COLLABORATIVE INTRUSION DETECTION IN FEDERATED CLOUD ENVIRONMENTS USING DEMPSTER-SHAFER THEORY OF EVIDENCE

By

Áine MacDermott BSc (Hons), AFHEA

A thesis submitted in partial fulfilment of the requirements of Liverpool John Moores University for the degree of Doctor of Philosophy

September 2017

Acknowledgements

I would like to thank my Director of Studies, Professor Qi Shi for his support, guidance and patience throughout the duration of my studies. I would also like to express my gratitude to Professor Madjid Merabti and Dr Kashif Kifayat. The support, guidance, and opportunities provided to me throughout the course of my research by my supervisory team have greatly contributed to its successful completion.

My friends and family have been brilliant over the duration of this project, giving me advice and comfort when needed. Special thanks to my parents Pat and Roisin, my aunt Kay, and my amazing siblings Danny, Seán, & Aoife who have been so supportive and encouraging throughout the duration of my studies. Thanks to my Bootle family Kate and Squishy too!

Special thanks go to the Department of Computer Science, Liverpool John Moores University for their funding of this project and the encouragement I received. I would also like to thank my colleagues and friends Paul Buck, Kirsty Lever, Tom Berry, Carol Oliver, Tricia Waterson, Mohammed Algamedi, Will Hurst, Chelsea Dobbins, Dave Tully, to name a few. Additionally, special thanks to Callum Green, Alex Hermon, Jamie Hubbard, and Laura Verdin.

Abstract

Moving services to the Cloud environment is a trend that has been increasing in recent years, with a constant increase in sophistication and complexity of such services. Today, even critical infrastructure operators are considering moving their services and data to the Cloud. As Cloud computing grows in popularity, new models are deployed to further the associated benefits. Federated Clouds are one such concept, which are an alternative for companies reluctant to move their data out of house to a Cloud Service Providers (CSP) due to security and confidentiality concerns. Lack of collaboration among different components within a Cloud federation, or among CSPs, for detection or prevention of attacks is an issue. For protecting these services and data, as Cloud environments and Cloud federations are large scale, it is essential that any potential solution should scale alongside the environment adapt to the underlying infrastructure without any issues or performance implications.

This thesis presents a novel architecture for collaborative intrusion detection specifically for CSPs within a Cloud federation. Our approach offers a proactive model for Cloud intrusion detection based on the distribution of responsibilities, whereby the responsibility for managing the elements of the Cloud is distributed among several monitoring nodes and brokering, utilising our Service-based collaborative intrusion detection – “Security as a Service” methodology. For collaborative intrusion detection, the Dempster-Shafer (D-S) theory of evidence is applied, executing as a fusion node with the role of collecting and fusing the information provided by the monitoring entities, taking the final decision regarding a possible attack. This type of detection and prevention helps increase resilience to attacks in the Cloud.

The main novel contribution of this project is that it provides the means by which DDoS attacks are detected within a Cloud federation, so as to enable an early propagated response to block the attack. This inter-domain cooperation will offer holistic security, and add to the defence in depth. However, while the utilisation of D-S seems promising, there is an issue regarding conflicting evidences which is addressed with an extended two stage D-S fusion process. The evidence from the research strongly suggests that fusion algorithms can play a key role in autonomous decision making schemes, however our experimentation highlights areas upon which improvements are needed before fully applying to federated environments.

Table of Contents

List of Figures

List of Tables

List of abbreviations

AI: Artificial Intelligence

APT: Advanced Persistent Threat

BPA: Basic Probability Assignment

BPAF: Basic Probability Assignment Function

C2: Command and Control server

CIDS: Cloud Intrusion Detection System

CIIP: Critical Information Infrastructure Protection

CNSMS: Collaborative Network Security Management System

CPNI: Centre for Protection of National Infrastructure

CPU: Central Processing Unit

CSP: Cloud Service Provider

CUSUM: Cumulative Sum Control Chart

DDoS: Distributed Denial of Service

DES: Discrete Event Simulator

DoS: Denial of Service

D-S: Dempster Shafer

eDOS: Economic Denial of Service

ENISA: European Network and Information Security Agency

EWMA: Exponentially Weighted Moving Average

FIS: Fuzzy Inference Systems

FN: False Negative

FP: False Positive

HMI: Human Machine Interface

IaaS: Infrastructure as a Service

ICS-CERT:	Industrial Control Systems Cyber Emergency Response Team
ICT:	Information Communication Technology
IDPS:	Intrusion Detection / Prevention System
IDS:	Intrusion Detection System
IED:	Intelligent Electronic Devices
IoT:	Internet of Things
IP:	Internet Protocol
IPS:	Intrusion Prevention System
LDDoS:	Low-rate Distributed Denial of Service
LGP:	Linear Genetic Programming
MARS:	Multivariate Adaptive Regression Splines
MN:	Monitoring Node
NIST:	National Institute of Standards and Technology
NMA:	Node Monitor Agent
NSA:	Node Supervisor Agent
OS:	Operating System
PaaS:	Platform as a Service
PLC:	Programmable Logic Controller
RAT:	Remote Access Trojan
RED:	Random Early Drop
RRED:	Robust Random Early Drop
SaaS:	Software as a Service
SCADA:	Supervisory Control and Data Acquisition
SLA:	Service Level Agreement
SN:	Super Node
SNMP:	Simple Network Management Protocol

SVM: Support Vector Machine

TCP/IP: Transmission Control Protocol / Internet Protocol

TN: True Negative

TNR: True Negative Rate

TP: True Positive

TPR: True Positive Rate

UTM: Unified Threat Management

VLAN: Virtual Local Area Network

VM: Virtual Machine

WSN: Wireless Sensor Network

Chapter 1

Introduction

Critical infrastructures are essential for the functioning of society and the economy; examples include telecommunications, energy, oil, gas, and transport. Technologies associated with operating and controlling the operations of these critical infrastructures are industrial control systems, more commonly known as SCADA (supervisory control and data acquisition) systems. In recent years, critical infrastructures have become increasingly interlinked and networked, often connected to the Internet; consequently making these systems more vulnerable and exposed to the threat of cyber-attacks [1]. In this thesis, critical infrastructures are referred to as those whose disruption could have a high socioeconomic impact.

The critical infrastructure systems that support major industries, such as manufacturing, transportation, and energy, are highly dependent on Information Communication Technology (ICT) for their command and control. Whilst a high dependence on industrial control systems, such as SCADA still exists, critical infrastructure systems are migrating to new communication technologies. As a result, common communications protocols and open architecture standards are replacing the diverse and disparate proprietary mechanics of industrial control systems — this can have both positive and negative implications [2]. Cybercrime affects society as a whole; not only threatens individuals' privacy, but it may also potentially compromise a country's critical infrastructure and its ability to provide essential services to its citizens [3]. Exploitations that can affect countries' infrastructure are usually infiltrated by simple or sophisticated tools that can access mobile and other personal devices to infiltrate high-value sectors, such as transportation, energy, or financial systems [3]. The consequences of an attack on one of these could result in loss of life, economic damage or a devastating effect on the operation of government services and military defence.

Operators of critical infrastructures, in particular the ICT that supports gas and electricity utilities and government services, are considering using the Cloud to provision their high assurance services. This is reflected in a white paper produced by the European Network and Information Security Agency (ENISA) [4] in 2013, which provides specific guidelines in this area, with particular emphasis on the technical, procedural/policy-based and legal implications. The European Commission [4] has defined Critical Information Infrastructures as ICT systems which are either, a) critical infrastructure themselves, or b) essential for the operation of other critical infrastructures. Many operators do not have the infrastructure to support the growing need for accurate predictive and historical simulations imposed by the adoption of renewable energy sources and the on-going development of smart grids. To overcome this, Cloud computing allows these operators to reduce or avoid over investment in hardware resources and their associated maintenance [5].

While this may offer improved performance and scalability, the associated security threats impede this progression. Availability issues and real world implications would be the main concern for providers of critical infrastructure, depending upon the operations or services they are hosting [6]. To address these concerns, a range of security measures must be put in place, such as intrusion detection and prevention techniques, cryptographic storage, and network firewalls. Several recent studies have proposed intrusion detection approaches that are tailored to the needs of industrial installations. However, at this time, the more challenging problem of integrating large scale critical infrastructure communications, intrusion detection and risk levels in a comprehensive framework for distributed intrusion detection system (IDS) design has not been addressed adequately [7].

Cloud computing is already a successful paradigm for distributed computing and is still growing in popularity. However, many problems still linger in the application of this model and some new ideas are emerging to help leverage its features even further. One of these ideas is the Cloud federation, which is a way of aggregating different Clouds to enable the sharing of resources and increase scalability and availability [8]. The federation of Cloud resources allows an enterprise to distribute workloads globally, move data between disparate networks and implement innovative security models for user access to Cloud resources. Federation across different Cloud resource avenues allows applications to run in the most appropriate infrastructure environments.

Based on the problem at hand, it is clear that the development of a Cloud-based intrusion detection protection method for hosting critical infrastructure services in the Cloud environment is required. However, the greater scalability and larger size of Clouds compared to traditional service hosting infrastructure involve more complex monitoring systems. Monitoring systems must be scalable, robust, and fast; and able to manage and verify a large number of resources and effectively and efficiently. This has to be achieved through short measurement times and fast warning systems, able to quickly identify and report performance impairments or other issues, and to ensure timely interventions such as the allocation of new resources. Therefore, monitoring systems and current IDS methods must be refined and adapted to differing situations in Cloud environments.

Within this thesis, a methodology that develops a Service-based collaborative intrusion detection framework in a federated Cloud environment is proposed. Our collaborative architecture consists of four major entities: the Cloud broker, the monitoring nodes (MNs), the local coordinators (Super Nodes - SNs), and the global coordinators (Command and Control server - C2). The C2 is located within each CSP domain and is effectively a domain management node. The C2 provides management of SNs and MNs, responses to attacks detected and reported by SNs and/or the broker, and cooperates with adjacent domains when polled which is illustrated in Figure 1-1.

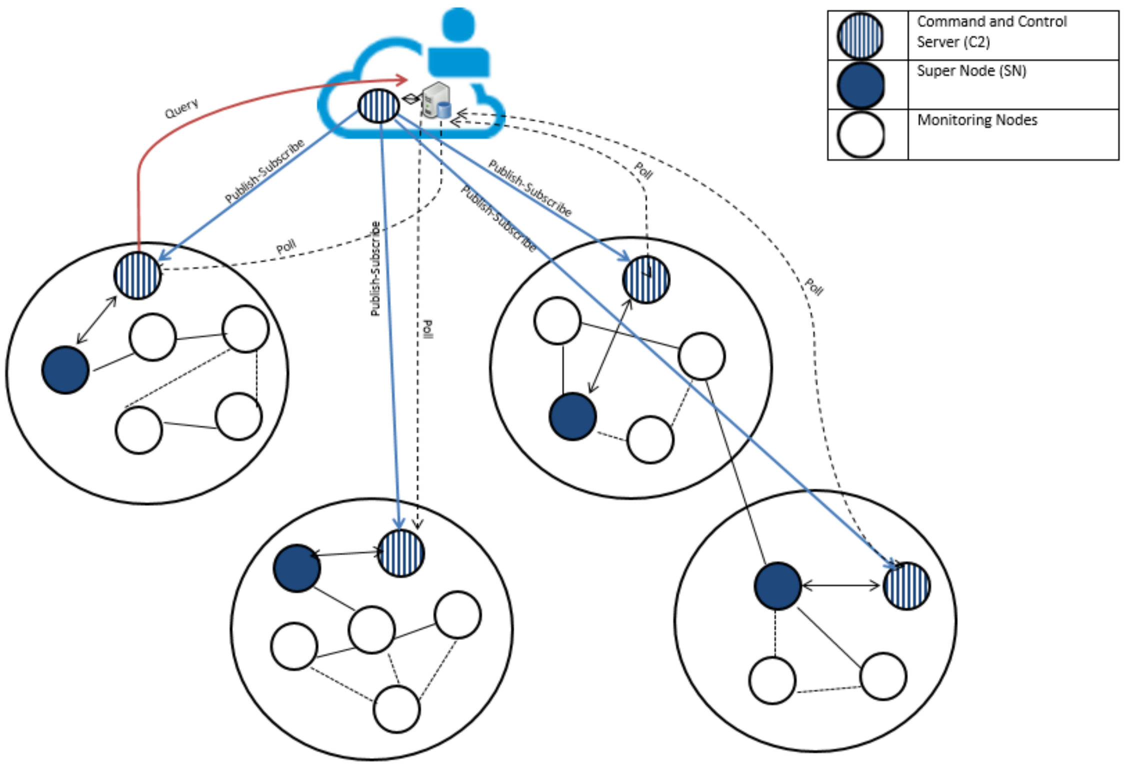


Figure 1-1: C2 role within the Cloud federation

Once a belief that an attack is underway is generated, an alert is sent to a C2. The C2 queries the broker, and if no information on the attack is possessed, a global poll procedure is taken, where C2s in adjacent domains are queried. The broker fuses the information provided by the C2s together and facilitates collaborative decision making – this in turn, could help trace the source of attack to the domain of origin. The broker coordinates attack responses, facilitating inter-domain cooperation, aiming to improve the resilience of the interconnected infrastructure.

The work that is presented in this thesis focuses on the problem of collaborative intrusion detection within a Cloud federation and autonomous sharing of information. The proposed solution looks towards improving the detection and prevention accuracy and speed in dealing with a compromised domain via collaborative decision making. The approach will not detect and prevent all intrusions but will focus on DDoS. There is no current solution that can provide adequate protection for Cloud federations, as existing solutions instead provide inadequacies which are detailed in Chapter 3.5. There is therefore a need to develop a collaborative Cloud IDS that can overcome the challenges within this area, and this thesis presents such a solution Service-based collaborative intrusion detection framework – "Security as a Service".

1.1 Motivation

Adoption of Cloud technologies allows critical infrastructure to benefit from dynamic resource allocation for managing unpredictable load peaks. Given the public awareness of critical infrastructures and their importance, the public wants to be assured that these systems are built to function in a secure manner [9]. Hence, appropriate security means have to be selected and documented accordingly when developing such systems. Most existing technologies and methodologies for developing secure applications only explore security requirements in either critical infrastructure or Cloud computing.

Individual methodologies and techniques or standards may even only support a subset of specific critical infrastructure requirements. Requirements based on security issues can be quite different for these applications and for common ICT Cloud applications but need to be considered in combination for the given context. Disruptions in one part of the infrastructure may spread out through the system and have cascading effects on other sectors [10]. Critical infrastructure protection relates to application processes, electronic systems, and information stored and processed by such systems.

The concern is that critical ICT resources and information in Cloud systems might be vulnerable to cyber-attacks or unauthorised access. The primary security concerns with Cloud environments pertain to security, availability, and performance. Many attacks are designed to block users from accessing services and providers from delivering services, i.e., Denial of Service (DoS). Service providers may face significant penalties due to their inability to deliver services to customers in accordance with regulatory requirements and Service Level Agreements (SLAs) [53].

Due to the potentially high profile effects of attacks on critical infrastructure systems, these industries have become even more attractive targets for cybercriminals [3] – and utilising Cloud computing into their paradigm increases the system complexity and vulnerability. It is imperative to develop a solution that can address the complexity of detecting attacks in a large scale interconnected infrastructure [7]. Traditional network monitoring schemes are not dynamic to cope with high speed networks such as Cloud networks, let alone Cloud federations, as conveyed in Chapter 3. A number of approaches have been proposed to resolve the problem, however they are insufficient. It is clear that an IDS alone cannot protect the Cloud environment from attack. If an IDS is deployed in each Cloud computing region, but without any cooperation and communication, it may easily suffer from a single point of failure. The Cloud environment could not support services continually, as it is not always easy for the victim to determine that it is being attacked, or where the attack is originating from.

DDoS is a serious and growing problem for corporate and government services conducting business on the Internet. Resource management to prevent DDoS attacks is receiving attention, as the Infrastructure as a Service (IaaS) architecture effectively 'supports' the attacker as when the Cloud system observes the high workload on the flooded service, it provides more computational resources in order to cope with it. Additionally, a unique type of attack is the Economic Denial of Sustainability (eDoS) [11]. This type of attack is directly connected with a DoS or DDoS attack, but has financial implications as well.

1.2 Aims and objectives

The research presented in this thesis has highlighted the security concerns for migrating critical infrastructure services to the Cloud environment. This is predominantly caused by the inadequacies and limitations of current security protection measures which fail to cope with the sheer size and vast dynamic nature of the Cloud environment. The sensitive nature of critical infrastructure services deems their protection critical, and their services additionally. Attacks and failures are inevitable; therefore, it is important to develop approaches to understand the Cloud environment under attack. Lack of collaboration among different components within a Cloud federation, or among different CSPs, for detection or prevention of attacks is a key focus of our work.

Additionally, our research focuses on maintaining the availability of the data, as previously described, the service in question could be financial, organisational, or on demand. Protecting the Cloud environment from DDoS attacks is imperative as these attacks can threaten the availability of Cloud functionalities.

To address the security challenges in this area and provide solutions for the protection of critical infrastructure services, we summarise our research aims as the following:

- Develop a Cloud-based intrusion detection method for hosting critical infrastructure services in the Cloud environment having identified the limitations within existing intrusion detection techniques.
- Create a model that can be tailored to the different Cloud environments as these may comprise similar functionality but can differ depending on the services needed.
- Focus on the availability of services and assets, making sure the high value processes continue to function regardless of what is occurring elsewhere.

In this thesis the above issues are addressed by proposing a collaborative intrusion detection methodology which offers the following features:

- Monitoring system that can run in a non-intrusive and transparent manner to any underlying virtualised infrastructure.
- Application adaptive which diminishes the need for re-contextualisation each time an application and/or resource related parameter changes.
- Generates high-level application metrics dynamically at runtime by aggregating and grouping low level metrics.

- Adopts a collaborative mitigation strategy, which focuses on:
    - Containing the attack close to the source(s)
    - Robustness to attack via a distributed mitigation architecture
    - Autonomous sharing of threat information

The key objective of this research is to produce a methodology for the protection of critical infrastructure services in the Cloud environment through collaborative intrusion detection. The project objectives will be achieved by examining further the current approaches, and evaluating and advancing the best methods and combination of approaches to be used.

A breakdown of our project objectives is provided below:

- Design of a collaborative Cloud-based framework that focuses on the monitoring and protection of the critical infrastructure services in the Cloud computing environment. This has been achieved by the development of our Security as a Service architecture for CSPs within a Cloud federation.

- Demonstrate a solution and technique that can effectively monitor Cloud domains and reliably help to secure and block threats, through efficient communication and exchanging threat information before a threat could propagate throughout the network. This approach is inspired by agent based intrusion detection but with a combination of communication algorithms and a hierarchical monitoring structure will have improvements in terms of scalability and message dissemination.

- Create a technique to analyse attack data in multiple domains and come to a decision on its importance as to whether an anomaly has occurred. This has been fulfilled with the application of the Dempster-Shafer (D-S) theory of evidence being used for belief generation by monitoring entities, and data fusion by the Cloud broker to facilitate collaborative intrusion detection.

- Develop appropriate assessment metrics to analyse if the methods proposed, or the combinations of selected methods, effectively address the issue of infrastructure service protection in the Cloud environment via collaborative intrusion detection. This has been achieved via the comparison of our extended D-S fusion process to the works of Lo et al. and a high level assessment using our design requirements in Chapter 2.6 – the comparison outcomes to highlight the merits of our solution.

1.3 Novel contributions

The research within this thesis presents the principles, techniques, protocols and algorithms adapted from other distributed computing paradigms to the development of our Service-based collaborative intrusion detection framework, providing “Security as a Service” within Cloud federations.

This thesis makes the following novel contributions to the field of intrusion detection within Cloud federation environments:

1. A collaborative intrusion detection framework that can detect and prevent intrusion in Cloud federations and/or collaborative domains in real-time via the autonomous sharing of information. This features a novel application of the D-S theory of evidence algorithm to detect intrusions and fuse generated beliefs for collaborative intrusion detection, and an extension of D-S to include confidence values for conflicting decisions. There is no current solution that can provide adequate protection for Cloud federations, or identified solution which implements the D-S algorithm and produces collaborative decisions in Cloud federations to improve upon the Cloud security contribution.

2. The Cloud broker coordinates attack responses, both within the domain itself, and with other domains, and is facilitating inter-domain cooperation. D-S is used to fuse the generated beliefs and make a system-wide decision. This cooperation between CSPs ensures that the scalable defence required against DDoS attacks is carried out in an efficient way; aiming to improve the overall resilience of the interconnected infrastructure.

3. Cumulative sum control chart (CUSUM) and Exponentially Weighted Moving Average (EWMA) algorithms are used for adaptive threshold calculation. A local propagation mechanism collects statistics at a local level via MNs, and in order to minimise

detection delay and reduce the communication overhead, this is propagated among MNs using a gossip algorithm.

4. The main novel contribution of this project is that it provides the means by which DDoS attacks are detected within a Cloud federation, so as to enable an early propagated response to block the attack, particularly by the interdependent CSPs within the Cloud federation. This is effectively inter-domain cooperation as these CSPs will cooperate with each other to offer holistic security, and add to the defence in depth. The D-S theory of evidence is used to facilitate this autonomous sharing of information, and to fuse the generated beliefs to form a system-wide decision.

1.4 Publications resulting from this research

Work presented in this thesis has been peer reviewed and published in conferences and journals throughout. The references for these publications are listed below.

Journal Papers

1. Á. MacDermott, Q. Shi, M. Merabti, and K. Kifayat, "An elastic scaling method for Cloud security," in *The Journal of Internet Technology and Secured Transactions (JITST),* Volume 3, Issues 3/4, pp. 254 – 262, 2014, ISSN 2046-3723 (Online).

2. Á. MacDermott, Q. Shi, M. Merabti, and K. Kifayat, "Hosting critical infrastructure services in the Cloud environment" in *Inderscience International Journal of Critical Infrastructures*, 2015, Volume 11, No. 4, pp. 365 – 381.

3. W. Hurst and Á. MacDermott, "Evaluating the Effects of Cascading Failures in a Network of Critical Infrastructures" in *Inderscience International Journal of System of Systems Engineering (IJSSE),* 2015, Volume 6, No. 3, pp. 221 - 236.

4. Á. MacDermott, Q. Shi, and K. Kifayat, "Collaborative Intrusion Detection in Federated Cloud Environments" in *Science and Education Publishing Journal of Computer Sciences and Applications,* 2015, Volume 3, No. 3A, pp. 10-20.

5. C. Chalmers, M. Mackay, Á. MacDermott, "Securing the smart grid: threats and remediation" in *International Journal of Smart Grid and Green Communications,* 2016, Volume 1, No. 2, pp. 166-190, ISSN: 2052-2010.

Conference Papers

6. Á. MacDermott, Q. Shi, M. Merabti, and K. Kifayat, "Intrusion Detection for Critical Infrastructure Protection," in *Proceedings of 13th Annual Postgraduate Symposium on Convergence of Telecommunications, Networking and Broadcasting (PGNet 2012),* 2012, pp. 224-229.

7. Á. MacDermott, Q. Shi, M. Merabti, and K. Kifayat, "Protecting Critical Infrastructure Services in the Cloud Environment," in *Proceedings of the 12th European Conference on Information Warfare and Security (ECIW),* 2013, pp. 336–343.

8. Á. MacDermott, Q. Shi, M. Merabti, and K. Kifayat, "Detecting Intrusions in the Cloud Environment," in *14th Annual Postgraduate Symposium on Convergence of Telecommunications, Networking and Broadcasting (PGNet 2013*), 2013, pp. 336-343.

9. Á. MacDermott, Q. Shi, M. Merabti, and K. Kifayat, "Considering an elastic scaling model for Cloud security," in *The 8th International Conference for Internet Technology and Secured Transactions (ICITST-2013),* 2013, pp. 150-155.

10. Á. MacDermott, W. Hurst, Q. Shi, and M. Merabti, "Simulating Critical Infrastructure Cascading Failure," in *IEEE UKSim 16th International Conference on Computer Modelling and Simulation (UKSim 2014),* 2014, pp. 324-329.

11. Á. MacDermott, Q. Shi, M. Merabti, and K. Kifayat, "Security as a Service for a Cloud Federation," in *15th Annual Postgraduate Symposium on Convergence of Telecommunications, Networking and Broadcasting (PGNet 2014),* 2014, pp. 77-82.

12. Á. MacDermott, Q. Shi, and K. Kifayat, "Collaborative intrusion detection in a federated Cloud environment using the Dempster-Shafer theory of evidence", in *14th European Conference on Cyber Warfare and Security (ECCWS),* 2015, pp. 195-203.

13. Á. MacDermott, Q. Shi, and K. Kifayat "Detecting Intrusions in Federated Cloud Environments Using Security as a Service", in *IEEE 8th International Conference on Developments in eSystems Engineering (DeSE 2015),* 2015, pp. 91-96.

14. Á. MacDermott, Q. Shi, and K. Kifayat, "Distributed Attack Prevention using Dempster-Shafer Theory of Evidence", in *Huang DS., Hussain A., Han K., Gromiha*

M. (eds) Intelligent Computing Methodologies. ICIC 2017. Lecture Notes in Computer Science, vol 10363. Springer, Cham, pp. 203-212.

Posters

Á. MacDermott, Q. Shi, M. Merabti, and K. Kifayat, "Cloud Federation & Security" in *15th Annual Postgraduate Symposium on Convergence of Telecommunications, Networking and* Broadcasting (PGNet 2014), June 2014.

Á. MacDermott, and Q. Shi, "*Collaborative Intrusion Detection in Federated Cloud Environments*", Cutting Edge 2017 Conference, April 2017.

1.5 Thesis structure

The remainder of the thesis is arranged into six subsequent chapters; the order and contents of these chapters are as follows:

Chapter 2 – "Background", with contextual material on critical infrastructure and Cloud computing. Each of the topics is detailed and then focuses specifically on its evident utilisation, with particular focus on legal considerations, protection problems, and general issues that arise due to its advancement. The Cloud computing paradigm has security vulnerabilities and threats that need to be resolved, and through a Cloud-based IDS and collaborative intrusion detection framework.

Chapter 3 – "Related Work", an overview of the different types of protection and preventative measures in place for intrusion detection in the Cloud environment is presented. An in-depth critical review of related work and existing approaches within this area is undertaken, and an observation of the evident inadequacies are conveyed. These help identify the essential characteristics for our collaborative intrusion detection framework.

Chapter 4 - "Service-based collaborative intrusion detection framework" details the design and architectural details for the novel solution. The entities within the architecture and their functionality are conveyed, in addition to the algorithms and methods utilised. Our extended D-S theory of evidence fusion process is outlined, in addition to requirements and considerations for the implementation.

