rate of shill bidders reaching 100%. However, we have also witnessed a rise in false positives (9,5%). The acquired results were conflicting as the benefits from lowering $t$ were overtaken by the occurrence of false positives. False positives in market-oriented environments can be disastrous as they can defame legitimate bidders, damage the trustworthiness of the market and distort investment incentives. Therefore, it is better to adjust the detection threshold to a point where the defensive mechanism can detect a vast majority of the shill bidders in a market while minimizing false positive occurrences. Even if it implies that a small number of malicious bidders will still be able to deceive the defensive mechanism and occasionally shill.

### 5.2.2 Reputation Attack

#### 5.2.2.1 Defensive Mechanism

To counteract reputation attacks, the proposed defensive mechanism automatically intercepts bad feedback for analysis prior to their publication. During analysis the following aspects are examined: i) has the bid successfully received by the seller in question? ii) has the bid been served? iii) has the requested amount of bandwidth, CPUs and RAM been successfully allocated to the bidder? iv) is the agreed price paid? and v) have any hardware or software errors been reported during the utilisation of the resources by the bidder? If the analysis shows that the bidder has valid reasons for submitting bad feedback, the feedback is released and submitted to the seller, whereas if the analysis illustrates that the bidder had invalid reasons, the feedback is discarded and a bad feedback is instead submitted to the bidder. The algorithmic steps for the proposed defensive mechanism are presented in Algorithm 7. Our algorithm heavily relies on the assumption, section 3.2, that the coordinator is able to monitor the behaviour of bidders and sellers for compliance.

**Algorithm 7** Reputation defensive mechanism pseudocode

**Let:** b denote the bid associated with the submitted feedback, rBW denote the requested bandwidth, aBW denote the allocated bandwidth, rM denote the requested CPU MIPS, aM denote the allocated CPU MIPS, rR denote the requested RAM, aR denote the allocated RAM, z denote the price paid for the won services/resources and x denote the agreed bidding price for won services/resources.

**if** BadFeedback isReceived **then**
  Intercept(Feedback)
**end if**
**if** (b isReceived) & (b isServed) & (rBW $\geq$ aBW) & (rM $\geq$ aM) & (rR $\geq$ aR) & (z $\leq$ x) & (Payment is Performed) & (SoftErrors OR HardErrors not found) **then**
  Delete(Feedback)
  SubmitBadFeedback(bidder)
**else**
  ReleaseFeedback()
  SubmitBadFeedback(seller)
**end if**

#### 5.2.2.2 Results and Analysis

The proposed solution was able to detect all the cases of false feedback in CloudAuction (based on the experimental setup used in subsection 4.2.2.2) demonstrating that in some occasions it is feasible to proactively determine the validity of feedback in electronic markets without the need for human intervention and manual revision. By identifying all the cases of false bad feedback we have provided buyers with accurate information concerning the end-service quality and legitimacy of each seller, thus establishing fair competition between the sellers in the market. In subsection 4.2.2.2 we have shown that in a healthy market SP3 was able to acquire a cumulative profit of $2970.4 (Figure 4.2), wherein the presence of false bad feedback its profit declined to $1991.22 (Figure 4.3). In the presence of false bad feedback and our defensive mechanism, it was feasible to witness a cumulative profit similar to a healthy market reaching $2580. The small deviation in the cumulative profit of SP3 between a healthy market and the presence of attackers and our solution can be attributed to the generation of bidding prices with normal distribution between $10 - $50. In spite of the successful identification of all the cases of false bad feedback, the proposed solution is only applicable to online markets

that share a similar composition and trade similar services/resources to the CloudAuction market. Different markets have different requirements and necessitate different feedback information. Consequently, they also necessitate different defensive mechanisms that adhere to their idiosyncrasies.

### 5.2.3 Monopoly

#### 5.2.3.1 Defensive Mechanism

According to the work of Oswald [141] if a vendor owns more than 40% of the market shares it can be likely a case of monopoly. Based on [141], our monopoly solution flags sellers that own more than $m = 40\%$ of the market resources as suspicious. Suspicious sellers are further examined to determine whether they were intentionally buying up resources to corner the market. The analysis entails the examination of a seller's idle resources; used resources, and the number of requests made for leasing resources. In the case where a vast majority of a seller's resources are in idle state (current threshold set to $i = 70\%$) and has repeatedly requested to lease resources (current threshold set to the average number of leasing requests per hour) from rival sellers ($NumOfReq > MarketReqPerHour/NumOfSellers$), it is classified as malicious. If a seller is found malicious, its leasing capability is revoked for a period of time $t$. The algorithmic steps of the proposed monopoly defensive mechanism are presented in Algorithm 8.

#### 5.2.3.2 Results and Analysis

Based on the experimental setup used in subsection 4.2.2.3, for our monopoly attack experiment, the proposed defensive mechanism was able to successfully detect the attempt for acquiring monopolistic control. The obtained results have demonstrated that the use of our monopoly defensive mechanism can mitigate these attempts by minimising the incentives and expected gains of this attack. Thus, retaining the cost of resources in the market at low prices while protecting the financial interests of bidders. In particular,

**Algorithm 8** The monopoly defensive mechanism pseudocode

**Let:** m denote a flexible parameter set to 40%, i denote a flexible parameter set to 70% and t denote the duration of time the leasing capabilities of a seller are revoked.

**if** (sellerRAM $>$ marketRAM $*\, m$) OR (sellerStorage $>$ marketStorage $*\, m$) OR (sellerBW $>$ marketBW $*\, m$) OR (SellerMIPS $>$ marketMIPS $*\, m$) **then**
  flag(suspicious Seller)
**end if**
**if** (seller is suspicious) & ((idleSellerRAM $>$ totalSellerRAM $*\, i$) OR (idleSellerBW $>$ totalSellerBW $*\, i$) OR (idleSellerStorage $>$ totalSellersStorage $*\, i$) OR (idleSellerMIPS $>$ totalSellersMIPS $*\, i$)) & (NumOfReq $>$ MarketReqPerHour/NumOfSellers) **then**
  Flag(Malicious seller)
  PausePurchaseCapabilities(t)
**end if**

we have observed that in the presence of a monopolistic attempt and our solution, in CloudAuction, the average price for leasing RAM, storage and bandwidth resources was similar to the prices witnessed in a healthy market (Figure 4.4) reaching \$4.7, \$4.9 and \$5 respectively. The small variation in the selling prices between our experiments in a healthy market (\$4.8, \$4.8 and \$5.2) and the presence of attackers and our solution can be attributed to the dynamic generation of selling prices with normal distribution.

Regardless of the successful detection and mitigation of monopoly attacks, our algorithm is unable to resolve the situation where an attacker distributes the overall amount of owned resources to multiple fake seller accounts, to evade detection and still maintain the majority of resources in the market. However, even in this occasion, the attacker is penalised as it is both harder and more expensive to maintain multiple accounts.

### 5.2.4 Denial of Payment

#### 5.2.4.1 Defensive Mechanism

Similarly to our shill bidding defensive mechanism, this solution is grounded on the real time analysis of bidding records. Each bidder in the market maintains a bidding record that archives the following attributes: i) price difference between a submitted bid and the next highest bid in an auction, ii) a bidder's feedback, iii) number of bids submitted

from a bidder to each seller, iv) number of bids submitted from a bidder to the market, v) number of times that a bidder denied payment for won resources and vi) number of won auctions for a bidder. At the end of each auction round all bidding records are analysed. The outcome of the analysis is expressed as an unpaid-status value. The unpaid-status value is calculated by comparing the six archived bidding attributes with given thresholds. In the case that a comparison is true, 1 is returned and added to the unpaid-status value, where if it is false 0 is returned. The unpaid-status value illustrates the total number of malicious conditions met by a bidder in a single iteration of our algorithm. The higher the score of the unpaid-status, the higher the likelihood of a bidder being malicious.

The defensive algorithm initiates with examining a bidder's feedback to determine his/her credibility amongst market entities. In the case that a buyer's feedback is *feedback = bad OR neutral* his/her chance of being malicious increases. Following, the number of bids submitted to each seller (by each bidder) is examined. Bidders that intentionally refuse payment to sellers have specific targets to which they often submit a majority of their bids. Therefore, we examine whether a bidder has submitted more than a percentage (i.e. $d = 50\%$) of its bids to a single seller ($BidsToSeller > TotalBids \times d$). Following, we analyse the ratio between the total number of won auctions and the times that a bidder denied payment. We consider bidders that denied payment to more than $w = 50\%$ of their won auctions as probably malicious ($TimesDeniedPayment > WonAuctions * w$). As the sole goal of legitimate bidders is to bid and win resources (excluding few, exceptional circumstances) it is imperative that they are also willing to pay for their resources/services. Lastly, we examine the price difference between the bidding prices submitted by a bidder and the second highest price in an auction. We then compare the acquired price difference with the average price difference exhibited in similar auctions ($SubmitBidPrice\text{-}OverbidPrice > AverageMarketPriceDiff$). This allows us to determine if the prices submitted by a bidder are unnecessarily high, which is behaviour often witnessed by malicious bidders aiming to ensure winning their auctions and denying payment.

Once the unpaid-status value is calculated, it is compared with the threshold $u$ (currently set to 3), which determines if a bidder is malicious. If found malicious, he/she is warned and the maximum number of bids that is able to submit in the market is restricted to the $s = 30\%$ of the average number of bids submitted in the market per hour ($MaxBidsPerHour(AverageMarketBidsPerHour * s)$). If the bidder repeats the same malicious behaviour a number of times (i.e. $k = 3$), then he/she has to pay a fine $f$ and its account is discarded. The imposed fine is calculated by deriving the $c = 8\%$ of the total cost of the unpaid won auctions. Though we consider $f$ to be reasonably low for attackers to afford and high enough to discourage them, however, it can be adjusted according to how strict the penalty should be. The algorithmic steps for the proposed denial of payment defensive mechanism are presented in Algorithm 9.

#### 5.2.4.2 Results and Analysis

To determine the detection accuracy of our denial of payment defensive mechanism, we have simulated one hundred instances of denial of payment attacks in the CloudAuction market. The acquired results (based on the experimental setup used in subsection 4.2.2.4) signified a high detection rate, reaching the 89% of the malicious bidders in the market.

In subsection 4.2.2.4 we have compared the cumulative profit of sellers in the absence and presence of denial of payment attacks and demonstrated (Figure 4.5) that their profit can be severely damaged. In particular, in the absence of intentional denial of payments the cumulative profit of sellers reached \$12010, where in their presence the overall market revenue declined to \$4010. In this subsection, we examine the added value of our defensive mechanism by comparing the results acquired in Chapter 4 with the revenue witnessed in the presence of attackers and our solution. Our experiment illustrates that by deploying our defensive mechanism the profit of sellers escalated to \$10150, witnessing a profit margin similar to the one exhibited in a healthy market. The small price difference between the cumulative market revenue witnessed in a healthy market and the presence of attackers and our defensive mechanism can be attributed to the generation of bidding

**Algorithm 9** The denial of payment defensive mechanism pseudocode

**Let:** d denote a flexible parameter set to 50%, w denote a flexible parameter set to 50%, u denote a flexible parameter set to 3, s denote a flexible parameter set to 30%, k denote a flexible parameter set to 3, c denote a flexible parameter set to 8% and f denote the imposed fine payment.

```
for 1st to Nth Bidder do
  UnpaidStatus:=0
  if feedback = bad OR neutral then
    UnpaidStatus:= UnpaidStatus + 1
  end if
  if BidsToSeller > TotalBids * d then
    UnpaidStatus:= UnpaidStatus + 1
  end if
  if TimesDeniedPayment > WonAuctions * w then
    UnpaidStatus:= UnpaidStatus + 1
  end if
  if SubmitBidPrice-OverbidPrice > AverageMarketPriceDiff then
    UnpaidStatus:= UnpaidStatus + 1
  end if
  if UnpaidStatus ≥ u then
    FlagAsMalicious()
    WarnBidder()
    MaxBidsPerHour(AverageMarketBidsPerHour * s)
  end if
  if TimesFlagged ≥ k then
    f:= Sum(UnpaidResources) * c
    PayFine(f)
    DeleteBiddingAccount()
  end if
end for
```

and selling prices with normal distribution (Figure 5.2).

Regardless of the effectiveness of our solution, attackers can evade detection by discarding their fake bidding account after each auction and then create new ones for succeeding auctions of interest. To mitigate the certain shortcoming, we impose a registration fee (currently set to $80) on newly registered bidders to discourage the creation of multiple fake bidding accounts. The registration fee is reserved by the auctioneer and is used to pay the first won resources of the bidder.

Finally, we have attempted to improve the detection rate of our mechanism by lowering

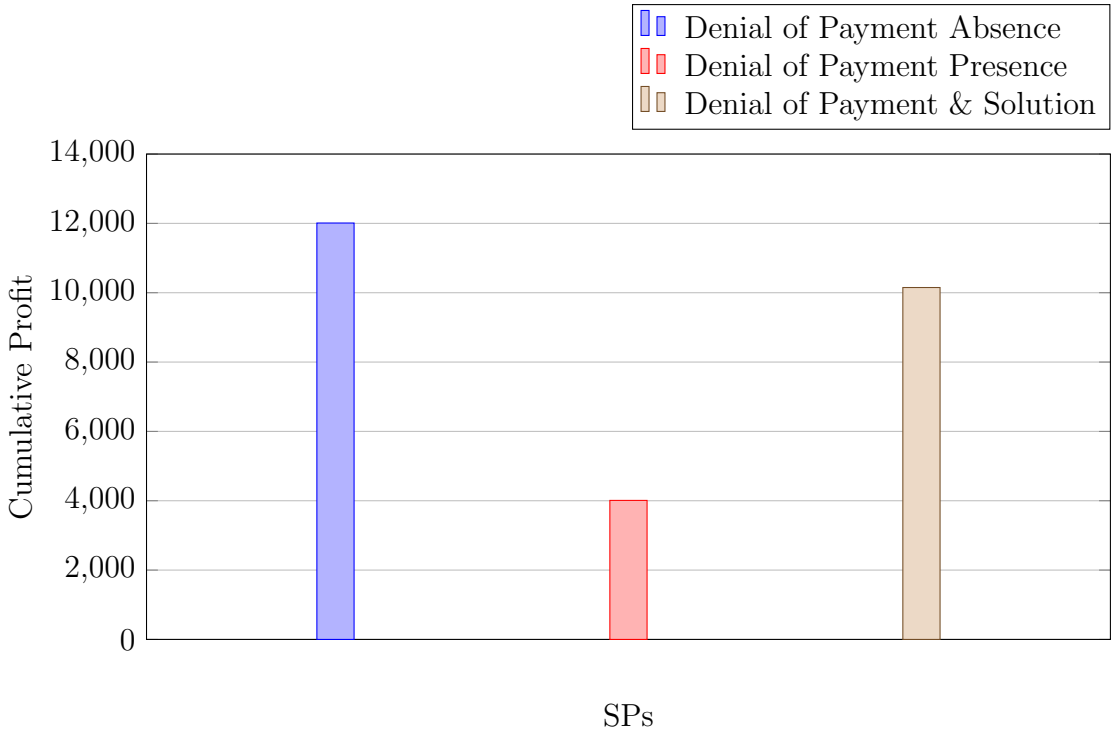


Figure 5.2: The Cumulative Profit of Sellers in the Absence, Presence of Denial of Payment Attacks and our Solution.

the detection threshold $k$ from 3 to 2. The results were contradictory, as the detection rate increased to 97.3%, but it also miss-classified 6% of the legitimate bidders as malicious. The increase in false positives can be attributed to the lack of sufficient bidding data, for some of the bidders, due to the short period they were registered in the market.

## 5.3 Summary

In this chapter, we have used the observations of the experiments conducted in Chapter 4 to engineer defensive mechanisms for securing market-oriented Clouds against market-specific attacks and in particular: Shill bidding, monopoly, reputation attack and denial of payment attack.

We have deployed our candidate solutions in the CloudAuction market to experimentally illustrate and report on the added value of introducing these mechanisms to secure electronic markets. We then asserted the sensitivity of our candidate defensive

mechanisms to the aforementioned attacks, where applicable, by adjusting the detection thresholds of our defensive mechanisms.

Our results demonstrated that market-specific defensive mechanisms can assist security engineers to deliver more dependable market-oriented Cloud systems, where the incentives, objectives and profit of bidders and sellers are not threatened. Although we have identified a few methods that attackers can exploit to bypass our candidate solutions, the attackers are still penalised as it is significantly harder to perform these attacks and the expected gains are considerably lessened.

The following chapter, Chapter 6, conducts a qualitative and reflective evaluation of the thesis with respect to our established research questions.

CHAPTER 6

# FURTHER EVALUATION AND REFLECTION

The techniques introduced in this thesis have been separately evaluated in their respective chapters. Throughout this work we have followed an experimentally-driven methodology, grounded on simulation, to examine the proposed techniques. The use of simulation as a suitable tool for evaluating our work has been justified in section 3.4. The aim of this chapter is to determine the extent to which this thesis has addressed the research questions reported in Chapter 1, discuss and reflect on the evaluation carried on in previous chapters.

## 6.1 Asset-Centric vs Aggregated Security

| Can asset-centric security be more cost-effective and efficient for the satisfaction of the runtime security goals of multiple assets compared to aggregated security? |
|---|

Throughout this thesis, we have promoted the enforcement of ad-hoc security that adheres to the runtime security requirements and constraints of individual assets. In spite of the importance of asset-centric security, our literature review (subsection 2.4.7) signified that existing self-adaptive security solutions have not been extensively concerned with asset-centric security. As a result, there is a pressing need for the development of self-adaptive security solutions that are able to handle complex and diverse requests that necessitate the enforcement of different security policies for different assets.