Chapter 5 – “System Implementation”, the implementation details of our intrusion detection approach for federated Cloud environments are highlighted. We begin with an overview of the architecture of the implementation and a discussion of its core components. Through the use of Riverbed Modeler 18.0 [12], the effects of DDoS attacks on Cloud federations are conveyed and analysed, and collaborative intrusion detection using the D-S theory of evidence is demonstrated using a proof of concept developed in C#.

Chapter 6 – “Evaluation”, we evaluate our novel solution against the metrics we consider to be the most important for determining the efficiency of a collaborative IDS. The solution is evaluated using the requirements established in 3.8 which define the characteristics which collaborative intrusion detection solutions must possess. The approach is also compared to related work within this research field.

Chapter 7 – “Conclusions and future work”, we provide a thesis summary, present a summary of our novelty, and express some concluding remarks. This chapter will also highlight how the work can be built on for future research projects.

1.6 Summary

This chapter has presented an overview of this thesis, outlining research motivation, aims and objectives, and highlighting the novel contributions. The increasing integration of critical infrastructure services and sensitive data to the ICT paradigm offers great concerns for the protection of these services and information. The more interconnected and available they become, the more threats they are opening themselves up to. Traditional intrusion detection mechanisms are not sufficient for protecting infrastructure services in the Cloud environment, as many solutions do not have the economy of scale and are inefficient at processing data of such a volume.

This chapter has provided further context on this emerging situation and identified a new and novel solution to the problem: our Service-based collaborative intrusion detection framework. In Chapter 2, background information to this problem is provided, focusing on the areas of critical infrastructure, Cloud computing, and their evident utilisation. Research challenges and requirements for collaborative intrusion detection will be identified and explored in the next chapter, and details of how the proposed solution can meet these will be conveyed.

Chapter 2

Background

Critical infrastructures, such as the power grid and water distribution facilities, include a high number of devices over a large geographical area. These infrastructures face a significant increase in threats targeting their operations due to the growth in their use of industrial control systems and their integration to networks. While this provides great capabilities for operation, control, and business, this augmented interconnectedness also increases the security risks due to the cyber related vulnerabilities they possess. The importance of protecting these infrastructures has been particularly highlighted by the increase in malware designed to target these systems and disrupt their functionality; examples include Stuxnet [13], Duqu [14] and Flame [7]. Effective protection of critical infrastructures is crucial as it is apparent that existing methods do not meet the security requirements of such interconnected infrastructures.

Infrastructures will inevitably fully grasp the benefits being offered by Cloud computing. Once their services are in the Cloud environment, resourceful functionality is essential. There needs to be an assurance that the Cloud computing environment can provide proficient protection of the sensitive critical infrastructure data. The reality of today's advanced malware and targeted attacks is that 100% protection is not realistic. Reducing attack vectors and marginalising the impact of an attack is a realistic approach. In this chapter background on Critical Infrastructures and their protection problem is provided, and how Cloud computing can provide notable advancements and benefits but can expose them to more attack vectors and vulnerabilities [15].

2.1 Critical infrastructure

The Centre for the Protection of National Infrastructure (CPNI) defines critical infrastructure as "*Those critical elements of national infrastructure (facilities, systems, sites, property, information, people, networks and processes), the loss or compromise of which would result in major detrimental impact on the availability, delivery or integrity of essential services, leading to severe economic or social consequences or to loss of life*" [16]. CPNI is focussed on providing advice and assistance to those who have responsibility for protecting these most crucial elements of the UK's national infrastructure from national security threats. These infrastructures have historically been physically and logically separated systems that had little interdependence [17]. Nowadays modern critical infrastructures largely make use of technologies where wireless sensor networks (WSNs) with open access have become an integral part of virtually any critical infrastructure. Due to advances in ICT and efforts to improve efficiency in these systems, these infrastructures have become increasingly automated and interlinked.

Critical infrastructures are controlled by networked computers and can be defined as sectors that would have a debilitating impact on national security if incapacitated [18]. Failures can result in devastating impact on critical sectors, such as national defence, the economy, communication, e-government systems, and society as a whole. Natural phenomena, system errors, or cyber-attacks have the ability to produce failures, which cascade as a result of the high level in interconnectivity between the infrastructures [19]. The resulting impact would cover large sectors causing devastating consequences. Critical infrastructure protection has now become a multi-disciplinary area, which requires interdisciplinary involvement.

Concern in the industry is how many of these infrastructures are dependent upon each other for functioning [20]. Interdependencies among computers, communication, and power infrastructures have increased security risks due to the complexity of the integrated infrastructures. Critical infrastructure control systems may vary depending on the different environments and infrastructure; additionally, they are created with different design goals compared to traditional systems. Securing control systems is difficult as the usual security assumptions and practices that are applied for protecting ICT systems are not sufficient. Each point in the SCADA network is a potential entry point. This is in part due to the fact that little work has been done to enhance their security because of the belief that they had no problem [21]. The connectivity of SCADA networks increases the risk of cyber-attacks and the critical need to improve the security of these networks. SCADA systems are rarely patched or updated as engineers are often hesitant to do so due to concerns that the patch itself could potentially negatively affect the operation of the system.

Critical infrastructure protection methods and resources deter or mitigate attacks and focus on protecting those assets considered invaluable to society. Security experts around the globe are now recognising the importance of effective simulation in planning the fight against the growing cyber-threat. The consequences of failure can produce unexpected results and must be planned for in order to prevent disasters escalating through a cascading effect [22].

2.1.1 Cloud computing and critical infrastructure utilisation

Due to virtually unlimited scalability of resources and performance, as well as significant improvement regarding maintainability, it is inevitable that Cloud computing will eventually reach the ICT services that are operating critical infrastructures [23]. These services have very high requirements towards trustworthiness, i.e., dependability, security, and performance, which cannot be provided by using the available off-the-shelf offerings [24]. Cloud computing proposes that, one day, all levels would become virtualised, i.e. “everything-as-a-service.” Critical infrastructure currently makes use of the benefits offered by general ICT services, so benefiting from the intricate Cloud computing paradigm is expected. Critical infrastructures provide essential utilities like water supply, electricity, or transportation. Such infrastructures need to cope with variable usage, high flexibility, and failovers to work properly.

Modern IP-based critical infrastructure control systems allow more efficient control than traditional systems. The variable workload, unpredictable usage spikes and outsourcing of data handling, make the Cloud an interesting alternative for critical infrastructure ICT due to the unlimited resource capabilities, ability to scale considerably, and storage advancements, i.e. resources, outsourcing, backups, etc. Another advantage of using the Cloud in this context is to aggregate data from the IP enabled control devices, which have limited resources and cannot process data locally. This means that sooner or later infrastructure providers will use Cloud applications for their systems and hence related issues need to be investigated [25] [26].

With the technical development and market growth in Cloud computing, organisations that provide, operate and maintain ICT systems for critical infrastructure are making the decision as to when they should make the paradigm shift. Cloud services can offer efficient access to large ICT infrastructures that benefit from the economy of scale. Therefore, it would be highly desirable to maintain irrecoverable and valuable data obtained from critical infrastructure within secure Cloud infrastructures. The current issue is that existing security mechanisms, including SLAs provided by current Cloud offerings, lack transparency. They cannot be adopted, negotiated, or verified against institutional policies [23]. This issue has led to hesitation from organisations that maintain sensitive data from pledging fully to the Cloud Computing movement, thus disabling them from utilising Cloud services and critical infrastructure fully.

The increasing flexibility and unpredictable usage of such utilities often means that many challenges that can occur in the utility networks used. The usage of modern ICT systems to control and manage critical infrastructure helps in dealing with such issues [9]. Many operators do not have the infrastructure to support the growing need for accurate predictive and historical simulations imposed by the adoption of renewable energy sources and the on-going development of smart grids. Cloud computing allows these operators to reduce or avoid over investment in hardware resources and their associated maintenance. Infrastructure vendors will inevitably take advantage of the benefits Cloud computing has to offer [27].

2.1.2 Use case

Most industrial plants employ networked process historian servers storing process data and other possible business and process interfaces. For example, direct file transfer from programmable logic controllers (PLCs) to spreadsheets [28]. PLCs in the control system generate a huge amount of data and logs. Logs of communication are stored in the historian databases. These databases possess historical data that is being logged 24/7 from over 6,700 data points so that it could be easily accessible by both operators and engineers [29], [30]. These historian servers receive data from field control processors or front-end processors, which issue control commands to and poll data from devices in field networks. The control network typically contains assets such as the human-machine interface (HMI) and other workstations, which run control system applications on conventional computer platforms. The field network devices directly monitor and control a physical process, such as refining, manufacturing, or electric power generation/transmission/distribution [31].

One way for critical infrastructure to utilise the Cloud environment would be for the historian database to send these historical processes to a private Cloud. The use of a private Cloud to audit the data from the system and process it more effectively would be valuable. This would overcome the challenges associated with processing vast data sets generated by the control systems. The Cloud environment is suitable as it has massive storage and computational capabilities, is distributed and elastic, offering improved processing rates and efficiency compared to current methods. Private Clouds grant complete control over how data is managed and what security measures are in place.

This is two-folds though. This collection of data could be used to perform behavioural analysis and modelling of this information flow – looking for trends and subtle changes in the data would be beneficial in achieving state awareness. Behaviour modelling can take place without affecting the system in any way. By monitoring the evolution of the plant process states, and tracking down when the industrial process is entering into a critical state, it would be possible to detect these attack patterns (known or unknown) aiming at putting the process system into a known critical state by using state of commands. In control system architectures, the major cyber-attack vector is the flow of network commands [32].

By processing the sensor data from the historian in a private cloud environment, the behaviour of the infrastructure and use the critical state metric as a trigger for logging a chain of packets can be analysed. It is possible to discriminate between critical states due to cyber-attacks and critical states due to faults/physical attacks. A reference behaviour model could be implemented with the use of Bayesian classification procedure associated to unsupervised learning algorithms to evaluate the deviation between current and reference behaviour [33]. By training with unsupervised learning algorithms, the log analysers can discover the inherent nature of the dataset by clustering similar instances into classes.

A further advantage of using the Cloud in this context is to aggregate data from the IP-enabled control devices, which have limited resources and cannot process data locally. Though there are benefits of this connection, one of the main concerns with utilising critical infrastructure services in the cloud environment is the threat of attack. Deploying high assurance services in the Cloud increases cyber security concerns, as successful attacks could lead to outages of key services, and/or disclosure of sensitive personal information.

Adoption of Cloud computing services allows critical infrastructure vendors to benefit from dynamic resource allocation for managing unpredictable load peaks, storing of historical process data (either on site in a private Cloud, or sharing among other related vendors in a hybrid Cloud), federating into a larger Cloud, and large scale data analytics based on historical data of consumers, to name a few.

2.1.3 Requirements for critical infrastructure

There are many benefits and limitations associated with the utilisation of Cloud computing by critical infrastructure operators. In Table 2.1 some of these attributes are identified and considerations for each point are made [34].

Table 2.1: Cloud computing benefits and risks for critical infrastructure

Cloud computing attribute	Benefits	Limitations
Agility	Ability to adapt to resource intensive tasks, and scale up/down with ease.	Lack of efficiency in ability to scale to match demand; and associated latency costs.
Device and location independence	Location and geographical independence.	Consistency of data in terms of connectivity, latency and performance issues.
Real time response and elastic performance	Quick response to fluctuations in service demand and distribution.	Consistency of data with regards to connectivity, latency and associated performance issues.
Self-healing	Enhance the robustness and resilience of critical infrastructure and interdependent systems.	Self-repair may lead to system inefficiencies or data accuracy as underlying issues may remain unsolved.
Virtualisation and automation of services	Faster response time, disaster recovery/planning, deployment of security implementations.	Data security due to hypervisor and VM vulnerabilities/misconfigurations/ exploitation.

Requirements in critical infrastructure regarding overall redundancy, data availability, authenticity, secure access, and low latency network connectivity are typically higher than in commercial applications. Critical infrastructure imposes much stronger requirements for security, reliability, and resilience on Cloud computing environments. There are also more stringent considerations for control of data, data-centric security requirements, protection of data, and legal issues surrounding location of data.

Control

An SLA defines how the consumer will use the services and how the provider will deliver them. Responsibilities of each party and remedies should be included. The SLA cannot be adopted or negotiated, which often deters organisations from pledging fully to the Cloud. Control is a common challenge as depending on which deployment model is chosen, control is not always in the hand of the owner. Private Clouds allow organisations to shape how their data is stored and controlled, and what security measures are in place.

Data-centric security approach

In general, a data-centric security approach must ensure that data protection mechanisms are deployed across all provided security solutions and that data owners have the full control over who has the right to use the data and what they are authorised to do with it. In addition, institutional security policies and access rules can be specified and mapped to the Cloud environment. Requirements-based security issues can be quite different for critical infrastructure applications and for common ICT applications but need to be considered in combination for the given context.

Protection

Since Cloud computing supports a distributed service oriented paradigm, multi-domain and multi-users administrative infrastructure, it is more prone to security threats and vulnerabilities, such as data breaches, data loss, service hijacking, DDoS attacks, and malicious insiders [35]. Tailored ID/IP mechanisms are essential. Compared to other systems and services in the Cloud environment, a critical infrastructure requires a much higher level of assurance. One of the risks in a multi-tenant environment is over provisioning of resources. Over provisioning of resources results in resource contention and potential lack of availability, effectively creating a DoS situation [36], and impacting on users of the Cloud service who depend on its continuity.

Legal issues

Legal requirements include data protection and regulatory requirements. Issues also surround data being exchanged across multiple countries that have different laws and regulations concerning data traversal, protection requirements, and privacy laws. Examples of such risks include, but are not limited to: risks resulting from possible changes of jurisdiction and the liability or obligation of the vendor in case of loss of data and/or business interruption [37]. There are also geographical requirements for healthcare data being stored.

2.2 Cloud computing

Cloud computing has emerged as a way to enable content providers to meet their application needs through either Cloud development environments or through outsourced CSPs. There are a variety of business models for Cloud infrastructure but they generally vary depending upon the business requirements. The infrastructure is both physical in the form of processing, as well as software that meets the needs of the user. An advantage of Cloud computing is that the enterprises using the services do not need heavy investment in capital equipment. Software services are provided by a vendor and paid for by a subscription or on the basis of the amount of use. The National Institute of Standards and Technology (NIST) defines five essential characteristics of Cloud computing [38]:

- **On-demand self-service:** A user should be able to acquire or release resources without requiring outside human interference.
- **Broad network access:** Resources should be available over the network.
- **Resource pooling:** Resources are pooled to serve disparate customers on the same or different physical machines. Resources can by dynamically assigned according to customer demands; examples of resources include storage, processing, memory, network bandwidth, and virtual machines (VMs).
- **Rapid elasticity:** Users can acquire, release, and scale resources in an elastic manner, making the available resources appear unlimited from the client’s point of view.
- **Measured service:** The Cloud management layer constantly monitors, controls, and reports resource use to both the provider and client, providing a metering capability.

CSPs usually build up large scale data centres and provide Cloud users with computational resources in three delivery models, distinguished by their level of resource abstraction [27] which are Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS).

- SaaS – CSP hosts applications and makes them available to the user over the Internet.
- PaaS – CSP provides a platform for customers to develop, manage, and run applications without the need to own the physical infrastructure.
- IaaS – CSP provide virtualised computing resources over the Internet.

Security is a major concern in Cloud adoption; the focus on cybercrime at a global level has led to “as-a-service” models for illegal activity. The cybercrime market now affords potential criminals with a multitude of services, which means that deep technical expertise is not a prerequisite [39]. Critical security issues include data integrity, user confidentiality, and trust among providers, individual users, and user groups. Additionally, availability issues and real world impact would be the main concern for providers of critical infrastructure, depending upon the operations or services they are hosting [7], [40]. There are security issues at each level of the Cloud computing paradigm; the application level, virtual level, and physical level.

The application level comprises SaaS, in which enterprises host and operate their applications over the internet so the customers can access them. One benefit of this model is customers do not need to buy any software licences or any additional equipment for hosting the application(s). The virtual level includes PaaS and IaaS. PaaS provides a platform for building and running customer applications. Enterprises can build applications without installing any tools on their local systems and can deploy them with relative ease. IaaS provides a convenient option for organisations by migrating the ICT infrastructure to the Cloud provider. This means it is the responsibility of the CSP to tackle the issues of ICT infrastructure management, such as configuring servers, routers, firewalls, to name a few. The physical level refers to the infrastructure upon which Clouds are deployed.

Cloud deployment models include public, private, community, and hybrid:

- **A public Cloud** is available to the general public or large industry group, owned by an organisation selling Cloud services. A third party provides infrastructure, platform and software. The management, operational, and security requirements are provisioned and shared between users and providers with an SLA [41].

- **A private Cloud** operates for a single organisation [41]. The infrastructure can be located in the organisational unit or in a third party unit's data centre. Private Clouds grant complete control over how data is managed and what security measures are in place. There are two types of private Cloud:
    - On-premise: A Cloud integrated into an organisation’s ICT process. These Clouds are better suited for organisations which desire greater configurability and control over their data infrastructure.
    - External private Cloud: A Cloud platform that is hosted by an external Cloud provider, but with the guarantee of privacy.
- **A community Cloud** is shared by several organisations, supporting a specific community. The infrastructure is placed in more than one organisation in the community or third party's data centre. Management and operational tasks are split between the data centre owner, organisations and third party [42].
- **A hybrid Cloud** is a combination of more than one Cloud deployment model, as previously described. All the infrastructure, platform and software are portable and can switch between the deployment models in the hybrid architecture [43].

The adoption of this innovative architecture may introduce a number of additional threats that vendors may not have considered. While there are many benefits to migrating to the Cloud computing paradigm, there are security requirements and threats associated with each of its service levels. Table 2.2 conveys these based on a survey of the literature performed in [41], detailing the type of service user, requirements for this service and highlights examples of associated threats.

Table 2.2: Cloud computing security threats per service level

Level	Service level	User	Security requirements	Vulnerabilities
Application level	Software as a Service (SaaS)	End client applies to a person(s) or organisation(s) who subscribes to a service offered by a Cloud provider and is accountable for its use.	• Access control • Application security • Communication protection • Data protection from exposure (remnants) • Privacy in multitenant environment • Software security • Service availability	• Data interruption • DDoS • Disrupting communications • Exposure in network • Impersonation • Interception • Modification of data at rest and in transit • Privacy breach • Programming flaws • Session hijacking • Software modification • Software interruption • Traffic flow analysis
Virtual level	Platform as a Service (PaaS) Infrastructure as a Service (IaaS)	Developer: Moderator applies to a person(s) or organisation(s) that deploys software on a Cloud infrastructure.	• Access control • Application security • Communication security • Cloud management control security • Data security (Data in transit, data at rest, data remanence) • Secure images • Virtual Cloud protection	• Connection flooding • DDoS • Defacement • Disrupting communications • Exposure in network • Impersonation • Programming flaws • Software interruption • Session hijacking • Traffic flow analysis

Physical level	Physical data centre	Owner applies to a person(s) or organisation(s) that owns the infrastructure upon which Clouds are deployed.	• Hardware security • Hardware reliability • Legal not abusive use of Cloud computing • Network protection • Network resources protection	• Connection flooding • DDoS • Exposure in network • Hardware interruption • Hardware theft • Hardware modification • Misuse of infrastructure • Natural disasters • Network attacks

Based on the problem at hand, it is evident that sufficient security metrics need to be developed for protecting the sensitive data being stored in the Cloud environment. The ability to clearly identify, authenticate, authorise, and monitor who or what is accessing the assets of an organisation is essential for protecting an information system from threats and vulnerabilities. The distributed and open structure of Cloud computing and services becomes an attractive target for potential cyber-attacks by intruders. In the Cloud environment, where massive amounts of data are generated due to high network access rates, an IDS must be robust against noise data and false positives. Since Cloud infrastructure has enormous network traffic, traditional IDSs are not efficient to handle such a large data flow. Due to the large data sets, classification techniques require a huge amount of memory and central processing unit (CPU) usage.

2.3 Cloud federations

Cloud computing hides resource availability issues, making this infrastructure appealing to users with varying computational requirements: from storage applications to processing intensive tasks. Large-scale parallel simulations often require computing time on high performance computing machines and clusters. In a Cloud environment, resources are often shared among multiple users, and the number and nature of the workload presented by these users can vary over time. A Cloud federation is an association among different CSPs with the goal of sharing resources and data [8].

In order to cope with the resource capacity limits of a single CSP, the concept of federating multiple heterogeneous organisations is receiving increasing attention. The effects of an attack on such an infrastructure can span from the loss of some data, to the potential isolation of parts of the federation [41]. Protecting the federated Cloud against cyber-attacks is a key concern, since there are potentially significant economic consequences [44].

Federated Clouds are a logical evolution of the centralised approach, they can enhance reliability through physical partitioning of the resource pool and also to address communication latency issues by binding clients to the nearest datacentre [45]. Furthermore, federated Clouds are an interesting alternative for those companies who are reluctant to move their data out of house to a service provider due to security and confidentiality concerns. By operating on geographically distributed data centres, companies could still benefit from the advantages of Cloud computing by running smaller Clouds in-house, and federating them into a larger Cloud.

A Cloud federation allows final users to transparently access resources and services, distributed among several independent CSPs [52]. However, in the Cloud computing context there are a lot of different interpretations for this idea, e.g. it is completely different if access must be given for federated resources to a final user, as opposed to a Cloud application developer as there are many actors involved in the federation.

The work of Rak el al. [46] identifies the following actors as key players in this scenario:

- Final Users: common users which access the Cloud and uses the Cloud services.
- Service Providers: acquire resources and services from the Cloud in a transparent way, and offer them to Final Users.
- Service Developers: develop applications using the Cloud's resources. Sometimes they also use services developed by other parties.
- CSPs: Offer Cloud resources and services.

Cloud federations introduce new avenues of research into brokering policies such as those techniques based on ensuring the required quality of service level or those aiming at optimising energy efficiency [47]. The goals of brokering methods and policies in federated Clouds can be found in different domains but the key goals are listed as follows [47]:

- Cost-effectiveness: federated Clouds provide a larger amount of resources, which may help improve cost-effectiveness. These include improvement for both the user and the provider such as, for a given cost, reducing the time to completion, increasing the system throughput or optimising the resource utilisation.

- Acceleration: Cloud resources can be used to exploit an additional level of parallelism by offloading appropriate tasks to Cloud resources.

- Conservation: federated Clouds can be used to conserve allocations, within the appropriate runtime and budget constraints.

- Resilience: federated Clouds can be used to handle unexpected situations such as an unanticipated downtime, inadequate allocations, or failures of working nodes. Additional Cloud resources can be requested to ease the impact of the unexpected situations.

- Energy efficiency: federated Clouds can facilitate optimising the energy efficiency of Clouds as multiple objectives can be combined as needed. An example is combining an acceleration objective with a resilience objective.

By providing security services from within the CSP infrastructure, enterprises are able to deploy security policies and rules between each VM or between VM centres. A feature of the CSP infrastructure is that enterprises can maintain corporate security policies and the data collected about them with the VMs. This allows them to enforce security services in the enterprise and the Cloud provider consistently [48].

Many issues need to be addressed in federated Cloud resource management with respect to adaptability, flexibility, and standardisation. Performance metrics, such as bandwidth overhead, security, and quality of experience, have to be considered while designing a resource management or monitoring scheme for such environments. Intelligent computational and cognitive software agents may provide flexible and adaptable services. Human reasoning can be embedded in agents by using cognitive models and may provide better performance metric values than traditional classical approaches [49].

Each interface can present specific vulnerabilities that can be exploited by malicious entities to perform cyber-attacks:

- Cloud users,
- Service instances,
- CSPs.

Figure 2-1 illustrates these attack interfaces within a Cloud federation.

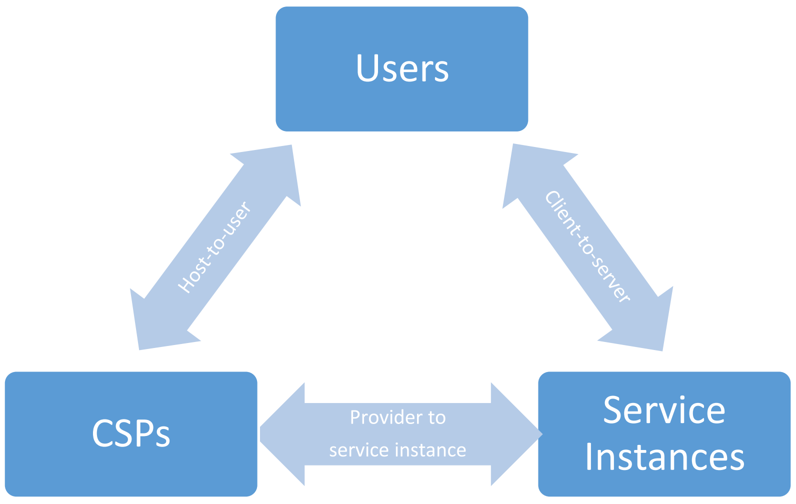


Figure 2-1: Attack interfaces in a Cloud federation

The interface between a service instance and a user can be considered as a client-to-server interface, that is vulnerable to all types of attacks that are possible in common client-server architectures, including SQL injection, buffer overflow, privilege escalation, SSL certificate spoofing, phishing attacks, and flooding attacks [50]. The interface between a service instance and a CSP is vulnerable to all attacks that a service instance can run against its hosting Cloud systems, such as DDoS, and Cloud malware injection. In the same way, a malicious CSP of the Cloud Federation may perform several attacks towards service instances running on it.

DDoS is a serious and growing problem for corporate and government services doing business on the Internet [51]. Targets for DDoS attacks include the computational resources, the memory buffers, the application processing logic, the communications bandwidth, and the network protocol, whereas their effects on the target system are the denial or degradation of provided services [46] [52]. Resource management to prevent DDoS attacks is receiving attention, as in these Cloud federations the IaaS architecture effectively supports the attacker as when the Cloud system observes the high workload on the flooded service, it provides more computational resources in order to cope with it. DDoS attacks are performed with the intention of interrupting or suspending the communication capability of any networked device or service by saturating the memory or bandwidth of the target device [34].

Haggerty et al. [53] define a DDoS attack as:

x distinct packets matching s signatures in y seconds to h host.

They also convey that it is the mass of all packets directed at a victim that poses the threat, rather than the packets themselves. For example, if resource management is not in place, a compromised VM could allow an attacker to starve all of the other VMs within that Cloud of their needed resources. By using resource management, a compromised VM can only affect itself and none of the other VMs within the Cloud [54]. If the Cloud system notices the lack of availability, it could move the affected service instances to other servers of the Cloud federation. This results in additional workload for such servers, and thus the flooding attack can propagate and spread throughout the whole Cloud federation [46], [47].