In Chapter 3, we have proposed a novel self-adaptive security approach that considers assets as independent entities with a need for customised, ad-hoc security. To assert the effectiveness of the proposed asset-centric security framework, in subsection 3.4.1, we have compared it with existing aggregated security solutions in terms of security goal satisfaction and computational costs. More specifically, our experiment entailed securing 50 files with both asset-centric and aggregated security (i.e., Cubby) to draw conclusions concerning the effectiveness of each method. The acquired results demonstrated that the usage of asset-centric approaches can be effective for the continuous satisfaction of the security goals of individual assets due to the provision of ad-hoc security (i.e. services and resources) that is tailored to the explicit requirements and constraints of each asset. On the contrary, the use of aggregated security has failed to satisfy the security needs and constraints of a vast number (i.e. 46 %) of the assets, with the provision of excessive or inadequate security. In terms of computational costs, aggregated security exhibited an excessive use of underlying resources (compared to asset-centric security) due to the over-provisioning of obsolete services and resources to users.

Despite the witnessed benefits of asset-centric security we have also shown, in subsection 3.4.4, that the computational overhead associated with the enforcement of ad-hoc security can be significantly higher than aggregated security approaches. In particular, our results signify that the proposed asset-centric solution requires up to 68% more memory (Figure 3.5) and up to 50% more CPU (Figure 3.6) resources to operate, compared to aggregated security solutions that follow the paradigm of "one solution fits all". Despite the high computational overhead witnessed by the proposed framework the benefits arising from asset-centric security can outweigh the strain imposed on resources. Although we have provided a speculation of the imposed computational overhead of our framework, the given estimation cannot be perceived as an accurate representation of the expected overhead in real-life due to the lack of networking operations in conjunction with the over-simplified user behaviour replicated by agents in our prototype simulation system.

## 6.2 Elastic, Cost-effective Mechanisms for Asset-Centric Security In the Cloud

| What techniques/mechanisms can be employed in the Cloud to secure multiple assets in a proactive, cost-effective and elastic manner? |
|---|

Whilst a naive bruteforce mechanism can be adequate for the identification and selection of suitable services and resources for security in the Cloud, the costs associated (i.e. computational and time) with these approaches are excessive. Due to the ultra-large and dynamic nature of the Cloud, it is imperative that lightweight and scalable security mechanisms are used for warranting the continuous satisfaction of the security requirements and constraints of multiple assets. Despite the need for scalability, our literature review has signified that dimensions such as elasticity (subsection 2.4.9) and cost-effectiveness (subsection 2.4.15) have been heavily neglected by existing self-adaptive security solutions, which make them ineffective for ultra-large and dynamic environments such as the Cloud. To overcome this challenge, in Chapter 3, we have motivated the usage of market-inspired methodologies as an effective optimisation mechanism for securing assets in the Cloud.

The choice of market as a candidate solution is heavily grounded on its proven effectiveness in dynamic allocation problems, witnessed by its wide usage in ultra-large environments [17], [18], [88], [91], [96], [97]. The decentralised decision-making nature of markets promotes the engineering of scalable (computations and decisions are performed in a decentralised manner) and dependable solutions (users can handle their own security requirements and data). Market approaches allow both users and providers to make their own decisions for maximising their utility and regulate the supply and demand of services and resources at market equilibrium. In the presence of limited resources, auctioning mechanisms can promote the effective allocation of services and resources by prioritising security requests based on their criticality (reflected in the bidding prices); thus, ensuring their provision to users that face an imminent threat. Lastly, the use of

market methodologies enable users to express their preferences concerning the type of resources needed and the time they want to acquire them. Thus, enabling a system to manage excess demand by spreading it out over time [91].

### 6.2.1 Market vs Non-Market Mechanisms

| Can market-inspired mechanisms be more effective in the allocation of services and resources compared to conventional non-market mechanisms in the presence of scarce resources in security constrained environments? |
|---|

Even though the market has been extensively used, the minute we factor security into the problem the available services and resources that could support security in a system may be narrowed down, thus making them scarce.

To determine whether the market can be effective for supporting security, in subsection 3.4.2, we have used our prototype system to compare how non-market and market-inspired mechanisms (i.e., our Posted-Offer variant model) can operate in the existence of scarce resources. Although the non-market mechanism was able to identify and allocate resources to users faster than our market mechanism, due to its simplistic nature (absence of coordination and bargaining), it was ineffective in prioritising security requests. Its simplistic first-come-first-served allocation policy allowed a large number of users that did not require the immediate usage of services and resources to reserve and waste them while refusing service to users that faced an imminent threat.

On the contrary, the usage of our Posted-Offer variant model demonstrated that market-inspired methodologies can be effective for the allocation of Cloud services and resources by allowing users to specify the significance of their security requests in their bidding prices. By doing so, the auctioning mechanism is able to prioritise bids according to the criticality of their security requests and provide service to users that face an imminent threat first. Demonstrating that even when we add more constraints (because of security) to the operating environment and services/resources are scarce, market mechanisms are still able to operate in an effective and dependable manner for securing multiple

assets. However, we need to state that different auctioning models can produce different results when compared to non-market mechanisms. In some situations, it can even be more effective to use non-market methodologies compared to certain market models.

### 6.2.2 Evaluating Auctioning Models

| How can the use of representative auction models influence the security and performance of the proposed system? |
|---|

Different auctioning models can bring different characteristics, benefits and limitation to the operating environment. To better understand how different auctioning mechanisms can influence the security and performance of the proposed solution, in subsection 3.4.4, we have analysed two auctioning algorithms, namely the English auction and a variant of the Posted-Offer auction model, and compared their exhibited performance and implications on security. Throughout this test, we have observed significant differences between the two algorithms. English auction demonstrated a more complex, time-consuming procedure that necessitate an average of 789.8 milliseconds to establish an auction match in contrast to the 577.8 milliseconds required by our Posted-Offer variant algorithm. As we have stated in subsection 3.4.4, the time difference exhibited between the two auction models may seem negligible at first, however it can play a major role when the number of users and assets significantly increase in a market which can exponentially increase the time needed for processing these requests. As security is time-critical, the security of assets needs to be adapted efficiently by avoiding time-delays which can cause the compromise of assets. Additionally, the English auction algorithm displayed signs of high computational overhead, reaching in some occasions 22% more than the CPU overhead generated by the Posted-Offer variant model. On the contrary, our Posted-Offer model proven an efficient and scalable procedure with low computational overhead, due to the lack of complex negotiations between SPs and user agents. Therefore, we argue that the Posted-Offer model is more suitable for occasions where user agents face an imminent

threat and time is a critical aspect of adapting security. On the contrary, the English auction is more fitting for occasions where a user agent is not constrained by time, its security is not threatened and is willing to sacrifice time to discover cheap and/or scarce services and resources.

Though we have only considered/examined the English auction and a variant of the Posted-Offer model as our auctioning algorithms, the proposed system can be instantiated with various auctioning mechanisms according to the needs of each market and its users. The choice of the English and Posted-Offer auctioning models can be attributed to their vast acknowledgement and usage in economics and computing, in conjunction with their highly divergent nature, which allows us to test our framework with the deployment of two distinct auctioning approaches. In the case where the selected auctioning algorithms are replaced different results will be witnessed. Based on the application environment and the security objectives of a system different auctioning models must be employed to acquire the desired qualities and results.

### 6.2.3 Learning Approaches for Security in the Market

| Can learning algorithms be used to arrive on more efficient bidding plans (identify and short list appropriate candidate solutions), instead of entering bidders into an exhaustive bidding for candidate offers? |
|---|

In Chapter 3, we have examined whether learning algorithms can be used to automate security while allowing bidders to arrive on more efficient bidding plans for candidate offers. To answer this question, in subsection 3.4.3, we have trained various learning algorithms and tested their classification accuracy against 5000 test samples (describing various scenarios in a university application environment) to identify the best learning algorithm for our solution. Our results identified the Random Forest algorithm as the dominant learning approach, witnessing a correct classification rate of 90.87% of the examined test samples. The acquired results demonstrated that learning algorithms can practically assist bidders in identifying appropriate services/resources (i.e. short listing

candidate solutions) for security in an automated and effective manner without the need for examining all available services and resources. In the occasion where no learning algorithms are employed, bidders need to perform additional effort to interact with the market (e.g take appropriate security decisions and manually deploy security measures) which can introduce significant delays and consequently stake the security of assets. As well as to damage the scalability of a system due to the employment of ineffective bidding methodologies. Furthermore, the learning approach allows users to form and deploy their own security (given that a learning algorithm is trained from a user's previous experiences), instead of relying on the biased security enforced by various Cloud administrators. By doing so it is feasible to overcome the "acceptable adaptation boundaries" challenge described in subsection 2.4.1. As users will enforce their own security, they can ensure that all enforced security adaptation strategies adhere to their specific requirements and policies, thus avoiding the execution of security measures that violate their policies.

Although we have witnessed a high classification rate by the Random Forest algorithm, we have argued that the accuracy of the algorithm is explicitly linked to the amount and consistency of the data used for training the algorithm. To identify the effects of limited training samples and unclassified/unknown test samples, we have examined the Random Forest algorithm with a varying number of training data samples from which some of them contained new values that were unknown to the classifier (i.e. new services and resources). The acquired results demonstrated that even when the Random Forest algorithm was trained with just 200 samples, it was still able to correctly classify 84.6% of them. However, it was unable to identify the new, unclassified parameters introduced. As the security decisions of classifiers are grounded on previous experiences, the introduction of new distinct parameters can remain unnoticed by learners. To overcome this challenge, users need to manually set a number of policies concerning the newly introduced values and re-train their classifiers or switch to the usage of semi-classifiers or clustering algorithms. Lastly, to ensure that the security of assets will not be compromised, users can set specific fail-safe policies that will be enforced whenever the classification rate of an asset

is below a certain threshold. By doing so it is feasible to enforce automated security even in situations where there is insufficient data for achieving high classification accuracy.

### 6.2.4 Threat-Aware Markets

| Can market-inspired mechanisms provide a dependable and secure optimisation tool for securing assets in the Cloud? |
|---|

As the proposed self-adaptive security solution is grounded on market-inspired methodologies, it is imperative to determine whether markets are secured against market-specific attacks. To assert the dependability of markets, in section 4.2, we have deployed four market-specific attacks (i.e. Shill bidding, Monopoly, Reputation attack and Denial of Payment attack) in CloudAuction, a market-oriented Cloud simulation tool, to identify the effects of each attack on bidders, sellers and underlying auctioning mechanism (i.e. combinatorial double auction). Our results have signified that existing markets are vulnerable towards market-specific attacks and demonstrated that the manipulation of underlying auctioning mechanisms can severely damage the incentives and profits of sellers and bidders respectively.

Grounded on our findings on market-specific threats, we have used the observations of our experiments to promote and engineer candidate, lightweight and scalable defensive mechanisms for securing market-oriented Clouds against the four selected market-specific attacks. Our efforts, in Chapter 5, have demonstrated that market-specific defensive mechanisms can thwart malicious attackers from manipulating auctioning algorithms, as well as to increase the profit and incentives of both bidders and sellers in the market. In particular, the experiments conducted in the presence of attackers and our defensive mechanisms demonstrated similar price fluctuations and profit margins to the ones exhibited in a healthy market (absence of attackers).

Despite the witnessed benefits arising from the deployment of our defensive mechanisms, we have also identified some occasions where attackers can bypass our solutions and still perform these attacks. However, the attackers are still penalised as it is significantly

harder to perform their attacks with their expected gains are considerably lessened. Furthermore, this thesis has restricted its analysis and mitigation of market-specific threats to a small subset of the available threats and in no way has foolproof markets against these threats. Our work on market-inspired threats can be a guide for further, in-depth research in the area. Finally, the applicability of the proposed defensive mechanisms is restricted to homogeneous markets. This arises from the varying nature and composition of different market systems, which render our defensive mechanisms system and case specific.

## 6.3 Summary

This chapter has provided further evaluation of the research questions of this thesis. The specific techniques introduced in preceding chapters have been evaluated in small, however, this chapter focuses on the big questions and the extent to which this thesis has addressed them.

We have demonstrated that our work has adequately addressed all the aforementioned research questions, but we have also identified multiple omissions. In terms of scalability, we have judged our work in terms of the effectiveness of the market concerning the optimisation of the usage of scarce resources as well as the efficiency of bidding for the identification of appropriate services and resources.

The next chapter, Chapter 7, summarises the contributions and the implications of this work. It then presents limitations of our work along with our thoughts on possible directions for future research and their potential impact on the field.

# CHAPTER 7

# CONCLUSION AND FUTURE DIRECTIONS

## 7.1 Thesis Contributions Revisited

This thesis promotes the engineering of elastic self-adaptive security solutions for the Cloud that consider the varying nature of assets and their need for customised, ad-hoc security. The thesis conducts (Chapter 2) an in-depth literature review on self-adaptive security methodologies in open, ultra-large environments, signifying a pressing need for the development of scalable, lightweight self-adaptive security systems for the Cloud. Driven by the taxonomic findings of Chapter 2 and the thesis research questions in Chapter 1, we have looked at market-inspired techniques as a candidate solution. In particular, Chapter 3 outlines a novel self-adaptive architecture which draws inspiration from market-inspired methodologies (i.e. English auction and a variant of the Posted-Offer model) and learning approaches (i.e. Random Forest classifier). The proposed approach manages the changing security requirements/goals of assets via the selection of shared, on-demand services and underlying resources while catering for their monetary and computational constraints. The use of auction procedures enable the proposed framework to deal with the scale of the problem and the trade-offs that can arise due to the self-interested and diverse nature of security requests coming from the assets of various users. Whereas, the use of a supervised learning technique allows our framework to operate in a proactive and automated fashion by detecting runtime anomalies that could be indicators of possible

security threats and mitigating them prior to their manifestation. By using the learning approach it is feasible to arrive on more efficient bidding plans, informed by historical data. Instead of entering the bidders into an exhaustive bidding for candidate offers, the learning helped us to effectively identify optimal security strategies (short list appropriate solutions) for securing assets.

As the proposed solution is grounded on market-inspired methodologies, this thesis has also been concerned with asserting the dependability of market mechanisms. Chapter 4, follows an experimentally driven approach to identify market-specific security limitations in the engineering of commonly used market-oriented Cloud mechanisms. We have used a market-oriented Cloud simulation framework, CloudAuction, to demonstrate that the designs of existing markets are limited when facing market-specific attacks and when thwarting malicious bidders and sellers from manipulating auction mechanisms for personal gain. Grounded on the observations made in our experiments with market-specific security attacks, Chapter 5 proposes and delivers candidate defensive mechanisms for securing market-oriented Clouds against selected market-specific attacks. To evaluate the effectiveness of our candidate solutions, we have deployed them in the CloudAuction simulation tool and analysed how the market is affected in the absence and presence of the selected attacks and the proposed solutions. We have then designated the added value of market-specific defensive mechanisms in the Cloud and promoted the engineering of threat-aware electronic markets for warranting the secure operation of bidders, sellers and auctioning mechanisms in market-oriented Clouds.

In particular, this thesis makes the following contributions:

- A literature review on self-adaptive security methodologies from the context of open, ultra-large environments and the architectural characteristics enabling their effective application in these environments. We propose a novel architecture-centric taxonomy for mapping and comparing the current research directions in the field. We reflect on the taxonomic findings and discuss design principles, limitations and research challenges in the current-state-of-art and practice. We then highlight di-

rections for future research, contributing to the effective application of adaptive security systems in ultra-large, dynamic environments, such as the Cloud.

- A novel agent-based, market-inspired architecture for satisfying the changing security requirements and constraints of assets in the Cloud at runtime.

- A learning approach (i.e. Random Forest classifier) that allow bidders to arrive on more efficient bidding plans, informed by historical data. The learning helps to identify optimal security strategies for securing assets, instead of entering the bidders into an exhaustive bidding for candidate offers.

- An analysis of existing market-specific security threats and their impact/effect on modern market-oriented Clouds.

- A set of novel defensive mechanisms for thwarting market-specific attacks. Based on the observations made in our experiments with market-specific security attacks we develop candidate, lightweight defensive mechanisms for securing market-oriented Clouds against these threats.

The contributions of this thesis have covered various aspects of the Cloud, among others asset-centric security, software architectures, elasticity, self-adaptation, optimisation methodologies, learning approaches and threat-aware markets that will assist security software engineers to further advance self-adaptive security solutions for the Cloud in terms of elasticity, cost-awareness, dependability and effectiveness. Although this thesis has made some contributions, it is significant at this stage to recognise the shortcomings of the approach and analysis presented. In order to assert the applicability and effectiveness of our conceptual architecture, we have instantiated and developed a variant of our framework in a simulated environment. Though this approach helped us to evaluate the proposed solution on a larger scale, it is limited in terms of real-world application (application to a wider real-world problem is going to be part of our future work). Therefore, the acquired simulation results can only be interpreted as a mere indication that

our framework can possibly be effective for securing multiple assets in ultra-large and elastic environments, such as the Cloud. Despite, that we have identified this shortcoming in the early stages of this research, it was impossible for us to engineer an ultra-large Cloud system to conduct our experiments. Another limitation of this work is that the effectiveness of the engineered market-specific defensive mechanisms cannot be guaranteed over different market-oriented Clouds. This can be attributed to the system-specific nature of our mechanisms and in particular the system/case-specific values used for the detection parameters of each defensive mechanism. The varying nature and composition of each market make it impossible for us to guarantee similar detection rates to the ones witnessed in our experiments. Therefore, it is necessary for each marketer to identify the optimal threshold values for the parameters of our defensive mechanisms to tune the solution to the needs and idiosyncrasies of each market.