The effects of attacks can span from the loss of some data to the potential isolation of parts of the federation. This could pose a substantial risk which is a great concern for critical infrastructure vendors. Protecting the federated Cloud against cyber-attacks is a key concern, since there are potential significant economic consequences. It is clear that Cloud federations, and the CSPs present, will benefit significantly if there is a comprehensive IDS that evolves based on their requirements. The security of applications and services provided in the Cloud, against cyber-attacks, is hard to achieve for the complexity, heterogeneity, and dynamic nature of such systems [44]. The collaboration of threat knowledge about both known and unknown threats among collaborative peers within the enterprise network or with other Cloud services providers will contribute to better incident detection and prevention. This will in turn enhance Cloud security providing faster and more effective incident response.

Cross-domain data leakage of applications and the internal configurations of a Cloud member need to be addressed to enable third-party monitoring and unified monitoring of federated environments. Currently no monitoring data of a single or federated Cloud environment is publicly available, and no workload traces of the monitoring solutions themselves exist to analyse the data by statistical tools to acquire more insight into the monitoring process. An autonomous monitoring tool for validation and performance measuring of heterogeneous application sets deployed in a federated Cloud environment is required [49].

2.4 Summary

In this chapter, details on critical infrastructure and Cloud computing have been presented, in addition to their evident security challenges. Important to note for this utilisation is overall redundancy, data availability, authenticity, secure access, and low latency network connectivity are typically higher than in commercial applications. Critical infrastructure imposes much stronger requirements for security, reliability, and resilience on Cloud computing environments. Issues also surround data being exchanged across multiple countries that have different laws and regulations concerning data traversal, protection requirements, and privacy laws. Examples of such risks include, but are not limited to, risks resulting from possible changes of jurisdiction and the liability or obligation of the vendor in case of loss of data and/or business interruption.

As evident, the joining of Cloud computing and critical infrastructure will provide many benefits in the form of scalability, improved performance, reachability, and will be cost effective for organisations and infrastructure vendors. However, the distributed and open structure of Cloud computing and services becomes an attractive target for potential cyber-attacks. Despite security issues slowing its adoption, Cloud computing has become a persistent force; thus, security mechanisms to ensure its secure adoption are an immediate need. In this scenario, conventional issues become even more sensitive and critical when dealt with in the Cloud environment. For example, data security becomes more critical and difficult to deal with because of the absence of administrative control of the data owner [55].

The complexity and scale of critical infrastructures, their strong security requirements and increasing costs require comprehensive methodologies for provisioning cost effective distributed IDSs. In the next chapter, a survey of existing work on detecting intrusions in the

Cloud environment is detailed, and existing approaches for protecting the Cloud environment, and for detecting intrusions are compared. Challenges in this area will be highlighted, and from this, an observation of the area and requirements will be apparent.

Chapter 3

Related work

This chapter focuses on critically analysing existing work and solutions, by outlining their merits and highlighting their shortcomings. It aims to provide evidence to emphasise the inadequacies of existing approaches and therefore justify the motivation behind this research. We provide an overview of the existing frameworks, techniques, and approaches for detecting intrusions in Cloud computing environments in Chapter 3.1. We assess the challenges addressed by existing work and identify the limitations of the various proposed solutions throughout. Based on the aims and objectives of the research, this exploration helps recognise the research challenges as detailed in Chapter 3.4, and the identification of requirements for collaborative intrusion detection in Cloud environments in Chapter 3.5.

3.1 Intrusion detection and intrusion prevention

The aim of this research is to create a Cloud-based solution capable of facilitating collaborative intrusion detection and autonomous sharing of information within a Cloud federation. In order to facilitate this, it is necessary to understand what techniques currently exist in similar research areas and to determine their inadequacies when applied to a Cloud federation. This Chapter will review and analyse intrusion detection techniques from various research domains within computing.

Intrusion detection is defined as the process of monitoring the events occurring in a computer system or network and analysing them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies, acceptable use policies or standard security practices [56]. Figure 3-1 conveys a high level view of different attributes associated within the area of intrusion detection, and these will be explored in the subsequent subsections.

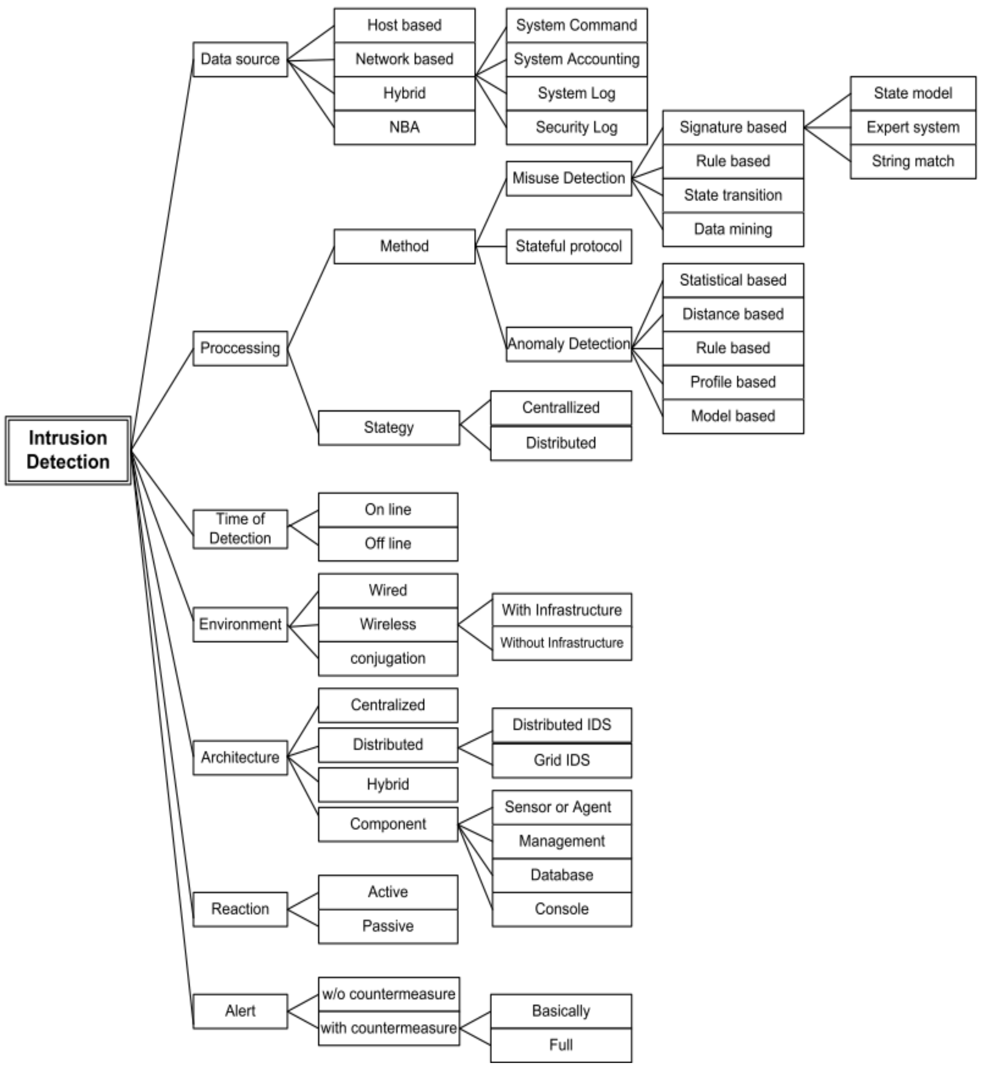


Figure 3-1: Intrusion detection and associated attributes

Computers control national infrastructure components such as the water grid and power grid. The integrity and availability of all these systems have to be protected against an increasing number of threats. The field of ICT security has become vitally important to the safety and economic well-being of society as a whole. Moreover, to expose privacy breaches, security needs powerful intrusion detection/prevention systems (IDPSs) [56] and organisational policies and procedures in place.

An IDS is a device or software application that monitors a network and/or information system for malicious activities or policy violations. They respond to suspicious activity by warning the system administrator, displaying an alert, and logging the event. An IDS can be defined as a function that maps the data input into a normal or an attack event *X* either by means of absence of an alert (0) or by the presence of an alert (1) respectively and is given by:

$$\text{IDS} : X \rightarrow \{0, 1\}. \tag{3.1}$$

To detect attacks in the incoming traffic, the IDS is typically parameterised by a threshold T. It uses a theoretical basis for deciding the thresholds for analysing the network traffic to detect intrusions. Changing this threshold allows the change in performance of the IDS. If the threshold is very low, then the IDS tends to be very aggressive in detecting the traffic for intrusions. However, there is a potentially greater chance for the detections to be irrelevant which results in large false alarms. A large value assigned to the threshold on the other hand will have an opposite effect; as being conservative in detecting attacks may lead to some being missed [57]. The most common IDS shortcomings include their low detection efficiency, high false positive rate and their vulnerability to attack based on centralised hierarchical structure (depending on their configuration) [58].

An intrusion prevention system (IPS) operates the process of performing intrusion detection and attempting to prevent the detected incidents. An IPS is a device or software application that has all the capabilities of an IDS and can also attempt to stop certain incidents. IPSs provide security at all system levels, from the operating system (OS) kernel to network data packets. IPSs also have the ability to prevent known intrusion signatures, besides the unknown attacks originating from the database of generic attack behaviours. IDSs and IPSs typically perform extensive logging of data that is related to detected events which can be used to confirm the validity of alerts, investigate incidents, and correlate events between the IDS and other logging sources [59].

3.1.1 Data source

Differentiating IDSs based on their data sources can be classified as host-based, network-based, or a hybrid (which is a combination of both data sources). Host-based IDSs provide local intrusion detection and support by monitoring user behaviour over an application layer protocol, such as the client-server protocol [60]. They run on an individual host or device on the network, monitor the inbound and outbound packets from that device only and alert the user or administrator if a suspicious activity is detected [61]. Network-based IDSs provide global intrusion detection, where they provide level monitoring of traffic flowing through the network and detect intrusions based on the nodes' behaviour over the network. A network-based IDS observes strategic points within the network to monitor traffic to and from all devices on the network [61]. Hybrid IDSs combine network and host-based monitoring in order to achieve maximum coverage, however there is an increase in system and security logs to analyse and an increase in system accounting.

Much of the proposed academic research on IDSs in the Cloud environment has focused on providing security mechanisms for specific security problems [62]–[94], while others adjust to the differing statistics available depending upon their data source [71-77]. Hamad and Al-Hoby [74] implemented the network-based Cloud Intrusion Detection Service (CIDS), which can be deployed by CSPs to enable clients to subscribe with the IDS in a service-based manner. It is a re-engineered version of Snort, which is an open source signature-based network IDS [75]. The model outperforms currently used solutions for service-based IDSs but at the same time provides minimal overhead to the case of traditional IDS deployment for single network protection. While it appears to be based on a network-centric approach, there are no details provided on how this approach can scale with an increasing Cloud environment. The option of a security service in a service-based manner is a future direction for CSPs so this approach is insightful.

Dhage et al. [76], convey that when there is only one IDS in the entire network, the load on it increases as the number of hosts increases. They highlight the challenge monitoring network based actions in the Cloud environment, in addition to actions on each of the hosts present in the network. An architecture in which mini IDS instances are deployed between each user of the Cloud and the CSP is proposed. As a result, the load on each IDS instance will be less than that on a single IDS and for this reason, the small IDS instance will be able to do its work in a more efficient way. By proposing a model in which each instance of IDS has to monitor only a single user, an effort has been made to create a coordinated design, which will be able to gather appropriate information about the user, thus enabling it to classify intrusions in a better way. The issue with such an approach is the mass of alerts that would be generated by all of the monitoring entities, so a clearer information hierarchy and communication structure is evidently needed.

Alsafi et al. [77] propose an integrated intrusion handling model for Cloud computing, which combines anomaly-based and signature-based detection. Their focus is on stopping an attack, rather than detecting it. Actions which their proposed method should take include terminating the user session that is being used during the attack, and blocking access to the target from the offending user account, IP address, or other attacker attribute. Their integrated model uses signature matching with normal traffic profiling to enhance attack detection. They propose to deploy their IDS in the VM itself as well as the virtual network in order to monitor the activities within the system.

Additionally, the use of multi-agents to increase the accuracy and scalability in network IDS is a thriving area [44] [78] [79] [80] [81]. It is suggested by Jansen et al. [82] that mobile agent technology can be beneficial in achieving the ideal behaviour desired in an IDS. The deficiency of a centralised IDS leads to the idea of multi or mobile agents. In an agent-based IDS there is no central point of failure and agents can detect and take predefined actions against malicious activity. The benefits of using a distributed model based on a mobile agent platform was highlighted by Patil et al. [58] where the key attributes were overcoming network latency, reducing network load, and being able to scale considerably. One challenge that would affect the application of an agent-based system is the application of ground truth for dynamic agent training [83], whereby training data and predefined information is needed for the functioning of agents.

Molina et al. [84] propose 'Distributed Architecture for Resource manaGement and mOnitoring in CloudS' (DARGOS), which is a monitoring support system for Cloud environments which is built upon a data-centric publish/subscribe paradigm. DARGOS's distributed architecture is based on two entities called Node Monitor Agent (NMA) and Node Supervisor Agent (NSA). NMAs are responsible for gathering monitoring data from the local node and delivering these data to interested nodes. NSAs are responsible for collecting monitoring data from remote hosts and making them available to final system administrators through DARGOS local application programming interface (API) and visualisation tools.

As highlighted, there has been an increase in intrusion detection and intrusion prevention methods for the Cloud environment [85-90]. Desired characteristics include optimised performance, minimum error and maximum protection [86]. The ability to adapt to changes in user behaviour and system behaviour over time is also anticipated. An IDPS should be part of normal services and not affect the operation of the Cloud environment in any way. Many solutions can only detect specific attacks, not unknown ones, and this is deterring the utilisation of the environment. A hybrid IDPS is needed for protecting the Cloud environment from attack with optimised performance and protection with a minimum error [86]. A hybrid approach combines two or more network intrusion detection techniques; signature-based detection, anomaly-based detection, and soft computing techniques. Using a hybrid approach can improve the accuracy of the IDS when compared to individual approaches.

3.1.2 Data processing

Regardless of whether they operate at the network, host, or application level, both IDSs and IPSs use one of two detection methods to classify and process data; anomaly-based or signature-based (also known as misuse detection) detection.

Anomaly-based IDSs detect abnormal patterns that deviate from what is considered to be normal behaviour [56]; examples include methods that are statistics-based [87], profile-based [88], or model-based [89]. While they can come in the form of different approaches, the overall method of detecting anomalous actions or behaviours is consistent. Anomaly detection does not require prior knowledge of intrusion and can detect new intrusions. However, a drawback of this is that they may not be able to describe what an attack is and may have a high false positive rate. Two problems that contribute to the large numbers of false positives produced include:

1) The decision whether an event should be classified as anomalous or as normal is made in a simplistic way. Anomaly-based IDSs usually contain a collection of models that evaluate different features of an event. These models return an anomaly score or a probability that reflects the 'normality' of the event according to their current profiles. However, the system is faced with the task of aggregating the different model outputs into a single final result [90].

2) They cannot distinguish between anomalous behaviour caused by unusual but legitimate actions and activity that is the manifestation of an attack. This leads to the situation where any deviation from normal behaviour is reported as suspicious, ignoring potential additional information that might suggest otherwise.

Misuse detection, commonly determined via signature-based IDSs, uses known patterns of unauthorised behaviour to predict and detect subsequent similar attacks [91]. Examples of misuse detection include approaches such as rule-based [92], state transition [17], and data mining [72]. Signature-based IDSs monitor and analyse network packets and compare them against the signature of the known threats. It is very effective in detecting known attacks or threats that are predefined in the database of IDS. Nonetheless, this systems main disadvantage that a new type of attack cannot be detected as its signature is not present [72]. Signature databases must constantly be updated, and IDSs must be able to compare and match activities against large collections of attack signatures. However, like anomaly-based detection approaches, false positives and false negatives are an issue.

Machine learning techniques or artificial intelligence (AI) systems have been applied to increase the efficacy of IDSs, however the focus is on improving the classifiers used [26]. Some of these techniques are neural networks, Linear Genetic Programming (LGP), Support Vector Machine (SVM), decision trees, Bayesian networks, Multivariate Adaptive Regression Splines (MARS) and Fuzzy Inference Systems (FIS) [70]. Viewed as a machine-learning algorithm, a Gaussian process [75] uses lazy learning and a measure of the similarity between points to predict the value for an unseen point from training data. In a Gaussian process, every point in some continuous input space is associated with a normally distributed random variable. Moreover, every finite collection of those random variables has a multivariate normal distribution, i.e. every finite linear combination of them is normally distributed.

The control centre models each malicious event as a Gaussian process, realised by a collection of random variables representing event features, such as the number of defective devices, and malicious authentication ratio. It uses the collected reports as observations to form the prior beliefs of the Gaussian process. Using the prior beliefs and observations, it computes the posterior probability distributions of the process through regression [93]. Gaussian processes are powerful nonparametric distributions over continuous functions that can be used for both supervised and unsupervised learning problems.

Many supervised and unsupervised learning approaches from the field of machine learning and pattern recognition have been using supervised learning approaches to classify their data. This involves using labelled samples to train a classifier, but obtaining sufficient labelled samples can be difficult, and requires the efforts of collaborative partners or expert connections. In contrast, unlabelled samples can easily be obtained for many real world scenarios. Compared to supervised learning approaches, semi supervised learning addresses this issue by considering a large number of unlabelled samples together with the labelled samples to build a better classifier [94].

In Mahmood and Agrawal [95], the focus is on ‘Principal Component Analysis Neural Network Algorithm’ (PCANNA) which is used to reduce the number of computing resources, both memory and CPU time required to detect an attack. Feature reduction is used to remove useless information from the original high dimensional database of Cloud traffic data. A back propagation algorithm is applied on reduced Cloud traffic data for classification. Their contribution shows that dimensional reduction techniques help compact similar alerts and correlate alerts coming from heterogeneous platforms on several sites to detect intrusions that are more complex.

3.1.3 Architecture

Many IDSs employ a centralised architecture and detect intrusions that occur in a single monitored system and/or network. Yet, centralised processers are not able to process collected data from massive network or distributed attacks, as currently more and more attacks appear to use a distributed architecture [59]. In centralised IDSs, the analysis of data is performed on a fixed number of locations, whereas in distributed IDSs analysis of data is performed on a number of locations that are proportionate to the number of available systems in the network. Within Figure 3-2 these different architectures and differing ways information is aggregated throughout are conveyed.

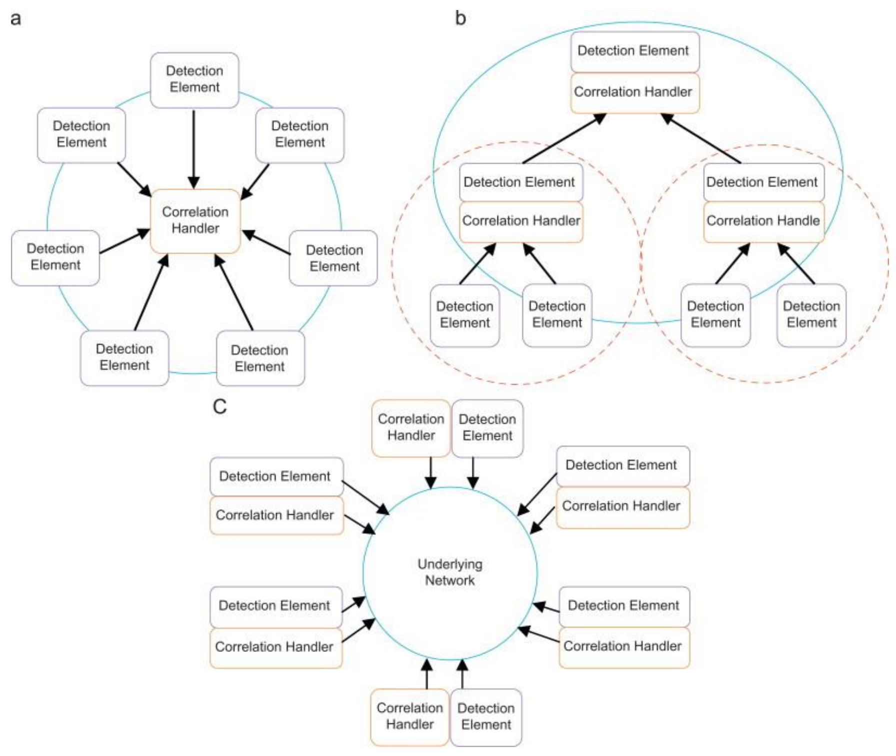


Figure 3-2: Comparison of communication architectures for detection systems

The scale of the data found in Cloud computing environments makes the application of existing techniques unfeasible, as they are unable to carry out analysis within reasonable temporal constraints. Table 3.1 presents a comparison between centralised and distributed IDS and their key attributes [72]:

Table 3.1: Comparison between centralised and distributed IDS desired characteristics

Characteristic	Centralised	Distributed
Run continually	A relatively small number of components need to be kept running.	Harder because a larger number of components need to be kept running.
Reliable	The state of the IDS is centrally stored, making it easier to recover it after a crash.	The state of the IDS is distributed, making it more difficult to store in a consistent and recoverable manner.
Resist subversion	A smaller number of components need to be monitored. However, these components are larger and more complex, making them more difficult to monitor.	A larger number of components need to be monitored. However, because of the larger number, components can crosscheck each other. The components are also usually smaller and less complex.
Minimal overhead	Impose little or no overhead on the systems, except for the ones where the analysis components run, where a large load is imposed. Those hosts may need to be dedicated to the analysis task.	Impose little overhead on the systems because the components running on them are smaller. However, the extra load is imposed on most of the systems being monitored.
Configurable	Easier to configure globally, because of the smaller number of components. It may be difficult to tune for specific characteristics of the different hosts being monitored.	Each component may be localised to the set of hosts it monitors, and may be easier to tune to its specific tasks or characteristics.
Adaptable	By having all the information in fewer locations, it is easier to detect changes in global behaviour. Local behaviour is more difficult to analyse.	Data are distributed, which may make it more difficult to adjust to global changes in behaviour. Local changes are easier to detect.
Scalable	The size of the IDS is limited by its fixed number of components. As the number of monitored hosts grows, the analysis components will need more computing and storage resources to keep up with the load.	A distributed IDS can scale to a larger number of hosts by adding components as needed. Scalability may be limited by the need to communicate between the components, and by the existence of central coordination components.
Graceful degradation of service	If one of the analysis components stops working, most likely the whole IDS stops working. Each component is a single point of failure.	If one analysis component stops working, part of the network may stop being monitored, but the rest of the IDS can continue working.

Dynamic reconfiguration	A small number of components analyse all the data. Reconfiguring them likely requires the IDS to be restarted.	Individual components may be reconfigured and restarted without affecting the rest of the IDS.

The greater scalability and larger size of Clouds compared to traditional service hosting infrastructure involve more complex monitoring systems, which have to be more scalable, robust, and fast. Such systems must be able to manage and verify a large number of resources and must do it effectively and efficiently. This has to be achieved through short measurement times and fast warning systems, able to quickly sport and report performance impairments or other issues, and to ensure timely interventions such as the allocation of new resources. Therefore, monitoring systems must be refined and adapted to different situations in environments of a large scale and highly dynamic like Clouds [96].

Cloud computing pushes to the extreme the concept of resource sharing in an environment that is inherently highly dynamic and with loads that are difficult to predict. Clouds are sometimes deployed in many data centres and clusters of servers, each of them possibly equipped with different characteristics and capabilities, i.e. in the same Cloud there could be clusters for CPU intensive computation, such as media transcoding and data indexing, while others will be optimized for input/output (I/O) throughput, such as media storage. As such, a Cloud monitoring system should be aware of logical and physical groups of resources and should organise monitored resources according to certain criteria so as to separate and to localise the monitoring functions.

Another crucial issue is system scalability in terms of processing and bandwidth overhead: for instance, medium-size Clouds data centres typically include hundreds of physical hosts and thousands of VMs; whereas larger ones can include thousands of physical hosts with multiple VMs (equipped with physical and logical sensors) for collecting monitoring data. These frameworks can eventually generate a great amount of network traffic that consumes precious network bandwidth; hence, the monitoring should be as least intrusive as possible by adopting lightweight processing and communication solutions that limit additional overhead. It also must assure timely monitoring data delivery.

A more network-centric approach is proposed by [68], in which they propose a distributed architecture for high speed intrusion detection involving the deployment of classification techniques to detect suspicious traffic patterns at the network layer. Moreover, the work of Bakshi and Yogesh [97] takes an innovative approach to securing the Cloud from DDoS attacks using an IDS in a VM. They propose the use of a virtual server to examine the security risks associated with the Cloud environment, and attempt to offer emergency response by identifying the source IP addresses involved in the DDoS attack. The virtual server would aim to respond to the attack by transferring the targeted applications to VMs hosted in another datacentre.

While anomaly-based network intrusion detection capabilities are becoming available and new schemes being explored [79], as with traditional IDS many key issues still remain to be solved [58], [59], [98]–[100] . Key research challenges in IDS include reduction of false positives and avoidance of false negatives, handling encrypted traffic, algorithms for effective content matching, deep packet inspection at wire-speed, performance improvement, latency reduction, behaviour-based detection, and environment awareness [101]. Decentralised network architectures allow participating nodes to share workload with others and thus avoid bottlenecks and single points of failure which are common weaknesses of centralised network architectures.

3.1.4 Reaction

An IDS, depending on the response system, is classified as either passive or active. Passive response is further divided into notification and manual response, whereas active response is considered automatic. By contrast, in a manual response system a predefined set of response options exists and is triggered by a security controller with the detection of an intrusion [102]. Intrusion responses are a series of actions and countermeasures when an intrusion is detected. In order to guarantee the security of computer networks, these actions and measures can prevent further attacks or restore the system to a normal state. These actions may come from human intervention or from computers.

According to the level of automation, current intrusion response systems can be categorised as notification systems, manual response systems, and automatic response systems. Notification systems mainly generate alerts about the intrusion which is then used by the system administrator to select an intrusion response [103]. A manual response system allows the administrator to manually launch countermeasures against a detected intrusion by selecting actions from a predetermined set of responses. In these two systems, the time duration within the detection and response activation opens an opportunity for attackers.

These systems rely on predefined datasets of normal and abnormal behaviour definitions. Also, many approaches rely on training data which can be classified through the use of appropriate data classifiers. These can be applied pre and post processing and can improve the accuracy of determining outliers from the data sets, but for real time monitoring can often have false positives and false negatives, as they often do not adjust well to the real time system.

The work of Lee [104] proposes a multi-level IDS and log management method based on consumer behaviour for applying IDS effectively to the Cloud system. They assign a risk level to users' behaviour depending on analysis of their behaviour over time. By applying differing levels of security to the users, in theory this would increase usage of resources as it would be based on the user behaviour over a period of time. Their method proposes the classification of generated logs by anomaly levels. This is so that the system administrator analyses logs of the most suspected users first.