We hope that the results presented in this thesis will stimulate additional research on market-inspired methodologies for security.

## 7.2 Future Directions

This thesis identifies a number of future directions to further consolidate the effectiveness of asset-centric self-adaptive security approaches and the development of threat-aware market-oriented Cloud systems. They are described as below:

### 7.2.1 An In-Depth Analysis and Counteraction of Market-Specific Attacks

Although we were the first to examine market-specific attacks in the Cloud and propose candidate defensive mechanisms for securing market-oriented Clouds against them, illustrated by our work in [125], we hope that our work will steer the future research directions in the field and stimulate interest in examining new attacks. To warrant the secure operation of bidders, sellers and auctioning mechanisms in market-oriented Clouds, including

our framework, it is not sufficient to investigate a small number of market-specific threats. Instead, we need to conduct an exhaustive examination of the landscape of market-specific threats and identify novel methods for thwarting them. Therefore, future work benefiting from this investigation can expand the analysis and counteraction of market-specific attacks on a larger scale. One of the potential future directions is to investigate multiple concurrent attacks and identify possible effects of various co-deployed attacks in market-oriented Clouds.

Furthermore, as malicious users continuously advance their attack strategies, it is imperative that we identify appropriate methodologies that will enable market-specific defensive mechanisms to co-evolve (e.g. dynamically adjust detection thresholds) and face these threats. Future research should focus on encapsulating sophisticated intelligence in our defensive mechanisms to allow the effective detection and resolution of market-specific threats when the attack landscape changes.

### 7.2.2 Dynamic Transitioning Auctioning Mechanisms

This work has exploited the English and (a variant of) Posted-Offer auction models for securing assets at runtime. Despite the selection of the two algorithms, there is a vast number of candidate auctioning models, with each introducing various characteristics, benefits and limitations to the problem environment. The selection of a sole auctioning algorithm, can in some occasions prove limited in the long-run, due to various changes that a market environment can undergo (e.g. available services, the number of sellers and buyers etc.). A solution to this challenge is the selection and deployment of different auctioning mechanisms at runtime based on the changing needs and constraints of a market. Thus, this thesis aims as part of its future research to focus on the development of an intelligent transitioning mechanism that will be able to identify and deploy appropriate auctioning algorithms based on the runtime changes in a market for maximising the utility of bidders and sellers. To perform this, it is necessary to profile various auctioning models to determine what are the appropriate auctioning mechanisms

for each situation. Furthermore, suitable learning strategies are needed that will enable the autonomic selection of auctioning mechanisms at runtime, based on contextual and content changes.

### 7.2.3 Industrial Application

The fundamental of our framework can help a number of emerging applications and paradigms that are enabled by the Cloud and/or exhibit similar characteristics to the Cloud. As an example, the vision of smart cities has been enabled by advancement in computing infrastructure, including the Cloud. Over the last years, smart cities got an increasing number of followers as they offer self-aware, automated procedures for simplifying our day-to-day lives. Despite their significance, the devices in these smart cities can be used by governments, organisations and lone hackers for data trace analysis. These traces comprise semantics for individual preferences, physical locations and social relations. They can be collected via GPS, GSM, WIFI, Bluetooth, RFID, published locations, installed applications, etc. These traces can enable the unlawful surveillance of users along with the disclosure of personal data to unauthorised third parties. According to the work of Pan et al., [142] one of the main challenges for data trace analysis is to ensure the "fidelity of data for applications while warranting the privacy of users". Driven by [142], Smart cities can potentially benefit from our framework to protect user devices at runtime against traceability. Smart cities are essentially ultra-large environments, where multiple users necessitate lightweight, scalable mechanisms to identify and use different services (e.g. anonymity proxies, firewalls, GPS jammers, etc.) and resources for protecting their devices against various security threats that can trace a user. Each device can be considered as a heterogeneous entity, with a need for ad-hoc privacy. Some devices (e.g. laptop used for classified research) necessitate higher levels of security/privacy, whereas other devices (e.g. e-Reader for novels) require lower security/privacy. The use of market mechanisms as part of our solution has enabled us to deal with the scale of the problem and the trade-offs that can arise due to the self-interested and diverse nature of

the security requests coming from various assets in the smart city setting.

### 7.2.4 Revising MAPE-K Architecture for Security

There are many open questions for MAPE-K elements such as monitors, planners, executors, etc., each can require vertical and in-depth treatment. This thesis has focused to a big extent on the analysis and plan phases, where its novelty lies. To ensure the effective usage of the MAPE-K architecture for security, research shall also look at securing the fundamental components of the MAPE-K to deliver more dependable and threat-aware self-adaptive solutions. To demonstrate the need for an in-depth analysis and refinement of the MAPE-K architecture for security, we provide a simple example based on monitoring mechanisms. Existing monitors are often deployed without any means for the verification of the authenticity of the recorded data. This practice leaves monitors vulnerable to malicious attackers that wish to trick them by submitting false monitoring data. Therefore, enabling attackers to perform their malicious actions, such as forcing a self-adaptive system to deploy wrong countermeasures. Similarly to monitors, there are multiple elements in the MAPE-K architecture that are susceptible to various threats, which can severely affect the dependability of self-adaptive security solutions. Therefore, it is imperative that we integrate proofing mechanisms in the MAPE-K architecture to promote the engineering of more effective and threat-aware self-adaptive security solutions.

## 7.3 Concluding Remarks

This thesis makes a novel contribution to the field of Cloud security by introducing a self-adaptive approach for securing assets, as well as candidate defensive mechanisms for facing market-inspired threats in the Cloud. The thesis presents an in-depth study and solution that exploits the fundamental characteristics of markets and learning techniques

for securing Cloud-based assets in the absence of closely related work.

We believe that the proposed framework can assist security engineers and researchers to design and engineer more cost-effective, elastic and dynamic self-adaptive security solutions for the Cloud. The conducted experiments signify the effectiveness, scalability and dynamism of the proposed framework in satisfying the changing security requirements, priorities and constraints of multiple assets from different users. In addition, the experiments demonstrate that market-inspired methodologies can assist in converging to a better solution (even in the presence of scarce resources) by maximising the utility of the whole environment while securing assets that face an imminent threat first.

The contributions of this thesis have covered various aspects of the Cloud, among others asset-centric security, elasticity, self-adaptation, optimisation methodologies and threat-aware markets which aim to advance our understanding and state-of-the-art practices in Cloud security. We hope that the presented work will stimulate and steer further research in self-adaptive security solutions for the Cloud in terms of elasticity, cost-effectiveness, dynamism and threat-awareness.

# LIST OF REFERENCES


[1] Gmail. https://www.google.com/gmail/about. Accessed 10 June 2017.

[2] Microsoft Office 365. https://www.microsoftstore.com/store/mseea/cat/Office. Accessed 10 June 2017.

[3] Microsoft Azure. https://azure.microsoft.com. Accessed 10 June 2017.

[4] Google Cloud Platform. https://cloud.google.com/appengine. Accessed 10 June 2017.

[5] Amazon EC2. https://aws.amazon.com/ec2. Accessed 10 June 2017.

[6] Rackspace. https://www.rackspace.com. Accessed 10 June 2017.

[7] Compute Engine Scalable, High-Performance Virtual Machines. https://cloud.google.com/compute. Accessed 10 June 2017.

[8] Qualcomm, Future of security: From reactive to proactive. https://www.qualcomm.com/news/onq/2015/08/13/future-security-reactive-proactive. Accessed 10 June 2017.

[9] M.T. Khorshed, A.S. Ali and S.A. Wasimi. A survey on gaps, threat remediation challenges and some thoughts for proactive attack detection in cloud computing. Future Generation computer systems, 28(6):833-851, 2012.

[10] S. Mazur, E. Blasch, Y. Chen, and V. Skormin. Mitigating cloud computing security risks using a self-monitoring defensive scheme. In Aerospace and Electronics Conference (NAECON), 2011 IEEE International Conference on, pages 39-45, 2011.

[11] J. Xu, J. Yan, L. He, P. Su, and D. Feng. CloudSEC: a cloud architecture for composing collaborative security services. In Cloud Computing Technology and Science (CloudCom), 2010 IEEE 2nd International Conference on, pages 703-711, 2010.

[12] A. C. Squicciarini, G. Petracca, and E. Bertino. Adaptive data management for self-protecting objects in cloud computing systems. In Network and Service Management, 2012 IEEE 8th International Conference on, pages 140-144, 2012.

[13] S. Ma, and Y. Wang. Self-Adaptive Access Control Model Based on Feedback Loop. In Cloud Computing and Big Data (CloudCom-Asia), 2013 IEEE International Conference on, pages 597-602, 2013.

[14] ISO/IEC 27001 - Information security management. http://www.iso.org/iso/iso27001. Accessed 20 June 2017

[15] A. Liaw and M. Wiener. Classification and regression by Random Forest. R news, 2(3): 18-22, 2002.

[16] J. Ketcham, S. L. Vernon, and W. W. Arlington. A comparison of posted-offer and double-auction pricing institutions. The Review of Economic Studies 51(4):595-614, 1984.

[17] L. Jiayin, M. Qiu, J. W. Niu, Y. Chen, and Z. Ming. Adaptive resource allocation for preemptable jobs in cloud systems. In Intelligent Systems Design and Applications (ISDA), 2010 IEEE 10th International Conference on, pages 31-36, 2010.

[18] D. Shin, and H. Akkan. Domain-based virtualized resource management in cloud computing. In Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2010 IEEE 6th International Conference on, pages 1-6, 2010.

[19] P. Ruth, J. Rhee, D. Xu, R. Kennell, and S. Goasguen. Autonomic Adaptation of virtual computational environments in a multi-domain infrastructure. In Autonomic Computing (ICAC'06), 2006 IEEE International Conference on, pages 5-14, 2006

[20] G. Tziakouris, C. J. Mera Gomez, R. Bahsoon. Securing Cloud Users at Runtime via a Market Mechanism: A Case for Federated Identity. In High Performance Computing and Communications (HPCC), 2014 IEEE 11th International Conference on, pages 221-228, 2014.

[21] Shill bidding policy. http://pages.ebay.com/help/policies/seller-shill-bidding.html. Accessed 23 Mar. 2017. Accessed 20 June 2017.

[22] Resolving Feedback problems. http://pages.ebay.com/help/feedback/feedback-disputes.html. Accessed 23 June 2017.

[23] E. H. Chamberlin. The theory of monopolistic competition. Cambridge, MA: Harvard University Press, 1933.

[24] S. Mikhail. What To Do When A Client Refuses To Pay. Forbes. Forbes Magazine, 2015.

[25] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya. CloudSim: a toolkit for the modelling and simulation of cloud resource management and application provisioning techniques. Journal of Software: Practice and Experience, 41(1):23-50, 2010.

[26] V. Kotov. Systems of systems as communicating structures. Hewlett Packard Laboratories, 1997.

[27] G. Tziakouris, R. Bahsoon, T. Chothia, and A. Babar. A Survey on Self-Adaptive Security for Large Scale Open Environments. ACM Computing Surveys (Under Review), 2017.

[28] A. Elkhodary and J. Whittle. A survey of approaches to adaptive application security. In Software Engineering for Adaptive and Self-Managing Systems, 2007 IEEE International Workshop on, page 16, 2007.

[29] D. Ghosh, R. Sharman, H. R. Rao and S.Upadhyaya. Self-healing systems survey and synthesis. Decision Support Systems, 42(4):2164-2185, 2007.

[30] M. Salehie, and L. Tahvildari. Self-adaptive software: Landscape and research challenges. ACM Transactions on Autonomous and Adaptive Systems (TAAS), 4(2):14, 2009.

[31] E. Yuan, and S. Malek. A taxonomy and survey of self-protecting software systems. In Software Engineering for Adaptive and Self-Managing Systems, 2012 IEEE 7th International Symposium on, pages 109-118, 2012.

[32] E. Yuan, N. Esfahani and S. Malek. A systematic survey of self-protecting software systems. ACM Transactions on Autonomous and Adaptive Systems (TAAS), 8(4):17, 2014.

[33] C. Cowan, C. Pu, D. Maier, J. Walpole, P. Bakke, S. Beattie, A. Grier, P. Wagle, Q. Zhang and H. Hinton. StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks. In Usenix Security, 98:63-78, 1998.

[34] B. L. Lai, and L.W. Chang. Adaptive data hiding for images based on harr discrete wavelet transform. In Image and Video Technology, 2006 Springer Pacific-Rim Symposium on, pages 1085-1093, 2006.

[35] R.O. El Safy, H.H. Zayed, and A. E. Dessouki. An adaptive steganographic technique based on integer wavelet transform. In Networking and Media Convergence (ICNM), 2009 IEEE International Conference on, pages 111-117, 2009.

[36] Y. Dodis, A. Sahai, and A. Smith. On perfect and adaptive security in exposure-resilient cryptography. In Theory and Applications of Cryptographic Techniques, 2001 Springer International Conference on, pages 301-324, 2001.

[37] A. Avizienis, J. C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. IEEE transactions on dependable and secure computing, 1(1):11-33, 2004.

[38] J. Siljee, I. Bosloper, J. Nijhuis, and D. Hammer. DySOA: making service systems self-adaptive. In Service-Oriented Computing, 2005 Springer International Conference on, pages 255-268, 2005.

[39] C. Chigan, L. L. and Y. Ye. Resource-aware self-adaptive security provisioning in mobile ad hoc networks. In Wireless Communications and Networking Conference, 2005 IEEE vol. 4, pages 2118-2124, 2005.

[40] M. Carney, and B. Loe. A comparison of methods for implementing adaptive security policies. In USENIX Security, 1998 7th International symposium on, pages 1-14, 1998.

[41] R. Buyya, C. S. Yeo, and S. Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. High Performance Computing and Communications (HPCC), 2008 IEEE International Conference on, pages 5-13, 2008.

[42] F. Dressler. Bio-inspired mechanisms for efficient and adaptive network security mechanisms. In Dagstuhl Seminar, Schloss Dagstuhl-Leibniz-Zentrum far Informatik, 2005.

[43] B.Yuriy, and N. Medvidovic. An Architectural Style for Solving Computationally Intensive Problems on Large Networks. In Software Engineering for Adaptive and Self-Managing Systems, 2007 IEEE International Conference on, pages 26-27, 2007.

[44] A. Toninelli, R. Montanari, L. Kagal, and O. Lassila. A semantic context-aware access control framework for secure collaborations in pervasive computing environments. In International semantic web Conference, Springer, pages 473-486, 2006.

[45] S. H. Son, R. Zimmerman, and J. Hansson. An adaptable security manager for real-time transactions, In Real-Time Systems, 2000 IEEE International Conference on, pages 63-70, 2000.

[46] A. Saxena, M. Lacoste, T. Jarboui, U. Lucking, and B. Steinke. A software framework for autonomic security in pervasive environments. In Information Systems Security, 2007 Springer International Conference on, pages 91-109, 2007.

[47] M. Y. Hsieh, Y. M. Huang, and H. C. Chao. Adaptive security design with malicious node detection in cluster-based sensor networks. In Computer Communications, 2007 IEEE International Conference on, pages 2385-2400, 2007.

[48] J. Xu, J. Yan, L. He, P. Su, and D. Feng. CloudSEC: a cloud architecture for composing collaborative security services. In Cloud Computing Technology and Science, 2010 IEEE International Conference on, pages 703-711, 2010.

[49] H. Abie, and I. Balasingham. Risk-based adaptive security for smart IoT in eHealth. In Body Area Networks, 2012 IEEE 7th International Conference on, pages 269-275, 2012.

[50] S. Mazur, E. Blasch, Y. Chen, and V. Skormin. Mitigating cloud computing security risks using a self-monitoring defensive scheme. In National Aerospace and Electronics Conference (NAECON), 2011 IEEE International Conference on, pages 39-45, 2011.

[51] J. Li, B. Li, T. Wo, C. Hu, J. Huai, L. Liu, and K. P. Lam. CyberGuarder: A virtualization security assurance architecture for green cloud computing. In Future Generation Computer Systems 28(2):379-390, 2012.

[52] J. Du, N. Shah, and X. Gu. Adaptive data-driven service integrity attestation for multitenant cloud systems. In Quality of Service, 2011 IEEE 19th International Workshop on, page 29, 2011.

[53] J. Youngmin, and M. Chung. Adaptive security management model in the cloud computing environment. In Advanced Communication Technology, 2010 IEEE 12th International Conference on, pages 1664-1669, 2010.

[54] S. Ma, and Y. Wang. Self-Adaptive Access Control Model Based on Feedback Loop. In Cloud Computing and Big Data, 2013 IEEE International Conference on, pages 597-602, 2013.

[55] C. G. Yee, W. H. Shin, and G. S. V. R. K. Rao. An adaptive intrusion detection and prevention (ID/IP) framework for web services. In Convergence Information Technology, 2007 IEEE International Conference on, pages 528-534, 2007.

[56] J. Kong, H. Luo, K. Xu, D. L. Gu, M. G. and S. Lu. Adaptive security for multilevel ad hoc networks. In Wireless Communications and Mobile Computing, 2002 IEEE International Conference on, pages 533-547, 2002.