Lo et al. [105] present a cooperative IDS framework for Cloud computing networks. They deploy an IDS in each Cloud region, and each entity cooperates with each other through the exchange of alerts to reduce the impact of DDoS attacks. A Snort-based IDS is implemented and the three main modules are plugged into the system: block, communicate, and defence. A cooperative agent is used to receive alerts from other IDSs, and they are analysed using a majority vote in order to determine the accuracy of results. If deemed as a legitimate alert, the blocking rule is implemented. By cooperative operation among these agents, an early detection and prevention technique is implemented. Therefore, IDSs deployed in Cloud computing regions except the victim one could prevent this kind of attack.

Distributed systems need to maintain a balance between communication overhead and the addition of process power, as resources can become constrained. Since Cloud computing supports a distributed service oriented paradigm, multi-domain and multi-users administrative infrastructure, it is more prone to security threats and vulnerabilities, such as data breaches, data loss, service hijacking, DDoS attacks and malicious insiders, to name a few [35].

3.2 Collaborative monitoring

Benefits of a collaborative monitoring scheme include greater efficiency and increased monitoring accuracy, which are a result from the collective pooling of resources for a single purpose. The structure of an IDPS is based upon two types: individual and collaborative. An individual arrangement of IDPS is achieved by physically integrating it within a firewall. A collaborative IDPS consists of multiple IDPSs over a large network where each one communicates with each other. Each IDPS has two main functional components: detection element and correlation handler. Detection elements consist of several detection components, which monitor their own sub-network or host individually and generate low-level alerts. The correlation handler transforms the low level alerts into a high level report of an attack [86].

For multi-tenant deployments, the monitoring system must also verify different requirements in terms of information granularity, accuracy, and update frequency. The monitoring infrastructure should be flexible enough to accommodate heterogeneous application-related requirements and to harmonize various tenant needs [84].

A further challenge identified is the ability to detect the threat to the domain of origin, and alert the user to their part in the attack, as it occurs or after the event. If DoS or spoofed traffic is originating from a VM within a CSP, then they would be charged for the excess traffic. This is an eDoS attack, due to the financial implications associated — the actual costings of such an occurrence are beyond the scope of our work. It is not always easy for the victim to determine that they are being attacked, or where it is coming from; as such, a Cloud-based monitoring system with the ability to trace the source and improve resilience to attacks within Cloud federation is essential.

The work of Calheiros et al. [106] conveys an InterCloud project through the use of agents called Cloud Coordinators, and allows for an increase in performance, reliability, and scalability of elastic applications. The architecture proposed for such a Cloud Coordinator could be applied to the intrusion detection domain. The Cloud Coordinator is the element that has to be present on each data centre that wants to interact with InterCloud parties. The Cloud Coordinator is also used by users and brokers that want to acquire resources via InterCloud and do not own resources to negotiate in the market.

In [107], Chen et al. aim to develop a new collaboration system to integrate a well deployed Unified Threat Management (UTM) via the Collaborative Network Security Management System (CNSMS). Such a distributed security overlay network coordinated with a centralised security centre leverages a peer-to-peer communication protocol used in the UTM's collaborative module and connects them virtually to exchange network events and security rules. The CNSMS also has a huge output from operation practice, i.e. traffic data collected by multiple sources from different vantage points, and operating reports and security events generated from different collaborative UTMs. As such, data is huge and not easy to analyse in real-time, and it needs to keep them archived for further forensic analysis.

The work of Wang et al. [108] proposes the use of D-S's theory of evidence to fuse local information. In the process of applying D-S in IDS, firstly, each agent collects information in its respective domain, and then the identification of some proposition is generated, which serves as evidence. Based on these, the degree of confidence is assigned on each proposition. As a basic probability assignment function (BPAF) and its corresponding frame of discernment are called a body of evidence, each sensor therefore corresponds to a body of evidence. The essence of multi-sensor data fusion is that within the same frame of discernment, different bodies of evidence, depending on fusion rules, are fused into a resultant BPAF, upon which the system makes the final decision based on decision rules.

Arachchilage et al. [109] convey the importance of collaborative parties having an established guarantee about the type of information they will be sharing in order to protect sensitive information. Their research focuses on the development of a taxonomy which can be applied to trust domains, aimed at managing trust related issues in information sharing schemes. The development of measurable trust characteristics is targeted at supporting collaboration and data exchange within and across multiple organisations.

While progress has been made on the problem of collaborative monitoring, there still remain a number of open problems that need to be addressed, such as [99]:

- Expressiveness – How to balance the trade-off between expressiveness of the correlation algorithm and corresponding computational complexity during alert correlation.

- Scalability – How to remove the need for a central controller in a Cloud-based IDS, without sacrificing the overall performance.
- Accuracy – How to improve the detection accuracy, i.e. how to balance the trade-off between the detection rate and false alarms.

3.3 Placement of solutions

Intrusion detection and prevention solutions are assessed on their capability to protect securely in large-scale networks, but the placement of such a solution can affect how it monitors the network. Non-intrusiveness is one of the key attributes for a monitoring entity, but monitoring large scale Cloud environments requires strategic planning and engagement. Considerations based on the size and the complexity of the architecture include: mobility of monitoring entities, no central points, constrained bandwidth of links, and limited resources. Some of the challenges remaining include questions on how to reinforce the intrusion detections and response elements to cope with intrusion and response in parallel, in addition to the coordination of information between monitoring entities, and management of multiple nodes. Attacks and failures are inevitable; therefore, it is important to develop approaches to understanding the Cloud environment under attack. Investigation into the appropriate points in the Cloud to deploy monitoring and attack detection functionality is imperative [85].

The four areas considered for deployment are in the VM, in the hypervisor or host system, in the virtual network, or in the traditional network.

- In the VM: Deploying a solution in the VM allows you to monitor the activity within the system, and detect and alert on issues that may rise.
- In the hypervisor or the host system: Deploying a solution in the hypervisor allows you not only to monitor the hypervisor, but anything travelling between VMs on that hypervisor. It is a more central location for intrusion detection, but there may be performance issues or dropping of some information if the amount is too large.
- In the virtual network/virtual local area network (VLAN): Deploying a solution to monitor the virtual network allows you to monitor the network traffic between VMs on the host, as well as traffic between the VMs and host. This 'network' traffic never hits the traditional network.

- In the traditional network: Deploying a solution here allows you to monitor, detect, and alert on traffic that passes over the traditional network infrastructure. However, this is quite problematic as virtual traffic as it is encrypted may be missed.

Xin et al. [65] present their innovative approach in the form of an intrusion detection mechanism for the Cloud computing environment. They propose a new intrusion detection mechanism based on Cloud computing called “IDCC”, and it is designed to support wide networks, offering high availability. Whereas, Montes et al. [110] implemented GMonE “Global Monitoring systEm", a Cloud monitoring tool which is capable of being adapted to different kinds of resources, services and monitoring parameters. GMonE performs the monitoring of each element by means of GMonEMon, which abstracts the type of resource (virtual or physical). GMonEMon service monitors the required parameters and then it communicates automatically with the monitoring manager (GMonEDB) to send it the monitored data. This communication is done through a standard Java Remote Method Invocation (RMI) process.

Zhang et al. [111] propose a hierarchical IDS framework with a distributed monitoring architecture. In their framework, an IDS module is distributed along the network on key services, i.e. on control centres, community gateways and smart meters. The IDS module is comprised of two components: classifier and recorder. The IDS module at the bottom layer accepts raw input data from smart meters; the module at a higher layer accepts input from the IDS module at the immediate lower layer invoking a hierarchical communication scheme which in turn would reduce network latency and throughput associated with monitoring large scale networks. As such, if an attack is detected by an IDS module, an alarm will be invoked on the corresponding layer. If a detection decision cannot be made at a certain layer it will be left for the upper to make it, since the upper layer has a wide scope of knowledge.

The optimal deployment location recognised is on the virtual network/VLAN. To communicate between VMs, they talk over a virtual network. This would be a suitable place for an IDPS as communicating occurs through this point, and it would be easier to build a nominal profile of activities and behaviours. Additionally, the use of a module that uses signature analysis of captured attack statistics, which also utilises a behaviour module to determine if the detected occurrence is actually an attack, could be beneficial. This could in turn improve efficiency over current methods that only utilise one method [85].

3.4 Research challenges

This chapter provides sufficient information to understand the challenges that are currently faced in relation to collaborative intrusion detection within Cloud federations. This section will summarise these research challenges.

- The first main challenge is due to the size and dynamic nature of the Cloud environment, namely the architecture and structure, which poses problems for the application of existing solutions. As Cloud environments, and Cloud federations, are large scale, it is essential that any potential solution should scale alongside the environment and have the potential to expand and scale considerably without any issues or performance implications. As every Cloud service delivery model is different from other similar service models, IDS techniques used for each Cloud service model also differ [27].

- The second main challenge is the issue of system scalability. The accuracy and efficiency of detection is important, but ensuring the solution is scalable and can deal with large volumes of logs from different sources is problematic. Having a solution with the ability to adapt to varying computational and network loads, in order to not be invasive, is essential. It is difficult, however, to achieve a good balance between IDS security level and system performance. Detecting intrusion patterns in the Cloud environment involves looking for behavioural changes. This process could involve anomaly-based detection for DoS/DDoS attacks, which must be robust against noise data, false positives and false negatives produced.

- A further challenge is that existing federation models are designed for static environments where priori agreements among the parties are needed to establish the federation [112]. Security vendors may not exchange information, e.g. malware reported from their customers, with other vendors because of privacy issues and competition. Providing prompt updates against the latest threats is important to dominate the market. Isolated antiviruses cannot obtain malware samples of zero-day threats to be analysed and may fail to protect their customers. However, from the customers' perspective, if diverse security vendors collaborate with each other, by means of providing feedback regarding the legacy of suspicious files, they may achieve even better accuracy [113]. It is with this in mind, that the benefits of security vendor collaboration, in our case, CSPs could benefit all parties involved.

3.5 Requirements for collaborative intrusion detection in Cloud environment

By using the information identified from the background research, it is possible to create a set of requirements which define the characteristics that potential collaborative intrusion detection solutions for the Cloud environment must possess - they combine the strengths and strive to eliminate the weaknesses of the work discussed previously. These requirements can therefore be used to assess the suitability of existing solutions, help to ensure the success of the proposed solution, and to provide a useful mechanism to evaluate the solution at a high level. These requirements were devised by examining the attributes of IDSs, monitoring schemes and their application to Cloud environments.

These attributes are as follows:

- Accurate: It must produce low levels of detection errors.
- Adaptable: It must be able to adapt to changes that occur within the underlying virtualised infrastructure, with considerations to roles, functionality, and structure.
- Collaborative: It should be able to exchange information and alerts within the infrastructure in an autonomous manner, without the need for human intervention. It should be able to take information and share it among key entities, receive and exchange relevant information when polled.
- Configurable: Each component may be localised to the set of hosts it monitors, and easier to tune to its specific tasks or characteristics.
- Dynamics: It must be able to cope with the adjusting behaviour and adaptive nature of the underlying virtualised infrastructure in order to formulate an accurate understanding.
- Efficient: Systems must operate seamlessly in real-time, and be able to manage and verify a large number of resources, and do it effectively and efficiently.

- Graceful degradation of services: If one analysis component stops working, part of the network may stop being monitored, but the rest of the monitoring schema can continue working.

- Low maintenance: It must not require much human intervention or offline maintenance to ensure accuracy or operation.

- Lightweight: It must be lightweight in terms of its resource and storage requirements (e.g. behavioural profiles, signature databases, and access lists).

- No prior knowledge: The operation of the solution should not depend on prior knowledge relating to the system and behaviour, due to the dynamic nature of the underlying environment and information may become outdated.

- Novel threats: It must possess the ability to detect novel threats, as due to the interconnected nature of Cloud federations, attacks may propagate from within the federation.

- Reliable: It must be able to operate constantly, have a high level of availability, and maintain an acceptable level of accuracy and efficiency. It must be able to recover from faults and ensure information is stored in a consistent and recoverable manner.

- Scalable: It must be able to automatically scale alongside the expansion of the underlying virtualised infrastructure, in order to adapt for its constantly changing needs.

These requirements will guide the research in terms of analysing the suitability of existing techniques, and developing a solution that satisfies the identified criteria. Table 3.2 provides a summary of the techniques discussed within this section, along with the benefits and drawbacks when considering their application for detecting intrusions within a Cloud federation.

Table 3.2: Intrusion detection techniques summary

Technique	Basic concept	Sub techniques	Benefits	Drawbacks
Anomaly detection-based	Detect abnormal patterns that deviate from what is considered to be normal behaviour.	1. Statistic based 2. Distance based 3. Rule based 4. Profile based	Ability to detect novel threats. No prior knowledge about infrastructure activity needed.	Cannot describe what the attack is. Can result in high false positive or false negative results.

		5. Model based	Accurate when deviance matches profiles.	
Signature detection-based	Use of known patterns of unauthorised behaviour to predict and detect subsequent similar attacks.	1. Misuse detection 2. Rule based 3. State transition 4. Data mining 5. State model 6. Expert system 7. String matching	It is very effective in detecting known attacks or threats that are predefined in the database of IDS.	Cannot detect novel attacks. If malicious activity appears like normal traffic to the system it will never send an alarm to administrator, so like anomaly based detection, false positives and false negatives is an issue. Signature databases must constantly be updated, and IDSs must be able to compare and match activities against large collections of attack signatures.
Machine learning based	Intelligent and adaptive classification of complex and uncertain behaviour.	1. Neural networks 2. LGP 3. SVM 4. Decision tree 5. Bayesian networks 6. MARS 7. FIS 8. Clustering and data outliers	Designed for complex environments. Tolerates incomplete data. Can adapt to the behaviour of the monitored environment.	Dependent upon knowledge of infrastructure boundaries and goals. Requires extensive training of data and classifiers. Over classifying can skew the results. Higher resource consumption.
Host-based monitoring	Provide local intrusion detection and support by monitoring user behaviour over an application layer protocol.	Host based monitoring can be applied either on the VM or on the host machines.	Host based monitoring has access to a greater spectrum of data sources covering the OS, as well as incoming and outgoing traffic. As host based IDS use system logs containing events that have actually occurred, they can determine whether an attack occurred or not with greater accuracy and fewer false positives than a network-based system.	It is unable to match the speed of network monitoring. Incurs resource and performance overheads. Suited to individual or isolated systems/infrastructures. Attacks can be identified by looking at log files, which often gives the attacker time to remove evidence of an attack.
Network-based monitoring	Examines and analyses the traffic to and from	Can detect different types of attacks in VMs and hypervisor.	Easy to deploy and does not affect existing	Network-based monitoring cannot

	all the devices on the network. It can scan all inbound and out-bound traffic and detect various types of instances.		systems or infrastructures. Once the attacks have been detected from this method the active system immediately takes necessary actions to tackle the attacks. Network-based IDS use live network traffic and perform real time intrusion detection. Therefore, the attacker cannot remove evidence of attack – this data can be used for forensic analysis.	detect the attacks inside virtual networks. Also the network traffic within and outside of the Cloud environment cannot be decrypted to analyse the network traffic. Generate many false positives because of the very fact that it detects malicious packets in real time and some of these packets could be from a trusted host. Cannot give information about system activities, whereas host based monitoring can detect policy breaches, abnormal user activity, user account modification, etc.
Agent-based monitoring	Agent-based monitoring use the independent and autonomy characteristics of agents to increase system scalability, and improve the problems caused by failure of single point, improve the system's fault tolerance.	1. Multi-Agent Distributed IDS (MAIDS) 2. Autonomous Agents For Intrusion Detection (AAFID) 3. Intrusion Detection Agent System (ADI) 4. Intelligent Mobile Agents for Intrusion Detection System (IMA-IDS)	Agent-based monitoring utilises multiple agents to achieve different modules of each intrusion detection unit. Each agent can communicate with each other, mutual cooperation. There is no central point of failure and agents can detect and take predefined actions against malicious activity. Scalability - agents reduce the computational load on the system by dividing it into hosts.	Need a clearly defined monitoring hierarchy and roles for agents.
Centralised monitoring architecture	Data collected from single or multiple hosts. All data shipped to a central location for analysis.	N/A	A smaller number of components need to be monitored. However, these components are larger and more complex, making them more difficult to monitor.	Single point of failure. Impose little or no overhead on the systems, except for the ones where the analysis components run, where a large load is imposed. Those hosts may need to be dedicated to the analysis task

				Hierarchical approaches are encouraged as they scale better than the centralised approach.
Distributed monitoring architecture	A distributed IDS consists of multiple IDSs over a large network, all of which communicate with each other, or with a server that facilitates advanced network monitoring, incident analysis, and instant attack data.	N/A	Ability to detect attack patterns across an entire corporate network, with geographic locations separating segments by time zones or even continents. This could allow for the early detection of a well-planned and coordinated attack.	Impose little overhead on the systems because the components running on them are smaller. However, the extra load is imposed on most of the systems being monitored. Network data rates are very high. Encryption of network traffic is becoming more popular.

Table 3.3: Comparison of existing techniques against collaborative intrusion detection requirements

	Anomaly detection	Signature detection	Machine learning	Host based monitoring	Network based monitoring	Multi agent based monitoring	Centralised architecture	Distributed architecture
Accurate	✓	✓	✓	✓	✓	✓	✓	✓
Adaptable	✓	▨	▨	▨	☐	✓	▨	☐
Collaborative	☐	☐	☐	☐	✓	✓	▨	✓
Configurable	✓	✓	✓	✓	✓	✓	☐	☐
Dynamics	✓	✓	✓	▨	✓	✓	▨	☐
Efficient	✓	✓	✓	✓	✓	✓	✓	✓
Graceful degradation of services	☐	☐	☐	▨	✓	✓	▨	✓
Low maintenance	✓	☐	☐	▨	☐	✓	▨	☐
Lightweight	✓	▨	▨	☐	✓	✓	☐	✓
No prior knowledge	✓	▨	▨	▨	☐	✓	☐	☐
Novel threats	✓	▨	✓	▨	☐	✓	☐	☐
Reliable	✓	✓	✓	✓	✓	✓	✓	✓
Scalable	✓	✓	✓	✓	✓	✓	✓	✓

✓	Requirements met
▨	Requirements not met
☐	Requirements partially met

Table 3.3 compares the existing techniques discussed in this chapter against the requirements established in Chapter 3.4. Taking the existing techniques, we identify if they: meet the identified requirements, do not meet the requirements, or partially meet the requirements. Partial requirements met would indicate that the technique could be tailored depending upon the user requirements.

3.6 Threshold algorithms

While conducting the requirements analysis collaborative monitoring has been identified as the most suitable technique for detecting intrusions in Cloud federations. It can be a highly effective approach but relies on accuracy of its set-up, and the predefined threshold parameters used to detect anomalous activities within the network.

The threat-awareness capability provides a key opportunity for an IDS to improve its detection rate. It can be inferred that as the use of Cloud in organisations grows, so will the rate of DoS attacks. These attacks against the Cloud are launched to deny service availability to end users. While DDoS attacks tend to generate a lot of fear and media attention, they are by no means the only form of DoS attack. Asymmetric application level DoS attacks take advantage of vulnerabilities in web servers, databases, or other Cloud resources, allowing a malicious individual to take out an application using a small attack payload – in some cases less than 100 bytes long [114].

For detecting such attacks there are a range of approaches considered in the literature [26], [45], [62], [72], [102], [115]–[119]. Algorithms for detecting attacks in the Cloud environment include adaptive threshold [115], Random Early Drop (RED) [120], RRED, and CUSUM. Each of these is used by a variety of solutions in the area of network and Cloud security, but each has differing benefits and drawbacks.

The adaptive threshold algorithm [115] detects anomalies based on violations of a threshold that is adaptively set, based on recent traffic measurements. Seasonal variations and trends are taken care of by using an adaptive threshold whose value is set based on an estimate of the mean number of packets under consideration or the rate, either of which is computed from recent traffic measurements.

RED takes a different approach, monitoring an average queue size and dropping packets based on statistical probabilities. If the buffer is almost empty, all incoming packets are accepted. As the queue grows, the probability for dropping an incoming packet grows too. When the buffer is full, the probability has reached 1 and all incoming packets are dropped. In comparison, RRED was proposed to improve the TCP throughput against DDoS attacks, particularly low rate DDoS (LDDoS) attacks. Experiments have confirmed that the existing RED-like algorithms are notably vulnerable under LDDoS attacks due to the oscillating TCP queue size caused by the attacks. The RRED algorithm can significantly improve the performance of TCP under LDDoS attacks [121].

LDDoS attacks reduce network service capabilities by periodically sending high intensity pulse data flows. For their concealed performance, it is more difficult for traditional DoS detection methods to detect LDDoS attacks; at the same time the accuracy of the current detection methods for LDDoS attacks is relatively low.

Many algorithms, such as random sampling, do not take into account traffic dynamics. As a result, they cannot guarantee the sampling error falls within a prescribed error tolerance level. How to discover the evolving process of the network traffic and how to improve the accuracy of real-time detection is problematic. For services in the Cloud environment, the confidentiality, integrity, and availability of the data is of utmost importance. Our Service-based intrusion detection framework will facilitate collaborative intrusion in a Cloud federation, protecting the Cloud environment from DDoS attacks, as these attacks can threaten the availability of Cloud functionalities and/or the services within.

CUSUM [122] involves the calculation of a cumulative sum. Samples from a process $\mathrm{p_t}$ are assigned weights $\mathrm{W_t}$, and summed as follows:

$$r_0 = 0 \tag{3.1}$$

$$r_{t+1} = \max(0, r_t + p_t - w_t) \tag{3.2}$$

When the value of r exceeds a certain threshold value, a change in value has been found. The formula only detects changes in the positive direction. t refers to a period of time.

There are preventive measures in place to protect against such attacks, but they seem to be focusing on generic DDoS, where the characteristics mimic previous attacks of such nature. However, the rise in high volume and low rate DDoS is a problem. They could be spread out over a period of time, and have random high bursts, which can confuse the preventative measures. Algorithms for detecting DDoS attacks in the Cloud environment often sample the packets and drop any deemed to be malicious – these can often be false positives.

The use of algorithms for detecting attacks in the Cloud environments has the following weaknesses:

- Sample packets are often inaccurate
- Vulnerability to unknown types of DDoS attacks
- Does not always ensure the accuracy of estimation and tends to over sample at peak periods when efficiency and timeliness are crucial
- Random sampling does not take into account traffic dynamics
- Inefficient on LDDoS attacks
- Prone to error on high rate DDoS attacks

There is an emerging need for the traffic processing capability of an IDS to match the high throughput of today's high-bandwidth networks. Recent research has shown that the vast majority of security solutions deployed today are inadequate for processing traffic at a sufficiently high rate to keep pace with the network's bandwidth [123]. Existing sampling algorithms are poorly suited for this task, especially because they are unable to adapt to the trends in network traffic. Satisfying such a criterion requires a sampling algorithm to be capable of controlling its sampling rate to provide sufficient accuracy at minimal overhead [123]. The adaptive threshold algorithm and CUSUM algorithm appear to be the most applicable for detecting attacks in the Cloud environment. Algorithms based on change point detection, such as CUSUM, can exhibit robust performance over a range of different types of attacks, without being more complex.

3.7 Decision making algorithms

There are many decision making algorithms available in the literature, however by adopting a vector-based voting solution the failure rate can be significantly reduced compared to non-vector voting by about 50% [124]. If a voter is a yes/no decision maker, its output space is binary; and if the output of a vote can be any value, its output space is infinite. The majority vote produces an output among variant results, where at least $(n + 1)/2$ variant results agree. The plurality voter is the relaxed form of majority voting, and implements m-out-of-n voting, where it is less than a strict majority (e.g., 2-out-of-5 or 3-out-of-7 voting) [124].

The disadvantage of the widely used majority vote, as well as the plurality, is that they may agree on incorrect variant results, where there is a consensus on identical incorrect inputs. In other words, these voters cannot distinguish between agreed correct and agreed incorrect variant outputs. The majority vote is often inaccurate, especially in automated approaches.

For example: assume a voter with 11 inputs received from software versions for which the output space is binary (0, 1). Five versions have reliability 0.99 (type A versions), and six versions have reliability 0.95 (type B versions). For a given notional correct input, the A-versions output 0, and the B-versions produce 1. According to the majority voter, the correct result is estimated to be 1. However, if the reliability information of variants is taken into account in estimating the output, a more accurate output may be obtained [124].

This shows the benefit of using the extra information from variants during the voting process, with the ultimate aim of having a more accurate result. A group of voters, which differ from generic voters, use extra information such as the reliability level of variants, online diagnosis information of modules, or various probabilistic information to improve voting performance, which is known as a ‘hybrid voter’ [124]. This type of approach would generate a more accurate output compared with the aforementioned voting algorithms.

D-S theory of evidence is an example of such an approach, and can solve the problem of collaborative intrusion detection in the federated Cloud environment. The D-S theory offers an alternative to the traditional probabilistic theory for the mathematical representation of uncertainty. D-S applications range from expert decision support systems to multi-attribute decision-making and data fusion [125]. The main advantage of this algorithm is that no a priori knowledge of the system is required, thus making it suitable for anomaly detection of previously unseen information [126]. A further point for decision making is the aspect of time criticality, which includes both the responsiveness aspect of the system and the timeliness of any related data being delivered within its designated time period.

3.8 Dempster-Shafer (D-S) theory of evidence

D-S theory of evidence in the context of distributed intrusion detection can demonstrate the theory's usefulness. Cooperative decision making is made after aggregating evidence using this approach. As mentioned, some work in this area [105] uses a simple majority vote to reach a decision when combining evidence, where the final decision is a binary decision. D-S produces a judgement value between 0 and 1 that reflects the degree of belief in that judgement [127]. The computational complexity of D-S increases exponentially with the number of elements in the frame of discernment (θ). If there are n elements in Θ, there will be up to 2^{n-1} focal elements for the mass function. Furthermore, the combination of two mass functions needs the computation of up to 2^{n} intersections. The D-S rule of combination is the procedure to aggregate and summarise a corpus of evidences.

D-S theory is a probabilistic approach, which implements belief functions which are based on degrees of belief or trust. Probability values are assigned to sets of possibilities rather than single events [60]. In the decision making process, the uncertainty existing within the network often leads to failure of intrusion detection or low accuracy. D-S was introduced as a mathematical framework for the representation of uncertainty and analysing it in a quantitative way. D-S theory introduces the concept of assigning beliefs and plausibilities to possible hypotheses of each decision maker and provides a combination rule to fuse multi-modal information. This theory allows each source to incorporate information with different levels of evidence. This property provides a significant benefit, in that, it assigns a possibility mass to a subset or interval; hence, the D-S theory based fusion approaches can efficiently address both probabilistic (or objective) uncertainty and epistemic (or subjective) uncertainty [128].