[57] D. M. Farid, and M. Z. Rahman. Anomaly network intrusion detection based on improved self adaptive bayesian algorithm. Journal of computers 5(1):23-31, 2010.

[58] S. Kurosawa, H. Nakayama, N. Kato, A. Jamalipour, and Y. Nemoto. A self-adaptive intrusion detection method for AODV-based mobile ad hoc networks. In Mobile Adhoc and Sensor Systems, 2005 IEEE International Conference on, pages 773-780, 2005.

[59] P. Robertson, and R. Laddaga. Adaptive security and trust. In Self-Adaptive and Self-Organizing Systems, 2012 IEEE International Workshop on, pages 55-60, 2012.

[60] P. Penfield. Principle of Maximum Entropy: Simple Form. Massachusetts: Massachusetts Institute of Technology, 2003.

[61] PA Schneck, and K. Schwan. Authenticast: an adaptive protocol for high-performance, secure network applications. Georgia Institute of Technology, 1997.

[62] J. Zou, K. Lu, and Z. Jin. Architecture and fuzzy adaptive security algorithm in intelligent firewall. In Military Communications, 2002 IEEE International Conference on, pages 1145-1149, 2002.

[63] G. Tedesco, and U. Aickelin. An immune inspired Network Intrusion Detection System utilizing correlation. SSRN, 2006.

[64] A. Awais, M. Farooq and M. Y. Javed. Attack analysis and bio-inspired security framework for IPMultimedia subsystem. In companion on Genetic and evolutionary computation, 2008 IEEE 10th International Conference on, pages 2093-2098, 2008.

[65] C. Bailey, D. W. Chadwick, and R. D. Lemos. Self-Adaptive Authorization Framework for Policy Based RBAC/ABAC Models. In Dependable, Autonomic and Secure Computing, 2011 IEEE 9th International Conference on, pages 182-196, 2011.

[66] B. Foo, Y. S. Wu, Y. C. Mao, S. Bagchi, and E. Spafford. ADEPTS: adaptive intrusion response using attack graphs in an e-commerce environment. In Dependable Systems and Networks (DSN'05), 2005 IEEE International Conference on, pages 508-517, 2005.

[67] R. Hassan, and H. Abdellatif. Self-Adaptive Security for Mobiles Agents. International Journal of Computer Applications, 94:24-29, 2014.

[68] C. Tsigkanos, L. Pasquale, C. Menghi, C. Ghezzi, and B. Nuseibeh. Engineering topology aware adaptive security: Preventing requirements violations at runtime. In Requirements Engineering (RE), 2014 IEEE 22nd International Conference on, pages 203-212, 2014.

[69] M. Salehie, L. Pasquale, I. Omoronyia, R. Ali, and B. Nuseibeh. Requirements-driven adaptive security: Protecting variable assets at runtime. In Requirements Engineering (RE), 2012 IEEE 20th International Conference on, pages 111-120, 2012.

[70] H. Shrobe, R. Laddaga, B. Balzer, N. Goldman, D. Wile, M. Tallis, T. Hollebeek, and A. Egyed. Self-Adaptive systems for information survivability: PMOP and AW-DRAT. In Self-Adaptive and Self-Organizing Systems (SASO), 2007 IEEE 1st International Conference on, pages 332-335, 2007.

[71] P. K. Harmer, P. D. Williams, G. H. Gunsch, and G B. Lamont. An artificial immune system architecture for computer security applications. IEEE transactions on evolutionary computation 6(3):252-280, 2002.

[72] H. Abie, R. M. Savola, and I. Dattani. Robust, secure, self-adaptive and resilient messaging middleware for business critical systems. In Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns (COMPUTATIONWORLD'09), 2009 IEEE International Conference on, pages 153-160, 2009.

[73] R. M. Venkatesan, and S. Bhattacharya. Threat-adaptive security policy. In Performance, Computing, and Communications (IPCCC), 1997 IEEE International Conference on, pages 525-531, 1997.

[74] C. Carver, J. M. Hill, J. R. Surdu, and U. W. Pooch. A methodology for using intelligent agents to provide automated intrusion response. In Systems, Man, and

Cybernetics Information Assurance and Security, 2000 IEEE International Workshop on, pages 110-116, 2000.

[75] A. Evesti, J. Suomalainen, and E. Ovaska. Architecture and knowledge-driven self-adaptive security in smart space. International Journal of Computers 2(1):34-66, 2013.

[76] M. E. Locasto, K. Wang, A. D. Keromytis, and S. J. Stolfo. Flips: Hybrid adaptive intrusion prevention. In Recent Advances in Intrusion Detection, 2005 Springer International Workshop on, pages 82-101, 2005.

[77] D. Garlan, and B. Schmerl. Model-based adaptation for self-healing systems. In Self-healing systems, 2002 ACM 1st workshop on, pages 27-32, 2002.

[78] I. Georgiadis, J. Magee, and J. Kramer. Self-organising software architectures for distributed systems. In Self-healing systems, 2002 ACM 1st workshop on, pages 33-38, 2002.

[79] H. Aydt, S.J. Turner, W. Cai, and M.Y.H. Low. Research issues in symbiotic simulation. In Simulation Conference (WSC), 2009 IEEE International Conference on, pages 1213-1222, 2009.

[80] H. J. Goldsby, P. Sawyer, N. Bencomo, B. H. Cheng, and D. Hughes. Goal-based modelling of dynamically adaptive system requirements. In Engineering of Computer Based Systems, 2008 IEEE International Conference on, pages 36-45, 2008.

[81] E. Letier, and A. V. Lamsweerde. Reasoning about partial goal satisfaction for requirements and design engineering. ACM SIGSOFT Software Engineering Notes, 29(6):53-62, 2004.

[82] J. Whittle, P. Sawyer, N. Bencomo, B. H. Cheng, and J. M. Bruel. Relax: Incorporating uncertainty into the specification of self-adaptive systems. In Requirements Engineering (RE), 2009 IEEE International Conference on, pages 79-88, 2009.

[83] A. Elkhodary, N. Esfahani, and S. Malek. FUSION: a framework for engineering self-tuning self-adaptive software systems. In Foundations of software engineering, 2010 ACM 18th SIGSOFT International symposium on, pages 7-16, 2010.

[84] M. C. Huebscher, and J. A. McCann, A survey of autonomic computing degrees, models, and applications. International Journal of ACM Computing Surveys (CSUR), 40(3), 2008.

[85] J. C. Georgas, A. Hoek, and R. N. Taylor. Architectural runtime configuration management in support of dependable self-adaptive software, ACM SIGSOFT Software Engineering Notes 30(4):1-6, 2005.

[86] J. A. McCann, R. D. Lemos, M. Huebscher, O. F. Rana, and A. Wombacher. Can self-managed systems be trusted? some views and trends. The Knowledge Engineering Review, 21(3):239-248, 2006.

[87] Y. Fu, J. Chase, B. Chun, S. Schwab, and A. Vahdat. SHARP: An architecture for secure resource peering. ACM SIGOPS Operating Systems Review 37(5):133-148, 2003.

[88] K. Lai, L. Rasmusson, E. Adar, L. Zhang, and B. A. Huberman. Tycoon: An implementation of a distributed, market-based resource allocation system. International Journal of Multiagent and Grid Systems 1(3):169-182, 2005.

[89] P. Ruth, J. Rhee, D. Xu, R. Kennell, and S. Goasguen. Autonomic live adaptation of virtual computational environments in a multi-domain infrastructure. In Autonomic Computing, 2006 IEEE International Conference on, pages 5-14, 2006.

[90] D. Irwin, J. Chase, L. Grit, A. Yumerefendi, D. Becker, and K. Yocum. Sharing Networked Resources with Brokered Leases. In USENIX Annual Technical Conference, pages 199-212, 2006.

[91] A. AuYoung, B. Chun, A. Snoeren, and A. Vahdat. Resource allocation in federated distributed computing infrastructures. In Operating System and Architectural Support for the On-demand IT InfraStructure, 2004 IEEE 1st International Workshop on, 9, 2004.

[92] P. Vytelingum, D. Cliff, and N. R. Jennings. Strategic bidding in continuous double auctions. International Journal of Artificial Intelligence, 172(14):1700-1729, 2008.

[93] L. Baresi, and S. Guinea. Towards dynamic monitoring of WS-BPEL processes. In Service-Oriented Computing, 2005 Springer International Conference on, pages 269-282, 2005.

[94] R. Ali, A. Griggio, A. Franzin, F. Dalpiaz, and P. Giorgini. Optimizing monitoring requirements in self-adaptive systems. In Enterprise, Business-Process and Information Systems modelling, 2012 IEEE International Conference on, pages 362-377, 2012.

[95] Y. Wang, S. A. Mcilraith, Y. Yu, and J. Mylopoulos. Monitoring and diagnosing software requirements. International Journal of Automated Software Engineering 16(1):3-35, 2009.

[96] I. Stoica, H. Abdel-Wahab, and A. Pothen. A microeconomic scheduler for parallel computers. In Job Scheduling Strategies for Parallel Processing, 1995 Springer International Workshop on, pages 200-218, 1995.

[97] C. Waldspurger, and E. Weihl. Stride scheduling: Deterministic proportional share resource management. Technical report, Massachusetts Institute of Technology, 1995.

[98] G. Tziakouris, M. Zinonos, T. Chothia, and R. Bahsoon. Asset-centric Security-Aware Service Selection. In Big Data (BigData Congress), 2016 IEEE International Congress on, pages 327-332, 2016.

[99] G. Tziakouris, C. M. Gomez, F. Ramirez, R. Bahsoon, and T. Chothia. Economics-Inspired Self-Adaptive Security for Assets in Cloud. IEEE Transactions on Services Computing (Under Review), 2017.

[100] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. In Future Generation Computing Systems, 2009 IEEE International Conference on, pages 599-616, 2009.

[101] A. V. Lamsweerde. Goal-oriented requirements engineering: A guided tour. In Requirements Engineering, 2001 IEEE 5th International Symposium on, pages 249-262, 2001.

[102] B. Zhang, Y. Zhou, and C. Faloutsos. Toward a comprehensive model in internet auction fraud detection. In system sciences, 2008 IEEE 41st International conference on, pages 79-79, 2008.

[103] M. F. Balcan, A. Blum, J. D. Hartline, and Y. Mansour. Mechanism design via machine learning. In Foundations of Computer Science (FOCS), 2005 IEEE 46th International Symposium on, pages 605-614, 2005.

[104] P. Hummel, and R. P. McAfee. Machine learning in an auction environment. In WWW, 2014, pages 7-18.

[105] E. Bonabeau. Agent-based modelling: Methods and techniques for simulating human systems. In National Academy of Sciences, 99(3):7280-7, 2002.

[106] Computing Autonomic. An architectural blueprint for autonomic computing. IBM White Paper, 2006.

[107] H. Psaier, L. Juszczyk, F. Skopik, D. Schall, and S. Dustdar. Runtime behaviour monitoring and self-adaptation in service-oriented systems. In Socially Enhanced Services Computing, 2011 Springer International Conference on, pages 117-138, 2011.

[108] A. Souag, C. Salinesi, and I. Wattiau. Security requirements analysis based on security and domain ontologies. In Requirements Engineering: Foundations of Software Quality (REFSQ), 2013 International Conference on, pages 1-3, 2013.

[109] T. Li, J. Horkoff, and J. Mylopoulos. Integrating Security Patterns with Security Requirements Analysis Using Contextual Goal Models. In Practice of Enterprise modelling (PoEM), 2014 International Conference on, pages 208-223, 2014.

[110] A. Dardenne, A. V. Lamsweerde, and S. Fickas, Goal-directed requirements acquisition. International Journal of Science of computer programming, Elsevier, 20(1-2):3-50, 1993.

[111] E. H. Durfee, and V. R. Lesser. Partial global planning: A coordination framework for distributed hypothesis formation. IEEE Transactions on systems, Man, and Cybernetics, 21(5):1167-83, 1991.

[112] F. Gul, and E. Stacchetti. The English auction with differentiated commodities. International Journal of Economic theory, 92(1):66-95, 2000.

[113] Ebay. https://ebay.com/help/buy/bid-increments.html. Accessed 25 June 2017

[114] Cubby. https://www.cubby.com. Accessed 25 June 2017.

[115] G. Holmes, A. Donkin, and I. H. Witten. Weka: A machine learning workbench. In Intelligent Information Systems, 1994 IEEE 2nd Australian and New Zealand Conference on, pages 357-361, 1994.

[116] R. Caruana, and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In Machine learning, 2006 23rd international conference on, pages 161-168, 2006.

[117] C. Vecchiola, X. Chu, and R. Buyya. Aneka: a software platform for .NET-based Cloud computing. In High Speed and Large Scale Scientific Computing, 2009 International Conference on, pages 267-295, 2009.

[118] R. Buyya, and S. Venugopal. The gridbus toolkit for service oriented grid and utility computing: An overview and status report. In Grid Economics and Business Models (GECON), 2004 IEEE 1st International Workshop on, pages 19-66, 2004.

[119] C. Wang, and H. F. Leung. Anonymity and security in continuous double auctions for Internet retails market. In System Sciences, 2004 37th International Conference on, pages 10-17, 2004.

[120] H. Lipmaa, N. Asokan, and V. Niemi. Secure Vickrey auctions without threshold trust. In Financial Cryptography, 2003 Springer International Conference on, pages 87-101, 2003.

[121] Y. Xun, and C. K. Siew. Secure agent-mediated online auction framework. International Journal of Information Technology, pages 1-14, 2001.

[122] F. Brandt, Secure and private auctions without auctioneers, In Technical Report FKI-245-02. Institut fur Informatick, Technishce Universitat Munchen, 2002.

[123] D. Rolli, M. Conrad, D. Neumann, and C. Sorge. An asynchronous and secure ascending peer-to-peer auction. In Special Interest Group on Data Communication (SIGCOMM), 2005 ACM International workshop on Economics of peer-to-peer systems, pages 105-110, 2005.

[124] M. K. Franklin, and M. K. Reiter. The design and implementation of a secure auction service. Software Engineering, IEEE Transactions on, 22(5):302-312, 1996.

[125] G. Tziakouris, R. Bahsoon, T. Chothia, and R. Buyya. Thwarting Market Specific Attacks in Cloud. In Cloud Computing (CLOUD), 2016 IEEE 9th International Conference on, pages 35-42, 2016.

[126] W. Wang, Z. Hidvgi, and A. B. Whinston. Shill bidding in English auctions. Emory University and the University of Texas at Austin, mimeo, 2001.

[127] B. Bhargava, M. Jenamani, and Y. Zhong. Counteracting shill bidding in online English auction. International Journal of Cooperative Information Systems, 14(2.3):245-263, 2005.

[128] R. J. Kauffman, and C. A. Wood. Running up the bid: detecting, predicting, and preventing reserve price shilling in online auctions. In Electronic commerce, 2003 5th International Conference on, pages 259-265, 2003.

[129] J. Trevathan, and W. Read. Detecting shill bidding in online English auctions. Handbook of research on social and organizational liabilities in information security, pages 446-470, 2009.

[130] K. Hoffman, D. Zage, and C. Nita-Rotaru. A Survey of attacks on Reputation Systems. Purdue University Computer Science Technical Reports, 2007.

[131] Y. Sun, and Y. Liu. Security of online reputation systems: The evolution of attacks and defences. IEEE Signal Processing Magazine, 29(2):87-97, 2012.

[132] T.D. Huynh, N.R. Jennings, and N.R. Shabolt. An integrated trust and reputation model for open multi-agent systems. Journal of Autonomous Agents and Multi-Agent Systems, 13(2):119-154, 2006.

[133] J. Patel, W.T.L. Teacy, N.R. Jennings, and M. Luck. A Probabilistic Trust Model for Handling Inaccurate Reputation Sources. Springer-Verlag, LNCS 377, pages 193-209, 2005.

[134] A. Josang, R. Ismail, and C. Boyd. A survey of Trust and Reputation Systems for Online service provision. Journal of Decision Support Systems, 43(2):618-644, 2007.

[135] A. P. Machado. Corruption and Monopolies-An Endemic Problem. Forbes. http://www.forbes.com/sites/arthurmachado/2015/07/20/corruption-and-monopolies, 2015.

[136] J. Lazarev. The welfare effects of intertemporal price discrimination: an empirical analysis of airline pricing in US monopoly markets. New York University, 2013.

[137] J. Haucap, and U. Heimeshoff. Google, Facebook, Amazon, eBay: Is the Internet driving competition or market monopolization?. International Journal of Economics and Economic Policy, 11(1-2):49-61, 2014.

[138] R. Jain, and P. Varaiya. The combinatorial seller's bid double auction: an asymptotically efficient market mechanism. International Journal of Economics Theory, 2006.

[139] The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne. CloudSim: A Framework for modelling and Simulation of Cloud Computing Infrastructures and Services. http://www.cloudbus.org/cloudsim.

[140] L. Makowski, and J. M. Ostroy. Vickrey-Clarke-Groves mechanisms and perfect competition. Journal of Economic Theory, 42(2):244-61, 1987.

[141] L. Oswald. The law of marketing. Cengage Learning, 2010.

[142] G. Pan, G. Qi, W. Zhang, S. Li, Z. Wu, L. T. Yang. Trace analysis and mining for smart cities: issues, methods, and applications. IEEE Communications Magazine, 51(6):120-6, 2013.