Other methods such as using Bayesian and learning based approaches cannot provide these benefits. They are dependent upon knowledge of infrastructure boundaries and goals, which leads to extensive training of data and classifiers. The Bayesian approach requires complete knowledge of both prior and conditional probabilities, which might be difficult to determine in practice. Prior probabilities are often estimated from empirical data, or, in the absence of empirical data, the assumption is that they are uniform or some other distribution. The Bayesian approach is not well equipped to handle states of ignorance [127].

Collaborative intrusion detection has been considered in several contributions where data provided by heterogeneous intrusion detection monitors is fused. Conversely, there is a trade-off between accuracy and efficiency while the decisions are made in an IDS. Cognitive algorithms, such as neural networks, have good adaptability but require a lot of training data, which is hard to capture in a real network environment [129]. The main advantage of D-S is that no priori knowledge of the system is required, i.e. state transition matrix or training data, thus making it suitable for anomaly detection of previously unseen information [126].

Within the domain of intrusion detection, comparison of the performance of voting and decision algorithms is complex for two reasons. Firstly, the large number of environmental parameters that have a direct impact on the voter output, e.g. input data profile, fault/error type, error probability distribution, reliability level of variants, number of variants, and the value of acceptance thresholds. Secondly, the selection of an appropriate metric for the evaluation.

Furthermore, the metrics are application dependent; for example, where the output space of a system is large and the probability of producing identical incorrect redundant results is low, the number of agreed results, during a specific running time, is a suitable metric. Whereas if the cardinality of voter output space is small and identical, incorrect redundant results are probable, then the number of agreed and correct results is a suitable metric.

In other applications, the ratio of correct results to agreed results may be the most suitable measure. Inevitably, the ranking of voter performance may change when using different metrics. Examples of typical metrics used include the error detection ratio, false alarm ratio, the number of normalised benign outputs, the number of normalised catastrophic outputs, and the probability of producing a correct voter output [124].

According to Chen and Aickelin, the computational complexity of D-S increases exponentially with the number of elements in the frame of discernment (Θ). If there are n elements in Θ, there will be up to 2^{n-1} focal elements for the mass function. The combination of two mass functions needs the computation of up to 2^n intersections. The D-S rule of combination is the procedure to aggregate and summarise a corpus of evidences. However, through literature [124], [126], [136] and implementing our Service-based collaborative intrusion detection solution, the limitations shown below are evident:

- Associative: for rule combination, the order of the information in the aggregated evidences does not impact the result. A non-associative combination is necessary for many cases.
- Non-weighted: rule combination implies all evidences are trusted equally. However, in reality, trust on different evidences may differ, which means various factors should be considered for each evidence.

3.9 Problem analysis

Anomaly-based IDSs are a principal focus of research and development in the field of intrusion detection [56]. It is clear there are two main weaknesses in present-day IDS techniques [101]. One weakness is that they do not take into consideration the threat exposures in the network while detecting intrusions, resulting in obtaining alerts for all types of events, many or most of which may not be relevant to the operating environment. In the context of dynamic network environments such as WSNs or Cloud networks, this approach may lead to a huge number of unnecessary alerts. Depending on the frequency of changes to the network environment, this may in turn affect the efficacy of the IDS or detection method itself.

A second weakness of present-day IDS techniques is that they are not devised to handle dynamic network environments, as they use a stringently predefined set of signatures and anomaly detection thresholds to detect intrusions. Furthermore, they are quite ignorant of any changes to the operating environment that may eliminate or introduce vulnerabilities and threat exposures. Due to this weakness, the monitoring technique may miss critical attacks and detect intrusions that are not relevant due to changes to the environment. Hence, introducing a threat awareness capability provides a key opportunity for an IDS to improve its detection rate [101].

From our analysis, it is clear that Cloud computing deteriorates the perception of perimeter security. It has become impossible to place a virtual moat around an organisation’s castle, as an abundance of services have been outsourced [15]. Security should be implemented in every layer of the Cloud application architecture. It is important to note that the existing approaches in this area do not tackle the protection of services migrating to the Cloud environment efficiently. In current solutions, they provide theoretical intrusion methods for the Cloud infrastructure, or for network protection, but these are not sufficient for our protection problem. The distributed nature of the Cloud model makes it an even more attractive target for intruders.

The deployment of WSNs and mobile ad-hoc networks in applications such as emergency services, warfare and health monitoring poses the threat of various cyber hazards, intrusions and attacks as a consequence of these networks' openness. Among the most significant research difficulties in such networks' safety is intrusion detection, whose target is to distinguish between misuse and abnormal behaviour so as to ensure secure, reliable network operations and services. Intrusion detection is best delivered by multi agent system technologies and advanced computing techniques. To date, diverse soft computing and machine learning techniques in terms of computational intelligence have been utilised to create IDPSs, yet the literature does not report any reviews investigating the performance and consequences of such techniques solving wireless environment intrusion recognition issues as they gain entry into Cloud computing [80].

According to Zuech et al. [130], improvements to intrusion detection could be achieved by embracing a more comprehensive approach in monitoring security events from many different heterogeneous sources. The associated benefits of such an approach could offer a more holistic view and greater situational awareness of cyber threats. The key problem being tackled is that, as cyber-attacks have evolved and grown in sophistication, monitoring an ever increasing number of event sources has grown in complexity. Intelligence awareness is a growing method which is the capability of automated intelligence sharing and alerting across a myriad of security systems, and a benefit is the ability to adapt, based on contextual or situational awareness.

A monitoring system that can run in a non-intrusive and transparent manner to any underlying virtualised infrastructure is an evident requirement within this area. Additionally, possessing the ability to adapt to behavioural changes (i.e. resource related parameter changes) within the system without the need for re-contextualisation each time. Design of a collaborative Cloud-based framework that focuses on detecting intrusions within an interconnected infrastructure, in combination with decision making and/or fusion algorithms is a must. As identified, one of the key issues identified is the lack of communication between entities monitoring the systems, or the sharing of information. Through efficient communication and exchanging threat information, a threat could be prevented from propagating throughout a network.

3.10 Summary

This chapter has provided a summary of related work and existing approaches for detecting intrusions in the Cloud environment. It has reviewed the identified techniques, providing an overview and outlining the benefits and drawbacks when considering application to a Cloud federation. This exploration has identified that there is a consensus in existing work on the need for improved approaches for protecting infrastructure services in the Cloud environment, and in Cloud federations [51], [74], [131]–[133]. This is a challenging and complex problem and environment, as the confidentiality, integrity, and availability of data can have high socioeconomic implications as there are more and more infrastructure services migrating to this environment. Information in the Cloud environment comes from many sources and in many formats.

During the literature review, each of the requirements identified in Chapter 3.5 were examined, and it can be concluded that there are no currently defined methods that properly fulfil the requirements for collaborative intrusion detection for Cloud federations. The review identifies solutions that could be built upon and scaled to deal with the dynamic nature of the Cloud environment. This chapter justifies why a novel approach and novel techniques are essential to addressing the issue of protecting critical infrastructure services in the Cloud environment via collaborative intrusion detection and autonomous sharing of information. The inadequacies of existing solutions provide both the motivations and aims for the solution proposed in this thesis.

Chapter 4

Service-based collaborative intrusion detection framework

In this chapter, we provide the design of our Service-based collaborative intrusion detection framework and the technical challenges that we address. The contents of this chapter are structured as follows. An overall overview of our solution is presented in 4.1. A high level overview of the framework is presented in 4.2, and a detailed explanation of the architectural attributes. The algorithms used for threshold adaptation and belief generation are presented in 4.3 and 4.4, and the collaborative decision making process is highlighted in 4.5 - with the extended D-S fusion process in 4.6. Lastly, 4.7 provides a summary of the framework.

In Chapter 3, we identified that there is a consensus in existing work on the need for improved approaches for protecting infrastructure services in the Cloud environment, particularly in Cloud federations. As such, Cloud defence strategy needs to be distributed so that it can detect and prevent the attacks that originate within the Cloud itself and from the users using the Cloud technology from different geographic locations. As the popularity of the services provided in the Cloud environment grows rapidly, the exploitation of possible vulnerabilities grows at the same pace [134].

It is obvious from Chapters 1-3 that there is a need for a new and capable collaborative intrusion detection method to cope with the challenging structure, dynamic nature and scale of the Cloud computing and Cloud federation environment. The literature review and survey of related work in Chapter 3 has shown that there are no entirely suitable solutions, and has detailed the limitations of existing solutions and the shortcomings of such techniques. Therefore, a novel approach is required to establish autonomous intrusion detection in a Cloud federation via the collaborative interconnected domains. CSPs within a Cloud federation are represented as interconnected domains, and based on the proposed monitoring structure, the infrastructure is dynamically provisioned to react and facilitate autonomous sharing of threat information via Cloud brokerage and the hierarchical monitoring entities. This approach needs to adhere to the aims and objectives established in Chapter 1.2 and the requirements outlined in Chapter 3.5.

This chapter proposes a novel approach to the problem through Security as a Service using our Service-based collaborative intrusion detection framework. Our approach is proactive and looks to identify patterns of traffic and behaviour, and through collaborative intrusion detection conveys a security posture rather than purely alerting to attacks. The main requirement of our solution is to provide protection for critical infrastructure services being hosted in the Cloud environment, in particular within Cloud federations, through novel intrusion detection techniques. Our solution encompasses the use of dynamic and multi-threshold-based algorithmic approach, and autonomous sharing of information to improve resilience to Cloud attacks.

4.1 Overview of Service-based collaborative intrusion detection framework

So far, this thesis has discussed in detail the extent of the limitations of existing solutions and the challenges faced for collaborative intrusion detection within Cloud federations. In order to resolve this, a novel solution has been devised called Service-based collaborative intrusion detection framework, providing "Security as a Service" for Cloud federations [135]-[136]. A high level overview of the key attributes and functionality of the solution includes an autonomous information sharing schema by the monitoring entities, whereby collaborative intrusion detection is facilitated.

It has been designed to ensure it fulfils the requirements established in Chapter 3.5 and that it is non-invasive and transparent to the underlying virtualised infrastructure. It operates by monitoring the live behaviour of the network in real-time, utilising pre-calculated CUSUM profiles for network thresholds. These profiles and their calculated values are discussed in Chapter 4.2. These algorithms are used to cope with the dynamic nature of the Cloud environment, and by utilising adaptive thresholds, attacks can be identified. Our solution generates high level application metrics dynamically at runtime by aggregating and grouping low level metrics.

Our solution conveys knowledge acquisition in dynamic Cloud environments through collaborative cooperation. Collaborative intrusion is achieved via the generation and exchange of beliefs – detailed in Chapter 4.5 – based on observations of events within the network. Via the use of a broker, the ability to collaboratively make a system wide decision and propagate this through the interconnected domains is achieved.

4.2 Service-based collaborative intrusion detection architecture

CSPs within a Cloud federation are represented as interconnected domains, and based on the monitoring structure, dynamically provision the infrastructure to react and facilitate autonomous sharing of threat information via a Cloud broker. Brokering functions in federated Clouds at the IaaS layer can be decomposed into two aspects: resource provisioning and resource adaptation. In resource provisioning, the most appropriate mix of resource classes and the number of nodes of each resource class are estimated so as to match the requirements of the application and to ensure that the user objectives and constraints – i.e. throughput, precision and efficacy – are satisfied [47].

It is necessary to continuously monitor the application execution and adapt resources to ensure these objectives and constraints are fulfilled. Resource adaptation is responsible for provisioning resources dynamically and at runtime. Examples of such provisioning are assigning more physical CPUs to a given VM to speed up an application, or migrating VMs in order to reduce the resource sharing or optimise the energy efficiency.

With a Cloud federation, it is a requirement that each CSP has to share Cloud-related information with the federated Cloud providers. With this in mind, in our architecture, this sharing of knowledge in our approach would involve security information about malicious activities, new signatures, and suspicious IP addresses (with an SLA detailing such a requirement). Our architectural knowledgebase encompasses a hybrid approach that combines network intrusion detection techniques: signature-based detection, anomaly-based detection, and soft computing techniques.

Using a hybrid approach can improve the accuracy of the IDS when compared to individual approaches. Our Security as a Service entity would be deployed in the virtual network in each CSP domain, and Figure 4-1 illustrates such a model. Conceptually, the architecture is structured into the components presented in Table 4.1.

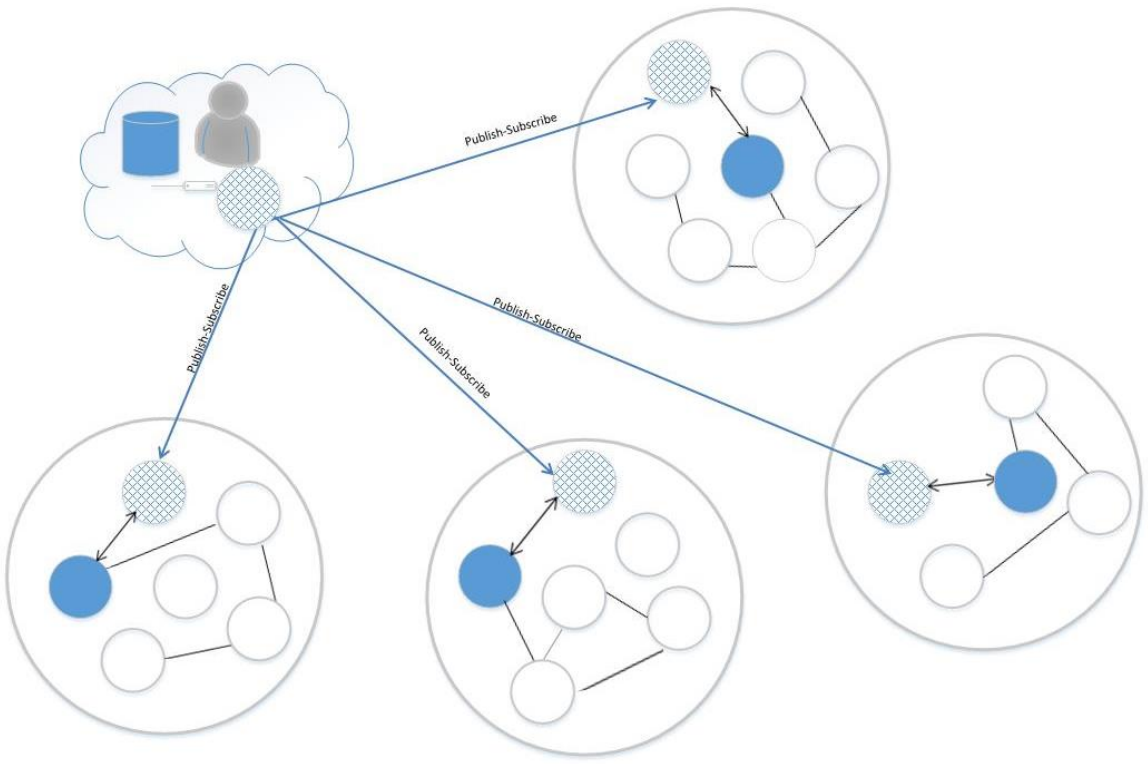


Figure 4-1: Service-based collaborative intrusion detection in a Cloud federation

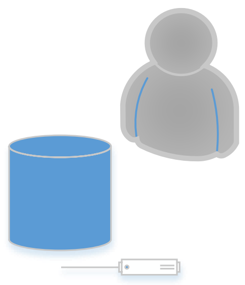
Within our architecture, time criticality includes both the responsiveness aspect of the system and the timeliness of any relevant data being delivered in its designated time period. The autonomous sharing of information within the Cloud federation promotes that resilience is a worthwhile and obtainable attribute.

Figure 4.1 shows how the key attributes are connected:

- Cloud broker
- Monitoring Nodes (MNs),
- Local coordinators (Super Nodes - SNs),
- Global coordinators (Command and Control server - C2).

The C2 provides management of SNs and MNs, responses to attacks detected and reported by SNs and/or the broker, and cooperates with adjacent domains when polled. Characteristics of each entity are detailed in Chapter 4.3.1 – 4.3.4, in addition to flowcharts illustrating their interactions.

Table 4.1: Notations for Figure 4-1

Icon	Attribute
	Cloud Broker with database server of signatures: • Black list • Grey list – local to each C2 • White list
◍	Command and Control server (C2)
●	Super Node (SN)
○	Monitoring Node (MN)

Due to resource limitations in our test environment, a smaller amount of attributes were included to demonstrate proof of concept but this distributed monitoring schema can be scaled considerably. As stated in Chapter 2, Cloud environments and Cloud federations are large scale, it is essential that any potential solution should scale alongside the environment and have the potential to expand and scale considerably without any issues or performance implications – this is achieved by the architectural design of our Service-based collaborative intrusion detection solution.

The 'Security as a Service' inspired architecture operates by monitoring the actions of the Cloud network in real time against pre-defined behavioural threshold profiles that take into consideration adaptations in performance. CUSUM is used for monitoring change detection

[120], detailed in Chapter 4.3. Additionally, in conjunction the EWMA can be determined, which is a statistic for monitoring the process that averages the data in a way that gives less and less weight to data as they are further removed in time. By comparing live network behaviour against these profiles, attacks against the network can be quickly detected and pre alarms generated based on the frequency of occurrence.

4.3 Threshold calculation algorithms – CUSUM and EWMA

Within our Service-based collaborative intrusion detection architecture, network-based intrusion detection is used to provide global intrusion detection, where it provides level monitoring of traffic flowing through the network and detects intrusions based on the node behaviour over the network. Related work within this research uses signature-based detection schemes to detect violations within their networks but due to the dynamic nature of the Cloud, an approach which considers fluctuations within this defined ‘norm’ is more appropriate. Adjusting thresholds based on these fluctuations would improve the accuracy and efficiency of our methodology, and lower the number of false positives and false negatives.

CUSUM is a sequential statistical analysis technique, which is used to monitor and detect changes within any sequence of quantitative observations in some experiment, i.e. monitoring for any sudden increase or decrease in the number of incoming messages or traffic fluctuations [137].

CUSUM takes samples p_t from a process and assigns weights W_t to these values, which is summed as follows: $r_0 = 0$

$$r_{n+1} = \max(0, r_t + p_t - W_t) \tag{4.1}$$

When the value of *r* exceeds a certain threshold value, a change in value has been found. The traffic threshold is given by:

$$(\alpha+1)\mu \tag{4.2}$$

Where:

- t: a period of time
- α: alpha
- μ: is the measured mean rate

Unlike the adaptive threshold algorithms [115], [117], which consider only violations of the threshold, the CUSUM algorithm considers the excess volume sent above the normal volume, hence accounts for the intensity of the violations.

There have been attempts to determine the optimal level of each threshold but the best performing approach is to use the standard deviation σ of the whole sequence. CUSUM is a recursive equation, meaning that a new CUSUM value is dependent on the previous CUSUM values. After a new input value to the sequence of the observations is obtained, it will be used to find a new CUSUM value. CUSUM is computed according to the following equation:

$$C_i = O_i - (\mu_o + Z + \sigma) + C_{i-1} \tag{4.3}$$

Where:

- C_i: is the most recent CUSUM value to be computed
- O_i: is the most recent observation value
- μ_0: is the overall observations mean value
- C_{i-1}: is the previous CUSUM value

Z is the reference value and is chosen about halfway between the target $\mu_{t-1}$and the control value of the mean μ_1. Z is calculated by:

$$Z = \frac{\delta . \sigma}{2} \tag{4.4}$$

$$\delta = \frac{|\mu_t - \mu_{t-1}|}{\sigma} \tag{4.5}$$

If the shift was expressed by the standard deviation units, then Z is one-half the magnitude of the shift and it is given by [137]:

$$Z = \frac{|\mu_i - \mu_{i-1}|}{2} \tag{4.6}$$

As CUSUM computes the differences between the values and the average, it is able to detect and plot small changes in a sequence of quantitative observations, so it can be employed for detecting DDoS as it can detect any sudden or subtle changes in the incoming traffic [137].

An alarm is signalled when the accumulated volume of measurements g_t that are above a traffic threshold, exceeds the aggregate volume threshold h. The threshold is set adaptively based on recent measurements of the mean rate. The classic CUSUM formulation is effective in change detection, however in an anomaly detection scheme there is no interest in the change itself, focusing instead on the intensity of the anomaly. Depending on the sensitivity imposed to the algorithm, it might result in many false positives [138].

Used in combination, CUSUM and EWMA have demonstrated an acceptable performance in detecting different shifts from the process mean which could help account for intensive attacks [138]. The EWMA chart plots the moving averages of data and assigns weights that decrease exponentially from the present to the past. As a result, the average values are influenced more by recent data points than older points. Equation 4.7 conveys this, illustrating how the relevant information for detecting a change lies in the value of the log-likelihood ratio and its current minimum value.

Where:

- $\bar{\mu}_t$: is the estimated mean rate
- β: is the EWMA factor
- $\bar{\mu}_{t-1}$: is the mean rate estimated from measures prior to t
- F_t: is the number of packets in the specified time interval

$$\bar{\mu}_t = \beta\bar{\mu}_{t-1} + (1-\beta)F_t, \tag{4.7}$$

If $F_t \geq (\propto +1)\, \bar{\mu}_{t-1}$ then the alarm is signalled at time t where $(\propto > 0)$ is a parameter that indicates the percentage above the mean value that considers the indication of anomalous behaviour. However, including this as an indicator of anomalous actions would include a high number of false positives as indicated by Siris et al. [116], whereby a modification that can improve its performance is to signal an alarm after a minimum number of consecutive violations of the threshold.

Equation 4.7 is adapted to take into consideration the tuning parameters of the CUSUM algorithm, which are the amplitude percentage parameter $\propto$, the alarm threshold h, the EWMA factor β, and the length of the time interval over which traffic measurements are taken [116].

It returns a vector of the EWMA of an input vector g_t, computing the known variance from the input vector and the cumulative sum over the input:

$$g_t = [g_{t-1} + \frac{\alpha\bar{\mu}_{t-1}}{\sigma^2}\left(F_t - \bar{\mu}_{t-1} - \frac{\alpha\bar{\mu}_{t-1}}{2}\right)]^+ \qquad (4.8)$$

Where:

- g_t: is the input vector
- $\propto$: is an amplitude percentage parameter, which intuitively corresponds to the most probable percentage of increase of the mean rate after a change (attack) has occurred.
- $\bar{\mu}_t$: is an estimate of the mean rate at time t which is computed using an EWMA as in equation 4.7.
- σ: is the standard deviation
- F_t: is the number of traffic packets in the n^{th} time interval

One of the key research challenges is owing to the size and dynamic nature of the Cloud environment, namely the architecture and structure, which poses problems for the application of existing solutions. CUSUM and EWMA in combination are appropriate threshold calculation and adaptive monitoring approaches for detecting small shifts within a sequential quantitative observation and its application for DDoS attacks is suitable as it constantly adapts its value based on continuous measurements and remains insensitive to the normality assumption [137]. This combination of algorithms helps the system adapt to the behaviour of the underlying infrastructure and have more accurate detections, in comparison to a strict threshold based scheme.

4.3.1 Monitoring Nodes

MNs try to deal with issues on a local level and communicate with their neighbouring nodes regarding systems states and signatures, via the use of the gossip protocol as its premise is to manage a large, unreliable resource pool without any central coordinator. The exchange of information throughout the network is rampant, but messages and alerts need to be delivered with high probability. These nodes contain a black and white list, and a local grey list. A pre-alarm is triggered when a predefined threshold is violated, in utilisation with CUSUM and EWMA as detailed in Chapter 4.3.

If there are M nodes and each node gossips to $\log(\mathrm{M}) + \mathrm{c}$ other nodes on average, the probability that everyone gets the message converges to $\mathrm{e}^{-\mathrm{e}^{-\mathrm{c}}}$, which is very close to 1 without considering failures – where c is a fixed parameter determined by the number of nodes, and structure of the network topology, within the domain. The number of communications necessary to spread global workload information to all the mapping nodes respects $\log(\mathrm{M}) / \log(\log(\mathrm{M}))$, which shows that it takes at most a logarithmic number of steps to reach every mapping node [139].

With the gossip protocol [140] each node chooses a neighbour randomly to transmit a message. Suppose that the node i chooses node j with probability P_{ij}, where zero probability implies that the two nodes are not within their communication range, and are therefore not neighbours. It can be shown that the communication delay of disseminating a message from a single node in the network to all M nodes in the network is $\mathrm{O}(\log \mathrm{M}/\emptyset)$, where $\emptyset$ is the conductance of the network, given by:

$$\emptyset = \min_{\mathrm{S}:|\mathrm{S}| \leq \mathrm{M}/2} \frac{\sum_{\mathrm{i} \in \mathrm{S}, \mathrm{j} \in \mathrm{S}^{\mathrm{c}}} \mathrm{P}_{\mathrm{ij}}}{|\mathrm{S}|} \tag{4.9}$$

Where:

- $\emptyset$*:* conductance of the network

- *S:* energy required to transmit the data
- *M:* nodes in the network
- i: network node
- j: network node
- P_{ij}: probability that the two nodes are within their communication range

The gossip protocol can have widespread broadcasting in an asynchronous style, but the communication is quite random [119]. Using this approach however there is a need to ensure that intrusions are detected fast and that information can disseminate throughout the federation quickly. Using this approach to communicate pre alarms through the network would cause an increase in network overhead, and a basic role of a monitoring entity is to be non-intrusive. In order to deal with the scale of the Cloud environment, introducing hierarchy via the role of the SN and local views is a more suitable approach.

The role of the SN, detailed in Chapter 4.3.2, to observe the pre-alarms generated by the MNs could compliment the gossip based approach and improve the speed of message coverage. Broadcasting the pre-alarms in this way would reduce the network throughput and associated latency as, if every node in the network was to receive and forward the alarm packet, this would consume a lot of energy. Specifically, a pre-alarm is sent when the observed value is compared with a global threshold, such as using CUSUM for traffic volume dynamics – CUSUM is a widely used anomaly detection algorithm that has its foundations in change point detection [116].

In particular, an alarm is signalled when the accumulated volume of measurements g_t exceeds an aggregate volume threshold. When a pre-alarm is sent, monitoring nodes add it to their local grey list.

Let a monitored value x_i on the MN i at time t be:

$$x_i(t), i \in [1, M], \tag{4.10}$$

Where M is the number of MNs involved in the monitoring task.

Given the global threshold T, the state at time t can be considered to be abnormal and trigger a state alert if:

$$\sum_{i=1}^{M} x_i\ (t) > T, \tag{4.11}$$

which is referred to as a global violation [141].

T can be decomposed into a set of local thresholds T_i for each MN i such that:

$$\sum_{i=1}^{M} T_i\ \leq T. \tag{4.12}$$

As a result, as long as the monitored value at any node is lower or equal to its local threshold, the global threshold cannot be exceeded, i.e.

$$x_i(t) \leq T_i\,, \forall i\ \in [1, M] \tag{4.13}$$

This is as:

$$\sum_{i=1}^{M} x_i(t) \leq\ \sum_{i=1}^{M} T_i\ \leq T. \tag{4.14}$$

In this case, MNs do not need to report their local values to the SN. In order to report a local violation, MN i sends a message to the SN with the value $x_i(t)$. The value of T_i is decided depending upon the local CSP but is calculated using the CUSUM algorithm where threshold calculation is detailed in Chapter 4.3. Each MN is independent and treated equally to ensure fair and accurate monitoring, and improve the reliability and efficiency of the monitoring schema. Each entity has a level of trust and is acting in a trustworthy manner – this is established in the assumptions identified in Chapter 5.1.

There is likely a case where $\sum_{i=1}^{M} x_i\ (t) < T$ but a particular $x_i\ (t)$ exceeds T_i so it is important to ensure the monitoring infrastructure can adapt to changes within the underlying infrastructure to gather local and global information that is accurate to the domain under observation – this is achieved using CUSUM and EWMA algorithms detailed in Chapter 4.3.

Figure 4-2 shows a flowchart conveying what operations the MN executes – beginning from initial start-up and showing the flow of information for detecting anomalous actions within the network.

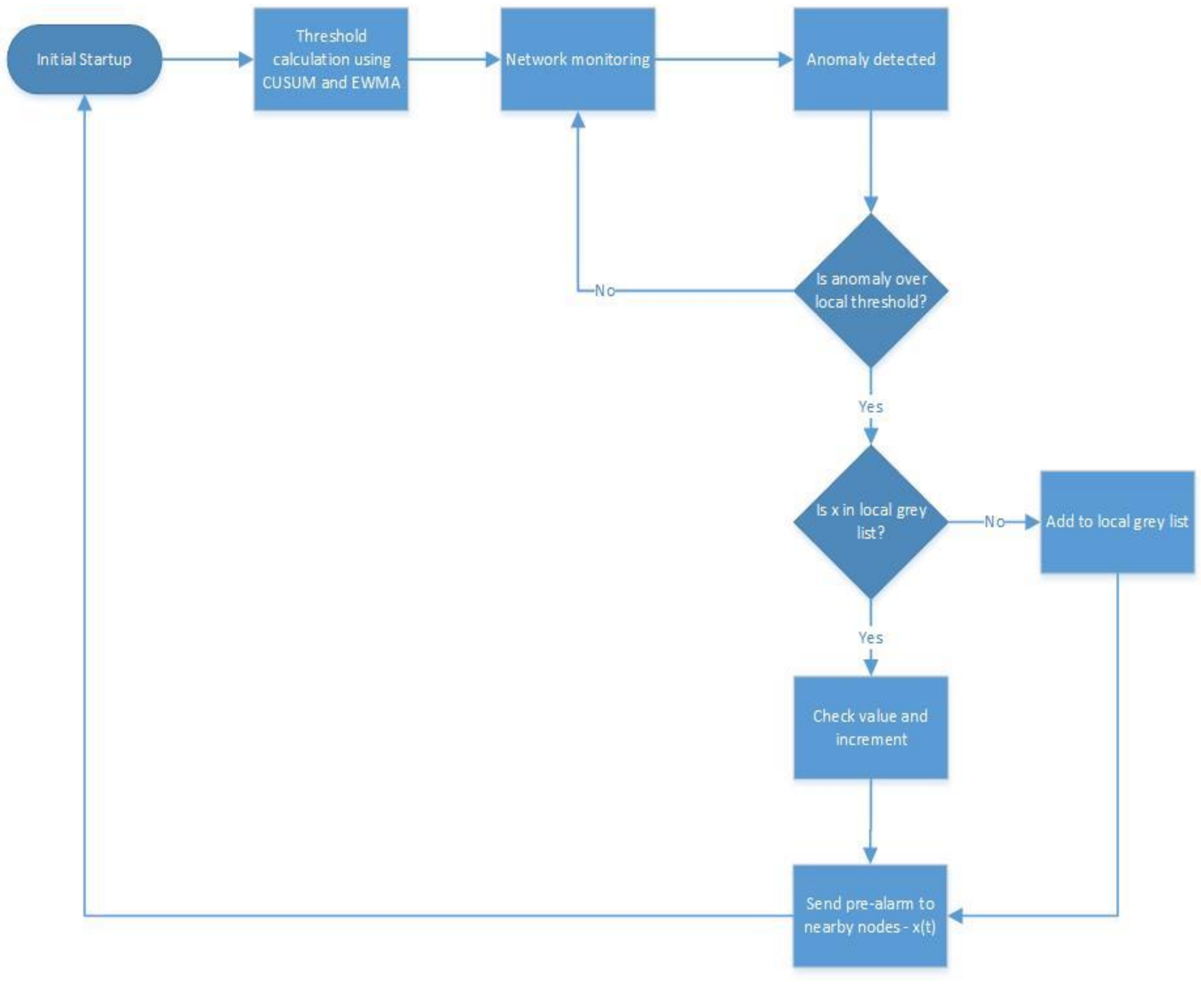

Initial Startup
Threshold calculation using CUSUM and EWMA
Network monitoring
Anomaly detected
Is anomaly over local threshold?
No
Yes
Is x in local grey list?
No
Add to local grey list
Yes
Check value and increment
Send pre-alarm to nearby nodes - x(t)

Figure 4-2: Monitoring node flow chart

After adding to the local grey list, the MNs role loop back to the initial start-up in order to adjust threshold algorithms to deal with the dynamic nature of the underlying Cloud infrastructure. In addition, the MN deals with communications from other MNs, as a local node may send a pre-alarm for them to add to their local grey list - this can occur during any stage at run-time. For values that are already present in the local grey list the score is incremented due to frequency of occurrence.

4.3.2 Super Node

A SN has a parent/child relationship with MNs under its management. The SN effectively communicates upstream with the C2 to query any suspicious actions. In order to address the problem of scale and network latency with centralised monitoring schemes, a hierarchical scheme with clearly defined roles and communication structure is utilised. The hierarchy of communication means network latency and throughput is reduced, and communication occurs only when essential, or when thresholds are violated. The SN, based on the number of MNs in its subnet, observes the generated alarms. When the pre-alarm count is more than or equal to the threshold based on the number of MNs, a belief is formed that there is an attack.

Based on x_i at time t, a belief is applied to the value:

$$\text{Bel(ID, timestamp, BPA)} \qquad (4.15)$$

Where *Bel* contains all the information associated to the observed occurrence, *ID* is an identifier unique to x_i, *timestamp* corresponds to time t, and the BPA (Basic Probability Assignment) which is a positive number between 0 and 1, and exists in the form of probability. For example, a belief value of 0.8 corresponds to the belief that there is an attack, and the corresponding hypotheses set information is detailed in Table 4.7 in Chapter 4.5. The SN then sends this belief to the C2, who queries the broker. Beliefs can be formed based on the belief function established and the associated hypothesis set.

Depending on established thresholds the belief generations could be associated with the conditions below:

- $\mathrm{b_a}$: the belief that there is an attack – 0.7-1.0
- $\mathrm{b_n}$: the belief that there is not an attack – 0-0.3
- $\mathrm{b_{na}}$: the belief expressing an ambiguity between attack and no attack – 0.4-0.6

Figure 4-3 shows a flowchart conveying what operations the MN executes and how the SN interacts and performs its tasks.

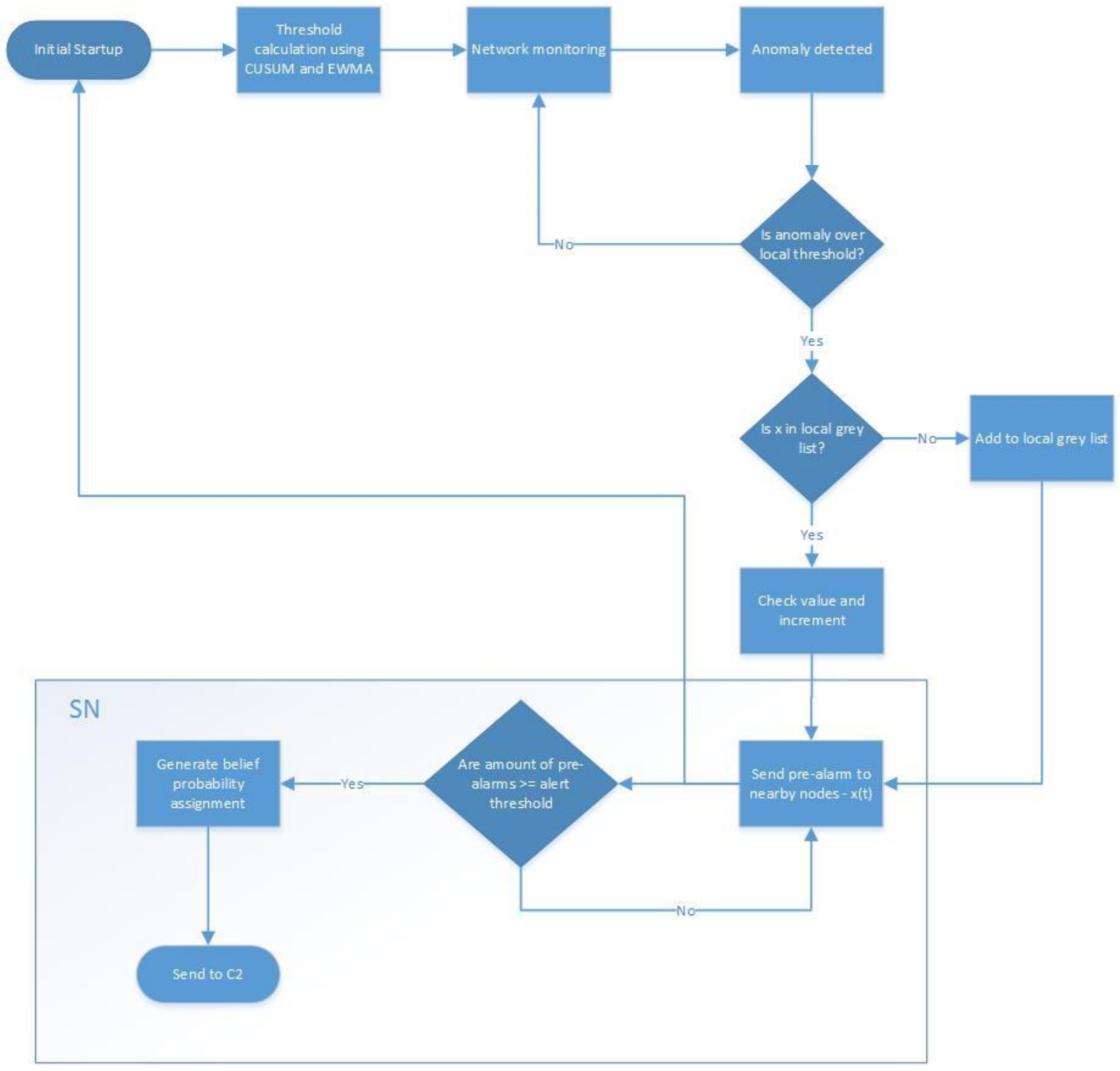
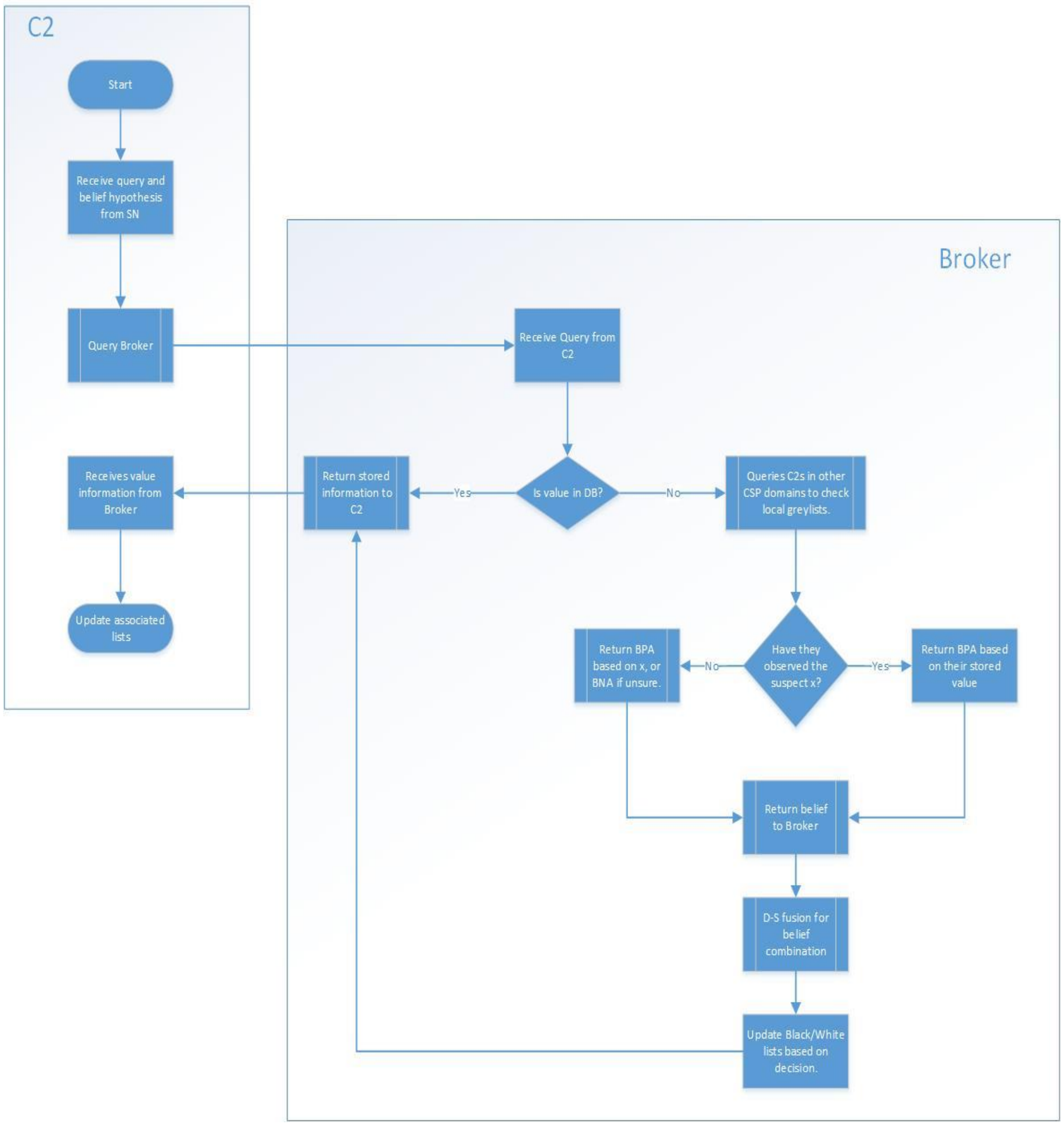

Initial Startup
Threshold calculation using CUSUM and EWMA
Network monitoring
Anomaly detected
Is anomaly over local threshold?
No
Yes
Is x in local grey list?
No
Add to local grey list
Yes
Check value and increment
SN
Generate belief probability assignment
Yes
Are amount of pre-alarms >= alert threshold
Send pre-alarm to nearby nodes - x(t)
No
Send to C2

Figure 4-3: SN flow chart

If the amount of pre-alarms is not more than or equal to the alert threshold the SN goes back to observing the MN alerts. However, if the amount of pre-alarms are more than or equal to the defined alert threshold then BPA occurs – which is forwarded to the C2.

4.3.3 C2

The C2 is located within its designated domain and is a domain management node. The key services that the C2 provides are the management of SNs and MNs, responses to attacks detected and reported by SNs and/or the broker, and cooperation with adjacent domains when polled. When a threat is detected in its domain, a belief is formed that an attack is underway. The C2 queries the broker about the generated belief, to see if it is legitimate or not (as quite possibly information on it could be known but the information might not have been propagated yet). C2s possess black lists and white lists comprised of blocked IP addresses, and local grey lists provided by the SN and MN which contain ambiguous observations. Figure 4-4 is a flowchart illustrating how components and operations between the C2 and broker interact.



C2
Start
Receive query and belief hypothesis from SN
Query Broker
Receives value information from Broker
Update associated lists
Broker
Receive Query from C2
Is value in DB?
Yes
No
Return stored information to C2
Queries C2s in other CSP domains to check local greylists.
Have they observed the suspect x?
No
Yes
Return BPA based on x, or BNA if unsure.
Return BPA based on their stored value
Return belief to Broker
D-S fusion for belief combination
Update Black/White lists based on decision.

Figure 4-4: C2 and Broker flow chart

The C2 receives a query and associated belief hypothesis from the SN, and queries the broker with this information. The broker checks if it possess information on the value in its databases and returns the stored information to the C2 if present. Otherwise, the broker queries C2s in other domains to check their grey lists and return a BPA based on their stored value, return a BPA based on the queried value, or returns b_{na} which is the belief expressing an ambiguity between attack and no attack.

4.3.4 Cloud broker

There is a need for secure communication in a Cloud-based collaborative environment. The question is then how to secure the intra-Cloud and inter-Cloud communications between the collaborative services deployed in the Clouds. Management from a Cloud broker is important as they provide the interoperability and portability of applications across multiple Cloud environments. A Cloud broker may enhance, combine, or integrate services from multiple CSPs in order to create new and value-added service offerings [135].

Currently, Cloud brokers offer tools to manage applications across multiple CSPs. In the future, Cloud brokers will offer services based on their knowledge of the CSPs' infrastructure [142]. In our solution, this information is used to offer Cloud CSPs "Security as a Service", where the broker uses the knowledge base of Cloud attacks and behavioural profiles to identify threshold violations. The broker with its database server of signatures contains a black list, white list, and grey list.

The black list contains signatures of traffic and behaviour blocked from the Cloud federation; the white list contains signatures of permitted actions; and the grey list is a local list to each C2 that is used to store ambiguous observations with frequency of occurrence – the grey list is a function mapping signatures to beliefs. Publish-subscribe is used to propagate this information from the broker to adjacent C2s in order to reduce overhead and network latency.

Publish-subscribe models have been widely for tasks such as event notification and mobility support services. Each subscriber maintains a local database that contains the latest values of each topic instance and can locally perform query operations without generating any network traffic. Using a data-centric approach decouples data sources and consumers: publishers and subscribers share the same data model for data (topics) and can build distributed architectures where the local application logic is decoupled from the data model. In the context of collaborative intrusion detection, when a participant detects a possible attack in its monitored sub-network, it generates an alert, which is reported to the higher entity which is known as subscription [99].

When queried with a belief value about suspect actions, the broker invokes a global poll procedure when a decision cannot be made. The broker then queries the C2s in adjacent domains, and asks them to generate their own beliefs based on the presented information. They check their local grey lists to see if they have encountered the suspect actions previously. Each C2 generates their own belief, and the broker uses D-S to fuse the different beliefs and to create a system wide decision.

This collaborative approach and sharing of information in turn can improve resilience to attack, as within Cloud federations, system vulnerability V_s is increased due to the interdependent services and multi tenants. Additionally, the data and services stored may have high availability requirements. Within a system s, vulnerability V is the maximum vulnerability level over a set of scenarios represented by D:

$$V_s = max\big(V(D)\big) \tag{4.16}$$

Whereby, scenario vulnerability is defined by:

$$V(D) = \{V(d_1), V(d_2) \dots \dots, V(d_K)\} \tag{4.17}$$

Where K is the number of scenarios to be evaluated; and d is an individual scenario.

The vulnerability of a scenario $V(d)$ through an access point is evaluated to determine its potential damage using the weighted sum of the potential damages over a set u. Liu [143] describes scenario vulnerability as:

$$V(d) = \sum_{j \in u} \pi_j \gamma_j \tag{4.18}$$

Here, π_j is the steady state probability that the system is attacked through a specific access point j which is linked to the system [143]. The damage factor γ_j represents the level of damage on the system if the service is removed, and in the case of critical infrastructure data, this would be availability of data and services. The data range for this can differ depending on the scenario but for probability based approaches a value between 0 and 1, where 0 represents no vulnerability and 1 represents catastrophic failure, but these can be established with the use of risk logs or risk graph modelling. Within Cloud federations the benefits of resource balancing and sharing outweighs the potential vulnerabilities, but the concern is that a vulnerability could propagate throughout a federation based on the many access points identified in equation 4.11.

The Cloud broker is an elastic scaling mechanism, but for the purposes of diagrams and testing there is one broker within explanations. In the real world one broker would be a single point of failure. As such, there would be multiple brokers which would sit behind a load balancer, sharing a repository of information, and using the resources of the Cloud to adapt to workload changes by provisioning and de-provisioning resources in an autonomic manner.

4.4 Collaboration

The size of the IDS is limited by its fixed number of components. As the number of monitored entities grows, the analysis components will need more computing and storage resources to keep up with the load. A distributed IDS can scale to a larger number of hosts by adding components as needed. Scalability may be limited by the need to communicate between the components, and by the existence of central coordination components. Distributed collaboration among heterogeneous components within and across independent domains has been indicated in recent literature [48], [63], [99], [113], [134], [144]. Collaboration among agents in agent-based systems is commonplace, but applying this information exchange could lead to increased situational awareness in interconnected infrastructures.

The cooperation of threat knowledge, both known attacks and unknown threats, among peers within the enterprise network or with other CSPs will contribute to better incident detection and prevention, enhancing Cloud security and providing more effective incident response [145]. Information sharing in this approach is autonomous, which is conceived to be an important aspect of this approach. Collaboration among CSPs in the federated Cloud could offer holistic security to those CSPs in this agreement.

Decision making processes involve differing hierarchical structures, and varying roles for the members within. The organisation of such structures varies widely, but each has been considered greatly for our solution. The democratic-style and centralised-style both have merits, as such has been determined that a combination of both would be the most beneficial and meet the design requirements in Chapter 3.5. In a democratic-centralised-style structure there is a leader member who relies upon the other members for information when making a decision.

This hierarchical communication schema is conveyed in Figure 4-5, which outlines how detection and correlation elements would communicate. Within such an approach, data is distributed, which may make it more difficult to adjust to global changes in behaviour, whereas local changes are easier to detect. The subordinate organisation members have limited power in the decision-making process but interact with each other fully. Information exchange is more structured and timely following this approach.

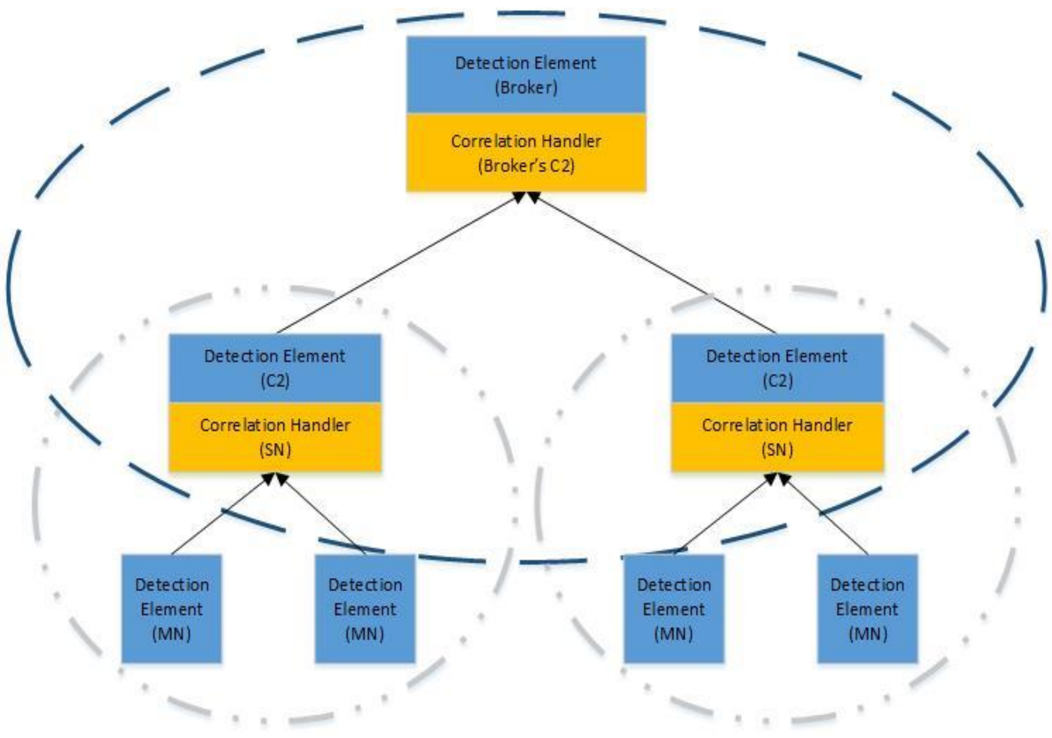


Figure 4-4: Hierarchical communication schema

Figure 4-5 shows how detection elements and correlation handlers work with regards to the entities presented within the Service-based collaborative intrusion detection architecture. Figures 4-2 – 4-5 show the execution of the workflow and present the flow of information within the structural design. Detection elements consist of several detection components which monitor their own sub-network or host individually and generate low level alerts. Then the correlation handler transforms the low level alerts into a high level report of an attack [86]. The biggest challenge in employing a sampling algorithm on a given network is scalability. For intrusion detection algorithms, sampling costs are of paramount importance. Within our approach, the communication architecture takes this into consideration and uses levels of hierarchy for information sharing conveyed in Figure 4-5. The generation of alerts has to exceed an expected threshold before this information is passed upwards to a superior attribute in the scheme.

Figure 4-6 visualises the levels of communication occurring between each entity in our solution. The hierarchical structure within our solution conveys how key entities exchange information with one another, and how the information flows. For illustration purposes, a small scale of entities is used but there can be more than one C2 within each domain — the relationship between SNs and MNs is as described in Chapters 4.3.1 and 4.3.2. The solution can scale considerably and based on the democratic-centralised communication style, entities can be added with ease to their role.

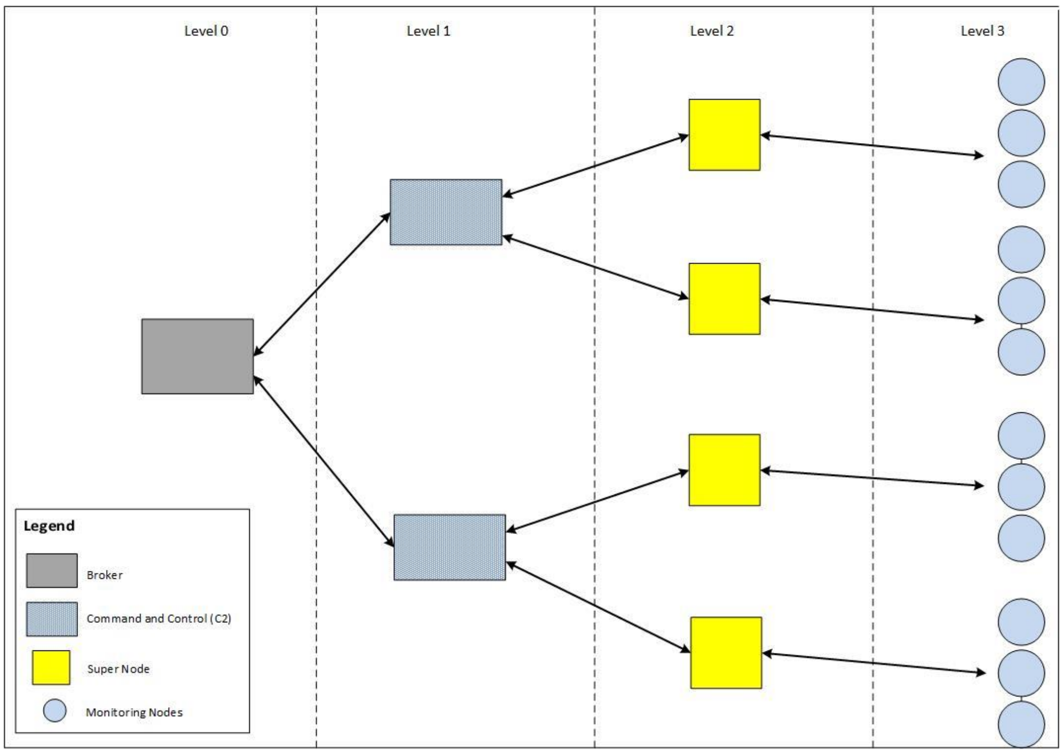


Figure 4-5: Levels of communication

4.5 Belief generation using Dempster-Shafer theory of evidence

For collaborative intrusion detection, an extended D-S fusion algorithm is utilised. Via the broker, D-S executes as a main fusion node, an entity with the role to collect and fuse the information provided by the monitors, taking the final decision regarding a possible attack. In the decision making process, the uncertainty existing in the network often leads to the failure of intrusion detection or low detection rate. The D-S theory of evidence in data fusion has solved the problem of how to analyse the uncertainty in a quantitative way [146]**.**

The key attributes of D-S theory of evidence include:

The frame of discernment:

The frame of discernment is a completable set which describes all the sets in the hypothesis space. Generally, the frame is denoted as θ, which is similar to a state space in probability [129]. The elements in the frame must be mutually exclusive, and while the number of the elements is n, the space will be 2^{n}.

Basic probability assignment:

BPA is a positive number between 0 and 1, and exists in the form of probability. For a particular event A in 2^{θ} (which is the hypothesis space), $m(A)$ characterises the degree of BPA supporting or refuting evidence.

Belief function:

For $2^{\theta} \in [0,1]$,

Since $m(A)$ measures the belief that one commits to the set A exactly and not to any proper subset of A, to obtain the total belief committed to A, one must add to $m(A)$ the quantities $m(B)$ for all proper subsets B of A. Given a mass function m, a belief measure and a plausibility measure can be uniquely determined to obtain a total BPA committed to a particular proposition. The belief function Bel is defined as [128]:

$$\mathrm{Bel}(\mathrm{A}) = \sum_{\mathrm{B} \subseteq \mathrm{A}} \mathrm{m}(\mathrm{B}) \tag{4.19}$$

where all $B \subseteq A$; $A \in 2^{\theta}$. $Bel(A)$ indicates the total evidence or belief that the element belongs to the set A ($x \in A$) as well as to the various subsets of A.

Plausibility function:

The plausibility Pl(A) is the sum of all the masses of the sets *B* that intersect the set of interest A:

$$pl(A) = \sum_{B|B\cap A\neq\emptyset} m(B) \tag{4.20}$$

The belief function and plausibility function are related by $\mathrm{Bel(A)} \leq \mathrm{Pl(A)}$. This is as D-S allows for belief about propositions to be represented as intervals, bounded by two values, belief (or support) and plausibility. Then calling $[\mathrm{Bel(A)}, \mathrm{Pl(A)}]$ the Belief Range. $Pl(A)$ is the mass of *A* and the mass of all sets that intersect with *A*, i.e. those that could transfer their mass to *A* or a subset of *A* [108].

Belief generation:

For pre-emptive warning, beliefs are generated and assigned to all subsets of possible outcomes based on the established CUSUM thresholds defined using equation 4.8.

Based on an occurrence and monitored value x_i at time t, a belief is assigned with the information:

$$\text{Bel(ID, timestamp, BPA)} \tag{4.21}$$

Where:

- *Bel:* contains all the information associated to the observed occurrence
- *ID:* is a unique identifier to x_i
- *timestamp:* corresponds to time t
- *BPA:* is a value from 0 to 1 - the associated hypothesis sets are within the Table 4.2.

Mass, Belief and Plausibility based on the equations 4.19 and 4.20 are presented. Belief in a hypothesis is comprised by the sum of the masses of all sets enclosed by it. A belief measures the strength of the evidence in favour of a proposition p, ranging from 0 (indicating no evidence) to 1 (denoting certainty). Plausibility is 1 minus the sum of the masses of all sets whose intersection with the hypothesis is empty. Or, it can be obtained as the sum of the masses of all sets whose intersection with the hypothesis is not empty. The universal hypothesis "Either" will always have a belief and plausibility assignment of 1.0 (having 100% belief and plausibility) because this value is always true. For simulation purposes in Chapter 5, threat values associated with each belief value are included also, and these reflect the range from 0-100 correlating with belief values between 0 and 1.

Table 4.2: Hypothesis sets of values for Belief values between 0 and 1.

Threat Value	Hypothesis	Mass	Belief	Plausibility	Belief Value
100	Attack	1.0	1.0	1.0	1.0
	No Attack	0.0	0.0	0.0	
	Either	0.0	1.0	1.0	
90-99	Attack	0.9	0.9	0.95	0.9
	No Attack	0.05	0.05	0.1	
	Either	0.05	1.0	1.0	
80-89	Attack	0.8	0.8	0.9	0.8
	No Attack	0.1	0.1	0.2	
	Either	0.1	1.0	1.0	
70-79	Attack	0.7	0.7	0.8	0.7
	No Attack	0.2	0.2	0.3	
	Either	0.1	1.0	1.0	
60-69	Attack	0.6	0.6	0.7	0.6
	No Attack	0.3	0.3	0.4	
	Either	0.1	1.0	1.0	
50-59	Attack	0.5	0.5	0.7	0.5
	No Attack	0.3	0.3	0.5	
	Either	0.2	1.0	1.0	

40-49	Attack	0.4	0.4	0.6	0.4
	No Attack	0.4	0.4	0.6	
	Either	0.2	1.0	1.0	
30-39	Attack	0.3	0.3	0.4	0.3
	No Attack	0.6	0.6	0.7	
	Either	0.1	1.0	1.0	
20-29	Attack	0.2	0.2	0.7	0.2
	No Attack	0.3	0.3	0.8	
	Either	0.5	1.0	1.0	
10-19	Attack	0.1	0.1	0.4	0.1
	No Attack	0.6	0.6	0.9	
	Either	0.3	1.0	1.0	
0-9	Attack	0.0	0.0	0.5	0.0
	No Attack	0.5	0.5	0.0	
	Either	0.5	1.0	1.0	

Using equations 4.1 – 4.8, threshold calculation algorithms such as CUSUM and EWMA are used to determine when a local threshold has been violated. As described Chapter in 4.3.2 – 4.3.4, belief generation occurs and this information is propagated to the relevant monitoring entities.

Based on the assigned threat score, the associated hypothesis sets are associated. Within an intrusion detection based system that monitors the underlying virtualised infrastructure, there are four clear variables that would involve detecting violations. These variables would have a BPA of { }, {attack}, {no attack}, and {attack, no attack}. { } represents an empty subset with a value of 0, that corresponds to "no solution". Whereas {attack, no attack} represents uncertainty, i.e. it could be either.

The monitors, based on the local detection algorithms, produce a single belief for each focal element and these would identify violations of local thresholds:

- $\mathrm{b_a}$: {attack}- threat value in the range of 0.7-1.0
- $\mathrm{b_n}$: {no attack}- threat value in the range of 0-0.3
- $\mathrm{b_{na}}$: {attack, no attack}- threat value in the range of 0.4-0.6

D-S combination rule:

D-S utilises orthogonal sum to combine the evidences, where $\oplus$ is the combination operator [129]. The belief functions are defined, describing the belief in a hypothesis A, as $\mathrm{Bel_1(A)}, \mathrm{Bel_2(A)}$; then the belief function after the combination is defined as:

$$\mathrm{Bel(A)} = \mathrm{Bel_1(A)} \oplus \mathrm{Bel_2(A)} \tag{4.22}$$

The mass function after the combination can be formulated as:

$$\mathrm{m(A)} = \mathrm{DSK^{-1}}.\textstyle\sum_{\mathrm{A_i \cap B_i = A}} \mathrm{m_1(A_i)\, m_2(B_j)} \tag{4.23}$$

Here, DSK is the Orthogonal Coefficient, and it is defined as:

$$\mathrm{DSK} = \textstyle\sum_{\mathrm{A_i \cap B_i \neq \emptyset}} \mathrm{m_1(A_i)\, m_2(B_j)} \tag{4.24}$$

D-S combines the beliefs expressed by monitors to produce a single combined belief that is finally compared with a set accumulative sum q of beliefs. If the combined belief is greater than q, an alarm is raised as this would indicate an attack [120]. If the output of a local detection algorithm is close to h, where h is the detection threshold, b_{na} increases to express a higher belief on the uncertainty of an attack or normal operation.

D-S proposes the concepts: belief and plausibility, which can aid the theory to analyse the "incomplete" or "missing" information quantitatively. In this way, the inference can guarantee the accuracy of the decision. Evidence is fused to reach the goal that can determine the current state of the network. A further step could include fitting time distribution curves, and this can improve detection engine efficiency and applicability [129]. D-S's theory of evidence can be regarded as the expansion of Bayesian inference, whereas Bayes requires priori knowledge as the foundation of inference.

Fusing beliefs based on equation 4.22, with reference to Table 4.2 conveys how belief, mass, plausibility and threat values could aid collaborative intrusion detection.

Scaling equation 4.22 to consider more beliefs would be represented by:

$$\mathrm{Bel}(\mathrm{A}) = \mathrm{Bel}_1(\mathrm{A}) \oplus \mathrm{Bel}_2(\mathrm{A}) \oplus \mathrm{Bel}_3(\mathrm{A}) \tag{4.25}$$

Examples of the scaling of this algorithm to perform equation 4.25 are as below:

Take three monitor values - $m_{1,}$ m_2 and m_3.

The belief that the proposition is true for state Attack which is 0.8 and the values for m_1 No Attack = 0.1 with m_1 Either = 0.1

Assessments m_2 and m_3 have the belief values with m_2 Attack = 0.6, m_2 No Attack = 0.3 and m_2({Either}) = 0.1

m_3 associated values include Attack = 0.0, m_3 No Attack = 0.5 with m_3 Either = 0.5

Based on equation 4.23, the mass function after the combination can be formulated as:

$m_{1,2,3}$(A) = (1/1-K) m_1({Either})m_2({Either}) m_3({Either}) with:

DSK = m_1({Attack}) m_2({No Attack}) + m_1({No Attack}) m_2({Attack}) + m_1({Attack}) m_3({No Attack}) + m_1({No Attack}) m_3({Attack}) + m_2({Attack}) m_3({No Attack}) + m_2({No Attack}) m_3({Attack}) = 0.8*0.3 + 0.1*0.6 + 0.8*0.5 + 0.1*0.0 + 0.6*0.5 + 0.3*0.0 = 1

So $m_{1,2,3}$(A) = (1/1-1) *0.1*0.1*0.5 = **n/a - cannot divide by 0**

A further example would be: three monitor values - m_1, m_2 and m_3.

The belief that the proposition is true for state Attack which is 0.5 i.e. m_1 Attack = 0.5 and similarly m_1 No Attack = 0.1 with m_1 Either = 0.1

Assessments m_2 and m_3 have the belief values Attack = 0.6, m_2 No Attack = 0.3 with m_2 Either = 0.1

m_3 associated values include Attack = 0.9, m_3 No Attack = 0.05 with m_3 Either = 0.05

Based on equation 4.23, the mass function after the combination can be formulated as:

$m_{1,2,3}$(A) = (1/1-K) m_1({Either})m_2({Either}) m_3({Either}) with:

DSK = m_1({Attack}) m_2({No Attack}) + m_1({No Attack}) m_2({Attack}) + m_1({Attack}) m_3({No Attack}) + m_1({No Attack}) m_3({Attack}) + m_2({Attack}) m_3({No Attack}) + m_2({No Attack}) m_3({Attack}) = 0.5*0.3 + 0.1*0.6 + 0.5*0.05 + 0.1*0.9 + 0.6*0.05 + 0.3*0.9 = 0.63

So $m_{1,2,3}$(A) = (1/1-0.63) *0.1*0.1*0.05 = **0.001351**

4.6 Extended D-S theory of evidence fusion process

Assume two BPAs m^a and m^b represent the beliefs about values of a state within a specific frame θ. The use of the orthogonal coefficient in Equation 4.23 and normalisation in Equation 4.24 conveys that Dempster's rule is mathematically possible only if m^a and m^b are not conflicting, i.e. if there is a focal element y of m^a and a focal element z of m^b satisfying the intersection ($\cap$) of the two sets, $(y \cap z) \neq \emptyset$, such that they have no elements in common.

Merging two belief masses with the conjunctive rule defined above produces a sub-additive basic belief assignment, meaning that the sum of belief masses on focal elements can be less than one, in which case it is assumed that the missing or complement belief mass gets assigned to the empty set. If desirable, the normality assumption m($\emptyset$) = 0 can be recovered by dividing each belief mass by a normalization coefficient [147].

This rule is associative, and the normalisation in D-S's rule redistributes conflicting belief masses to non-conflicting ones, and tends to eliminate any conflicting characteristics in the resulting belief mass distribution. This rule of combination can be applied to avoid this particular problem by allowing all conflicting belief masses to be allocated to the empty set. For rule combination, the order of the information in the aggregated evidences does not impact the result, however a non-associative combination is necessary for many cases.

Cooperative decision making is made after aggregating these evidences using D-S. The integration of the decisions coming from different IDSs has emerged as a technique that could strengthen the final decision. Sensor fusion can be defined as the process of collecting information from multiple and possibly heterogeneous sources and combining them to obtain a more descriptive, intuitive and meaningful result [53]. However, rule combination implies all evidences are trusted equally and that all sources have the same level of trust. However, in reality, the trust on different evidences may differ, which means various factors for each evidence should be considered.

Based on the identified limitations of the normalisation stage and weighted belief generation, D-S for autonomous sharing of information for detecting intrusions has application limitations as explained in Chapter 3.8. D-S when applied in an autonomous collaborative environment should apply a weight of confidence when the belief generation occurs. If there are three CSPs and one generates belief of an attack with a value of B_a 0.7, while the two remaining CSPs vote B_{na} 0.5, based on the D-S combination in Equation 4.25, the combined decision would be that there is no attack.

The issue here is that the CSP that generated the belief of attack can see there is clearly evidence of an attack but this value has been deemed legitimate. There should be a way to overrule the decision based on the strength of the associated trust or confidence value associated with the decision, or to deal with it locally but to warn globally of the pre-emptive threat.

This situation requires a two stage fusion process. Post belief generation processing is needed for application to this area to facilitate information exchange for defence. Via the inclusion of confidence values when belief generation occurs, the accuracy of decisions can be improved. Chapters 5 and 6 demonstrate how D-S can provide collaborative intrusion detection, however there may be cases where the decision may be inaccurate.

When fusing the beliefs the group consensus may be that it is not an attack, but it would be illogical for the domain of origin to not take action against the malicious occurrence. It could be an attack to the domain that generated the belief, but one that is not currently exceeding thresholds in other domains. While it is an issue in one, the others deem it not something currently applicable to them.

With the extended D-S theory of evidence fusion process, the CSP would deal with the threat locally if the domain is under attack, based on confidence values if there are conflicting decisions.

Let $P^{(a)} = [P_1^{(a)} \dots P_{Ka}^{(a)}]$ denote the K_a possible confidence values G associated with choosing $a \in A$ at time t_d.

The assigned confidence level $p \in P^{(a)}$ associated with deciding a after waiting for a period of $t_c = t_d + \tau$ is given as [148]:

$$p = P_1^{(a)} \text{ when } L(t_c) \in \left[G_{i-1,}^{(a)} G_i^{(a)}\right], \tag{4.26}$$

where $G_0^{(a)} = -\infty$ and $G_{Ka}^{(a)} = \infty$ for each $a \in A$, and the value τ is known as the inter-judgement time.

The remaining confidence parameters:

$$G^{(a)} = [G_1^{(a)} \dots G_{Ka-1}^{(a)}]$$

are chosen such that $G_{i-1} < G_i$ for each $i \in \{1 \dots \dots, K_a - 1\}$.

Adding a degree of confidence to each generated belief can improve the overall efficiency, and deal with the issue of conflicting beliefs during fusion. Pre-emptive warning in a Cloud federation could protect the local services of the CSP but proactively warn others of the potential threat.

If the fused decision is “No Attack” but the belief of origin has a high confidence value, then the domain of origin would take action against the suspect observation and send the belief value to the broker to store in its local Grey list. Should an adjacent CSP query the broker regarding the suspect IP in the future, the information from the origin CSP is present.

4.7 Summary

In this chapter, the design of the Service-based collaborative intrusion detection solution is presented; a collaborative intrusion detection framework that can detect and prevent intrusions in Cloud federations and/or collaborative domains in real-time via the autonomous sharing of information. This features a novel application of the D-S theory of evidence algorithm to detect intrusions and fuse generated beliefs for collaborative intrusion detection, and an extension of D-S. The extended D-S approach aims to tackle the issue of conflicting decisions within interconnected infrastructure, but used in combination with other computing techniques has the ability to improve decision making schemas.

Within the solution, the Cloud broker coordinates attack responses, both within the domain itself, and with other domains, and is facilitating inter-domain cooperation to improve upon the Cloud security contribution. This cooperation between CSPs ensures that the scalable defence required against DDoS attacks is carried out in an efficient way; aiming to improve the overall resilience of the interconnected infrastructure. The D-S theory of evidence is used to facilitate this autonomous sharing of information, and to fuse the generated beliefs to form a system-wide decision. The design of the solution is tailorable to different Cloud environments and has ability to scale considerably. Comparing the solution for distributed systems could involve adapting the hierarchical C2/MN/SN structure to the working of master/slave architecture model of distributed computing architecture where the master node has unidirectional control over one or more slave nodes. In this instance, the task(s) are distributed by the master node to the configured slaves and the results are returned to the master node. Distributed systems and Cloud computing environments slightly refer to different things, however the underlying concept between them is same. In the next chapter, an implementation of this solution is presented and a scenario to simulate events for a proof of concept.

Chapter 5

System Implementation

In the previous chapter, the design of the Service-based collaborative intrusion detection architecture was presented; comprising four tiers; Cloud broker, C2, SNs and MNs. Using our "Security as a Service" method, collaborative intrusion detection is possible in a federated Cloud environment. The system uses a Cloud broker to propagate information to the C2 entities in each CSP domain – this is in the form of Black lists and White lists. MNs are used to observe changes or suspicious activities in local domains, which are values stored in a grey list of ambiguous observations. In order to keep communication latency reduced, SNs monitor the alerts produced by MN and an alert is generated when a threshold based on the number of MNs within their subnet is triggered.

In order to evaluate the success of our Service-based collaborative intrusion detection framework, it is essential that a working implementation is used. This allows us to validate that our solutions meets the aims, objectives and requirements affiliated with this development. This chapter presents the implementation of a prototype of our Service-based collaborative intrusion detection framework for infrastructure services in a federated Cloud environment. The prototype enables the design presented in the previous chapter to be tested.

Collaborative security between CSPs in a Cloud federation can offer holistic security to those in this scheme. Information sharing in this scheme is automated which is an important aspect of the approach. For proof of concept, a lower number of entities are used but for future work these will be expanded and the solution scaled. Dividing the system into domains makes the system more scalable, and belief generation and sharing of threat information could be used as a warning of an imminent attack.

Assumptions made:

- Each CSP present in the federation has an SLA specifying the sharing of information – in this case it is of generated belief values, threat scores, and suspect IP addresses.
- Each entity has a level of trust and is not acting maliciously.

- Resource starvation and DDoS can propagate throughout the network; the performance issues are explored, not the financial implications.

5.1 Environment

Using Riverbed Modeler 18.0 [12], attributes of our Service-based collaborative intrusion detection architecture were implemented. Riverbed Modeler is a large and powerful software tool which enables the simulation of heterogeneous networks with various protocols. Riverbed Modeler consists of a high level user interface, which is constructed from C and C++ source code. One specific benefit of using this simulator is that all processes contain code to record performance metrics, which is favourable for observing both local and global statistics in our solution.

Riverbed Modeler allows multiple scenarios to be created and compared during parallel simulation, enabling comparisons between changes in the environment. As previously mentioned, our federated Cloud environment can be represented as interconnected domains which are dependent upon each other for operation. One of the key issues with Cloud federations is that a fault within the network can propagate throughout the federation, effecting key services of the CSPs present, so collaborative intrusion detection can benefit all parties within the federation.

The network topology for the solution was implemented using the Object Palette which has models, objects, and hardware ready for deployment; an empty scenario was created and populated with the required devices and objects. Within the network, CSPs are represented by interdependent domains named “Server Domain”, “Domain 1”, “Domain 2”, “Domain 3”, and “Broker”, which are all connected to “node_0” which represents an IP Cloud connection. Each of these CSPs is connected to sub-networks which contain the end users, routers and servers necessary for the network topology. Hierarchy in a network topology is achieved using subnets — which represent identical constructs in an actual network — allowing us to simulate end users of the CSP, and how malicious actions from one could affect the interconnected domains. These entities are connected via PPP_DS3 links which are used to connect nodes running IP protocols, as conveyed in Figure 5-1. The background is not a geographical representation but is a standard image used within the simulation package.

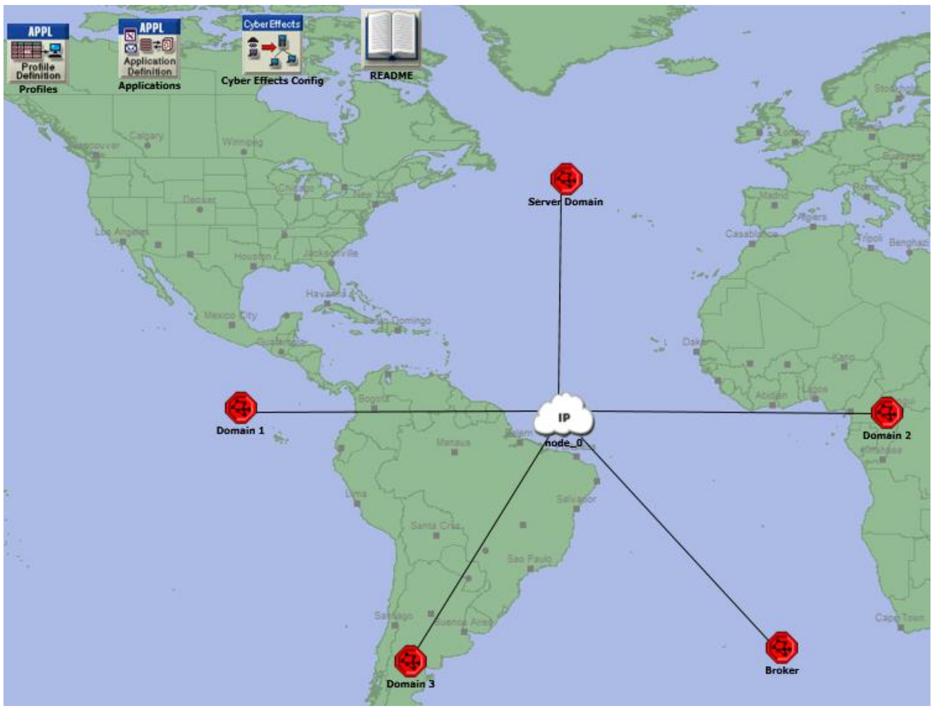
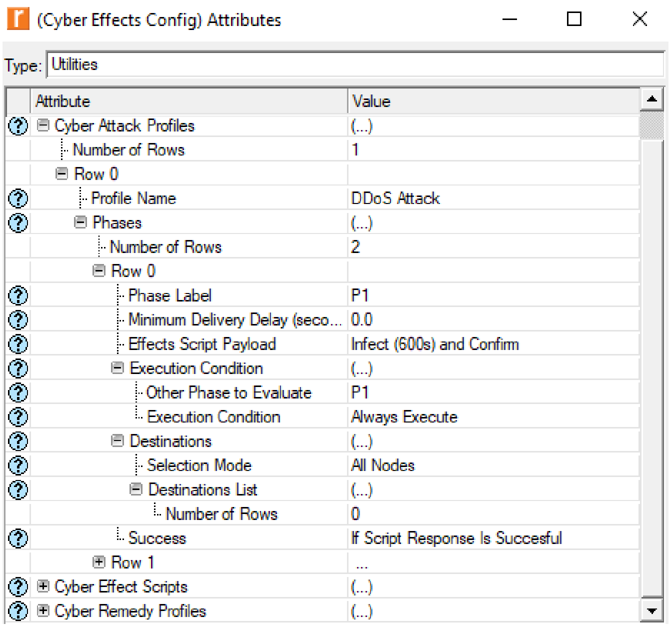


Figure 5-1: Overview of Cloud Federation topology

The Cyber Effects configuration module provided within Riverbed Modeler was enabled and tailored to our scenario in order to introduce cyber effects and attacks into our scenarios – this is shown in Figure 5-2. The DDoS attack profile was utilised, in addition to cyber effect scripts and remedy profiles.



(Cyber Effects Config) Attributes

Type: Utilities

Attribute	Value
⊟ Cyber Attack Profiles	(...)
Number of Rows	1
⊟ Row 0	
Profile Name	DDoS Attack
⊟ Phases	(...)
Number of Rows	2
⊟ Row 0	
Phase Label	P1
Minimum Delivery Delay (seco...	0.0
Effects Script Payload	Infect (600s) and Confirm
⊟ Execution Condition	(...)
Other Phase to Evaluate	P1
Execution Condition	Always Execute
⊟ Destinations	(...)
Selection Mode	All Nodes
⊟ Destinations List	(...)
Number of Rows	0
Success	If Script Response Is Succesful
⊞ Row 1	...
⊞ Cyber Effect Scripts	(...)
⊞ Cyber Remedy Profiles	(...)

Figure 5-2: Cyber effects configuration

The Cloud broker, within the Broker realm, is depicted as an Ethernet workstation with cyber effects remedy profiles configured, which is interconnected within the federation. Figure 5-3 illustrates this configuration. The broker sends information to the C2s in the adjacent domains when it is necessary to scan and clean the network.

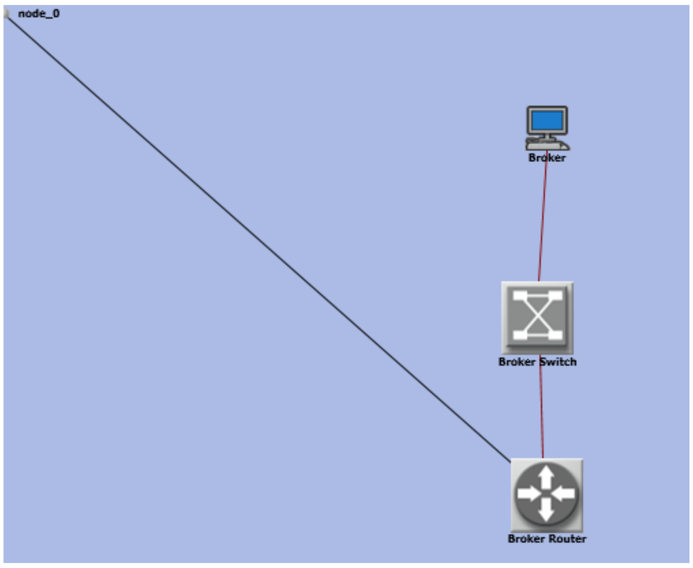


Figure 5-3: Cloud broker realm

Within each sub-network in the federation there are end users which can access the Cloud resources. Within each realm a C2 is present and they monitor the devices within their subnet. These devices represent legitimate users, and each of these devices has a range of infection probabilities when the simulation is running.

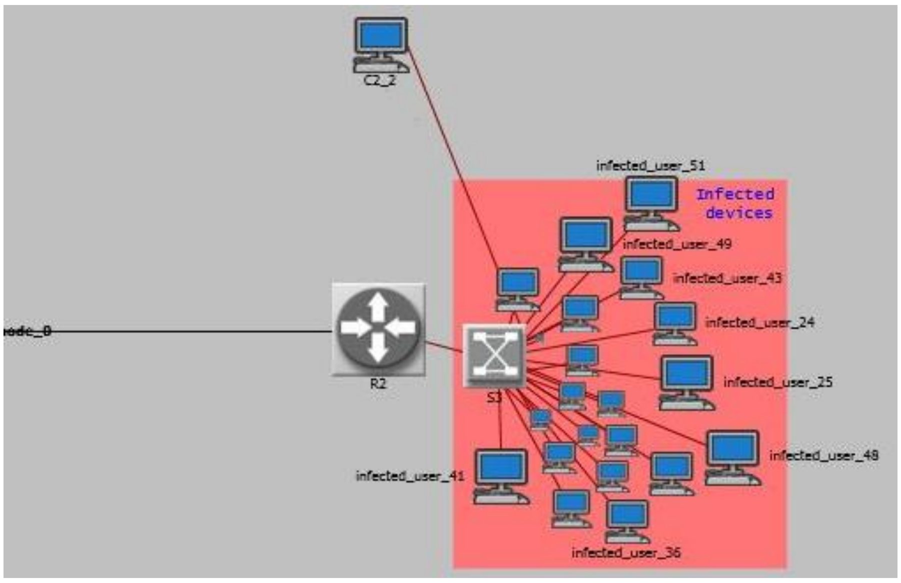


Figure 5-4: CSP 2 domain

These workstations are connected to a domain server and a gateway router, which are then connected to the Cloud IP in the main federation. These connections are illustrated in Figures 5-4 and 5-5.

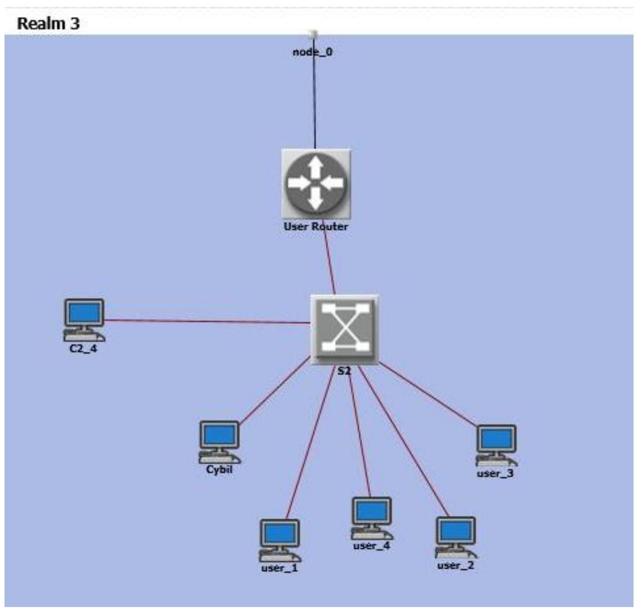


Figure 5-5: CSP 3 domain

Within the network an attacker entity was introduced. In order to determine the effects of a DDoS attack within the federation, the cascading effects an attacker within the network would have on the other devices present was modelled.

The attacker domain is illustrated in Figure 5-6. The attacker has an attack profile — DDoS attack — which is configured to start between 100 and 110 seconds into the simulation. When the attack begins a script is sent to all nodes within the federation with the cyber effects profiles enabled (in this case it is workstations within each sub-network), in an attempt to infect the devices. A message is returned back to the attacker indicating whether the infection was successful or not, as a code excerpt shows in Figure 5-7.

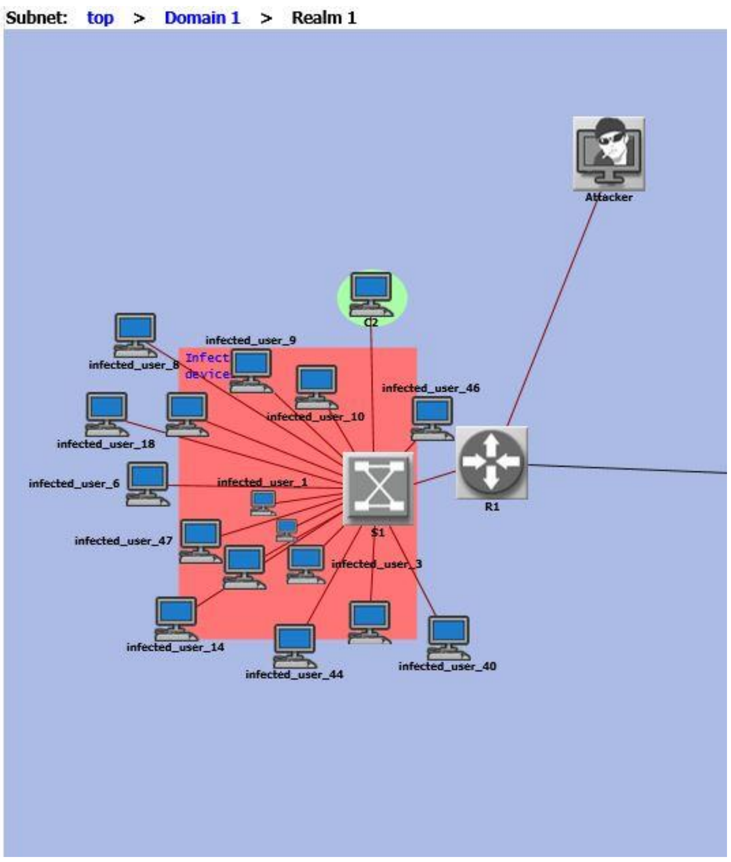
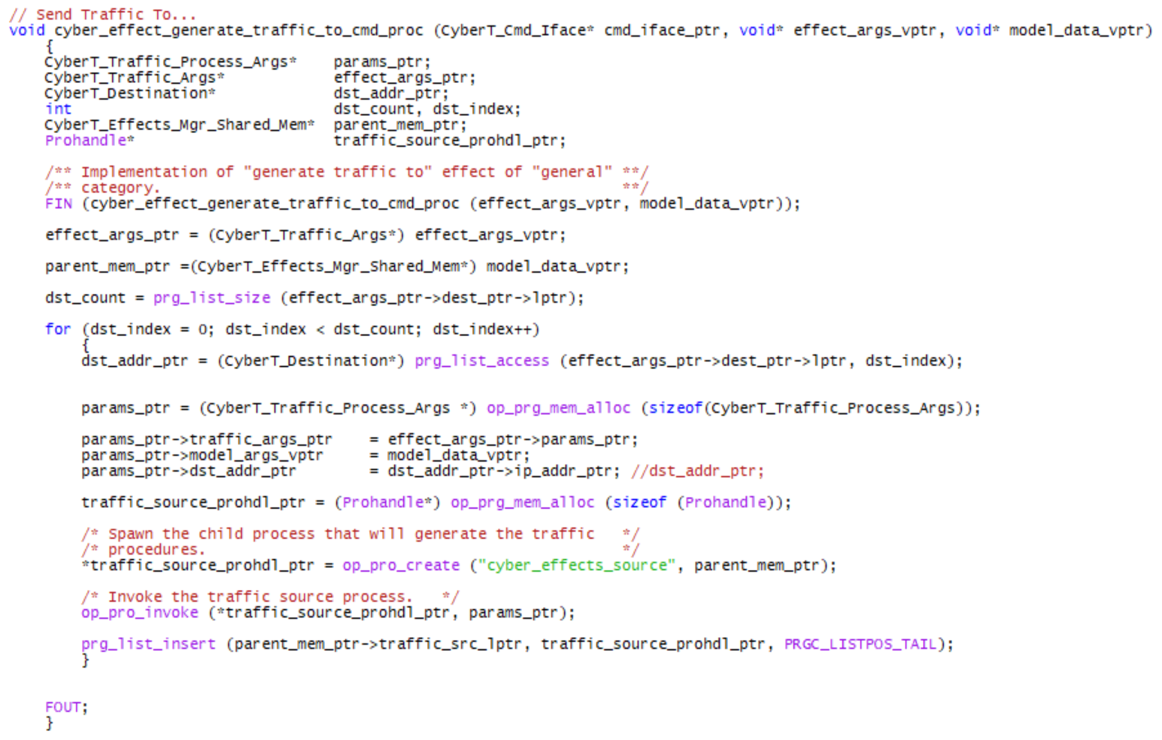


Figure 5-6: CSP 1/Attacker Realm

```
void cyber_effect_send_device_infection_status_cmd_proc (CyberT_Cmd_Iface* cmd_iface_ptr, void* effect_args_vptr, void* model_data_vptr)
    {
    CyberT_Infection_Args*          effect_args_ptr;
    CyberT_Effects_Mgr_Shared_Mem*  parent_mem_ptr;
    Packet*                         pkptr;

    /** Implementation of "return infect confirmation" effect of    **/
    /** "general" category.                                         **/
    FIN (cyber_effect_send_device_infection_status_cmd_proc (effect_args_vptr, model_data_vptr));

    if (cmd_iface_ptr == OPC_NIL)
        FOUT;

    if (cmd_iface_ptr->src_info_ptr == OPC_NIL)
        FOUT;

    effect_args_ptr = (CyberT_Infection_Args *) effect_args_vptr;

    parent_mem_ptr = (CyberT_Effects_Mgr_Shared_Mem *) model_data_vptr;

    pkptr = op_pk_create_fmt ("cyber_effects_resp");

    op_pk_nfd_set_int32 (pkptr, "infect_status", (int) parent_mem_ptr->infection_state);

    op_pk_nfd_set_ptr (pkptr, "src_info",
                    cyber_src_info_field_copy_proc (cmd_iface_ptr->src_info_ptr,0),
                    cyber_src_info_field_copy_proc,
                    cyber_src_info_field_dealloc_proc,
                    sizeof(CyberT_Msg_Info));

    cyber_effects_mgr_traffic_out_stats_write (shared_mem_ptr, pkptr);

    cyber_effects_udp_pk_send (pkptr,
                    RESERVED_UDP_PORT_NUM,
                    RESERVED_UDP_PORT_NUM,
                    shared_mem_ptr->inet_address,
                    cmd_iface_ptr->src_info_ptr->src_addr,
                    shared_mem_ptr->command_ici_ptr,
                    shared_mem_ptr->udp_output_strm);
    FOUT;
    }
```

Figure 5-7: Return infection status code excerpt

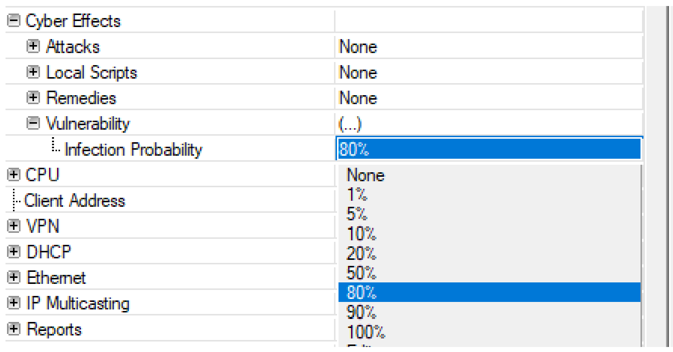
Once this occurs, another script is sent to the infected nodes to start sending traffic to the server in an attempt to flood this node with traffic, affecting availability of the device as shown in Figure 5-8.



```
// Send Traffic To...
void cyber_effect_generate_traffic_to_cmd_proc (CyberT_Cmd_Iface* cmd_iface_ptr, void* effect_args_vptr, void* model_data_vptr)
    {
    CyberT_Traffic_Process_Args*    params_ptr;
    CyberT_Traffic_Args*            effect_args_ptr;
    CyberT_Destination*             dst_addr_ptr;
    int                             dst_count, dst_index;
    CyberT_Effects_Mgr_Shared_Mem*  parent_mem_ptr;
    Prohandle*                      traffic_source_prohdl_ptr;

    /** Implementation of "generate traffic to" effect of "general" **/
    /** category.                                                   **/
    FIN (cyber_effect_generate_traffic_to_cmd_proc (effect_args_vptr, model_data_vptr));

    effect_args_ptr = (CyberT_Traffic_Args*) effect_args_vptr;

    parent_mem_ptr =(CyberT_Effects_Mgr_Shared_Mem*) model_data_vptr;

    dst_count = prg_list_size (effect_args_ptr->dest_ptr->lptr);

    for (dst_index = 0; dst_index < dst_count; dst_index++)
        {
        dst_addr_ptr = (CyberT_Destination*) prg_list_access (effect_args_ptr->dest_ptr->lptr, dst_index);

        params_ptr = (CyberT_Traffic_Process_Args *) op_prg_mem_alloc (sizeof(CyberT_Traffic_Process_Args));

        params_ptr->traffic_args_ptr    = effect_args_ptr->params_ptr;
        params_ptr->model_args_vptr     = model_data_vptr;
        params_ptr->dst_addr_ptr        = dst_addr_ptr->ip_addr_ptr; //dst_addr_ptr;

        traffic_source_prohdl_ptr = (Prohandle*) op_prg_mem_alloc (sizeof (Prohandle));

        /* Spawn the child process that will generate the traffic   */
        /* procedures.                                               */
        *traffic_source_prohdl_ptr = op_pro_create ("cyber_effects_source", parent_mem_ptr);

        /* Invoke the traffic source process.   */
        op_pro_invoke (*traffic_source_prohdl_ptr, params_ptr);

        prg_list_insert (parent_mem_ptr->traffic_src_lptr, traffic_source_prohdl_ptr, PRGC_LISTPOS_TAIL);
        }


    FOUT;
    }
```

Figure 5-8: Send traffic to destination code excerpt

The server has been set with an IP address of 192.102.100.1, so this is where the infected devices would send their requests. The attributes of these attack parameters can be changed by modifying the device attributes, or adding profiles into the cyber effects configuration manager. Currently the characteristics for the server are that it has an infection probability of 80%; this parameter can be adjusted throughout to compare the effects within the network, as illustrated in Figure 5.9. The server under attack domain is shown in Figure 5-10.



⊟ Cyber Effects	
⊞ Attacks	None
⊞ Local Scripts	None
⊞ Remedies	None
⊟ Vulnerability	(...)
Infection Probability	80%
⊞ CPU	
Client Address	
⊞ VPN	
⊞ DHCP	
⊞ Ethernet	
⊞ IP Multicasting	
⊞ Reports	

Figure 5-9: Infection probability adjustment

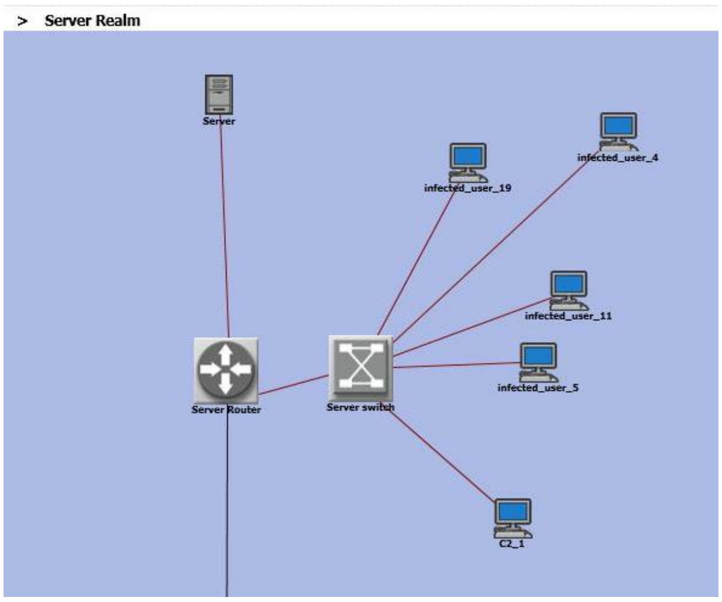


Figure 5-10: Server under attack domain

The broker node has a remedy profile configured, and this profile is also present with the C2s in adjacent domains. The “Scan and Clean” script scans the network for infected devices and cleans them, reducing the amount of traffic being sent to the Server – as shown in Figure 5-11.

```
void cyber_effect_scan_and_clean_device_cmd_exec (void* call_args_vptr, int value)
    {
    CyberT_Cmd_Iface*            cmd_iface_ptr;
    CyberT_Cmd_Model_Data*       cmd_data_ptr;
    int                          client_count = 0;
    int                          client_index;
    char*                        key;
    int                    cmd_count = 0, cmd_index;
    CyberT_Item_Set* cmd_definitions_ptr;
    CyberT_Effects_Mgr_Voids*    call_args_ptr = (CyberT_Effects_Mgr_Voids*) call_args_vptr;
    CyberT_Effects_Mgr_Shared_Mem* parent_mem_ptr;
    CyberT_Scan_Args* scan_args_ptr;

    FIN (cyber_effect_scan_and_clean_device_cmd_exec(call_args_vptr, value));

    scan_args_ptr   = call_args_ptr->one_ptr;
    parent_mem_ptr  = call_args_ptr->two_ptr;

    op_prg_mem_free (call_args_ptr);

    if (parent_mem_ptr == OPC_NIL || scan_args_ptr == OPC_NIL)
        {
        FOUT;
        }

    if (parent_mem_ptr->infection_state == CyberC_Infection_Infected)
        {

        if (op_dist_uniform (1.0) <= scan_args_ptr->clean_probability)
            // If cleaning is succesfull cancel the device infection.
            cyber_effect_infection_expiration ((void*)parent_mem_ptr, 10);
        else
            // If the device is still infected exit this function.
            FOUT;
        }
    else
        FOUT;
```

Figure 5-11: Scan and clean script excerpt

Those nodes that are successfully cleaned will stop attacking the Server, and regain normal operations within the federation. Through the Discrete Event Simulation (DES) the infected device count can be analysed and visualised via the use of graphs.

From Figure 5-12, an exploratory phase where the attacker is scanning the network can be seen, followed by a large spike in infected devices, and then there is a gradual decrease in infected devices once the cleaning phase has initiated, leading to a complete cleanse occurring by the 11 minutes 45 second mark. This occurs due to the cyber effects script which sends and receives confirmation of device infection, then device cleaning.

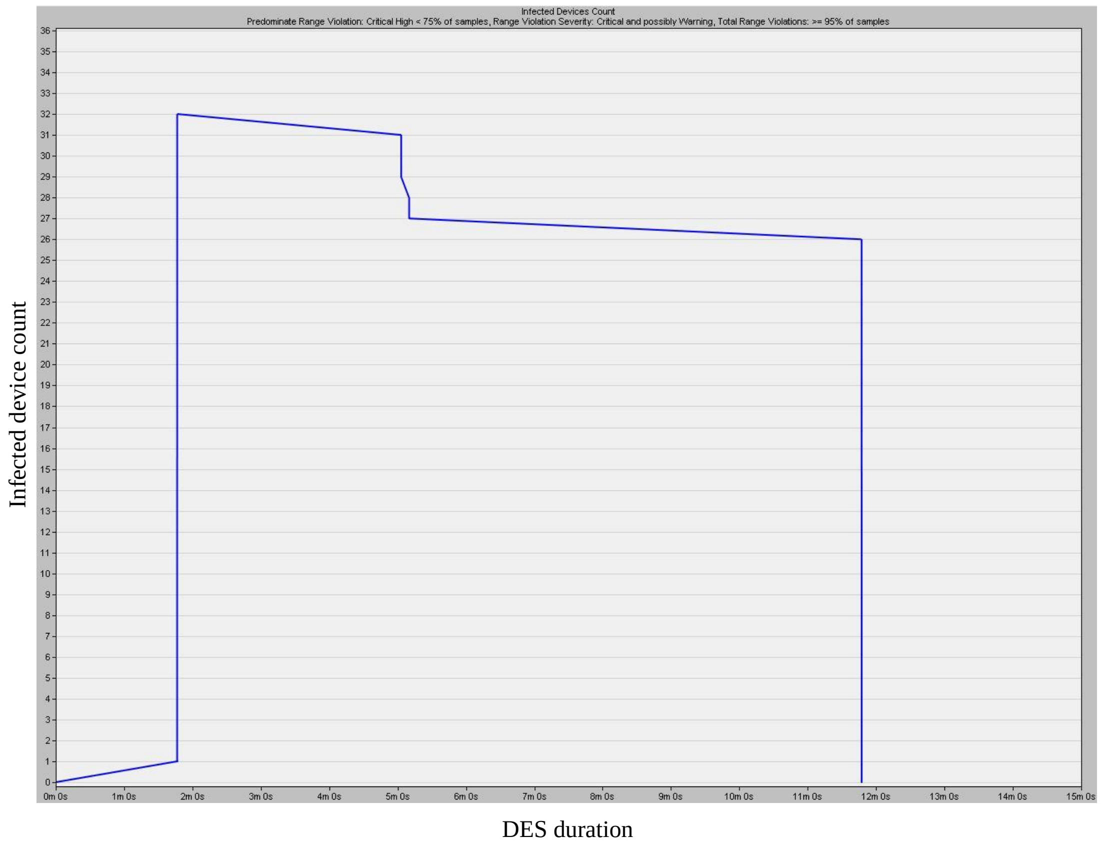


Figure 5-12: Infected Devices Count

While Figure 5-12 shows how the infected devices count fluctuates during the DES, statistics collected on the infected devices are presented in Table 5.1.

Table 5.1: Cyber effects infected devices count

Zone	0
Statistic	Cyber Effects Infected Devices Count
Length	117
Number of values	117
Horizontal, min	0
Horizontal, max	706.64
Vertical, min	0
Vertical, max	32
Initial value	0.0
Final value	1.0
Expected value	24.33
Sample mean	8.75
Variance	110.20
Standard deviation	10.49

Figure 5-13 illustrates the ethernet delay that occurs during the scanning and cleaning of the Cloud federation. Note that the spike in delay occurs two minutes into the simulation which is when the DDoS attack begins. Ethernet delay within the federation is evidently affected by the scanning and cleaning of the network during infection, and the abuse of resources as the attack uses devices within the networks.

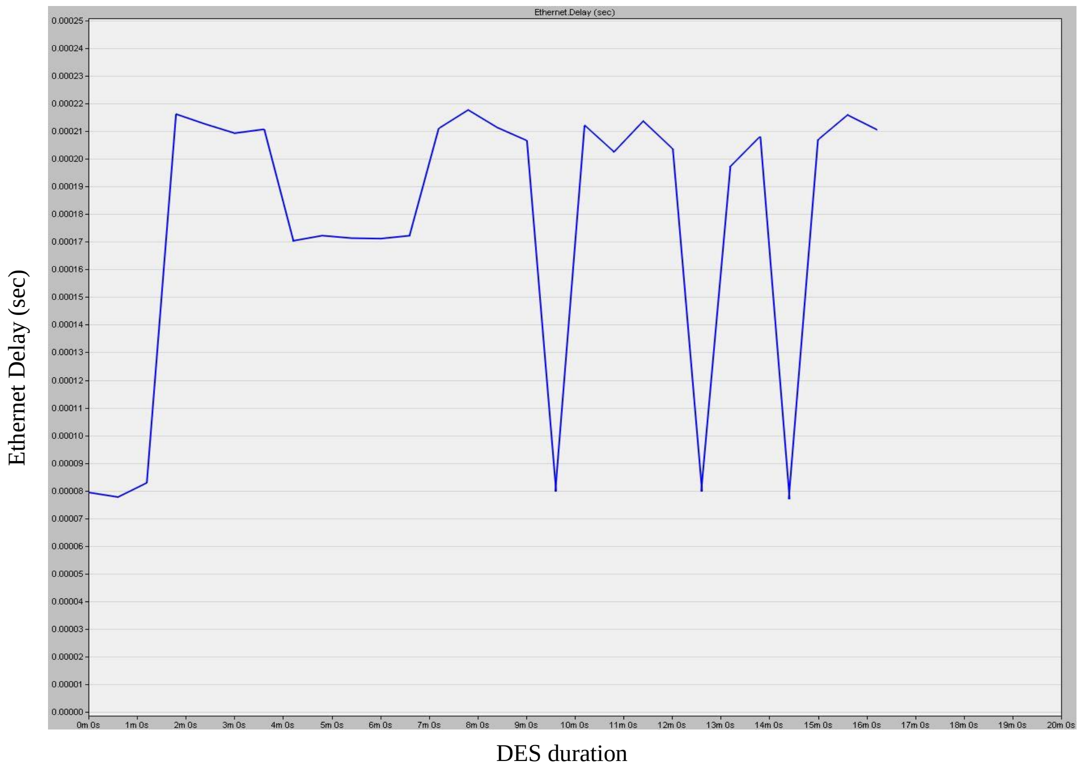


Figure 5-13: Ethernet delay within Cloud federation

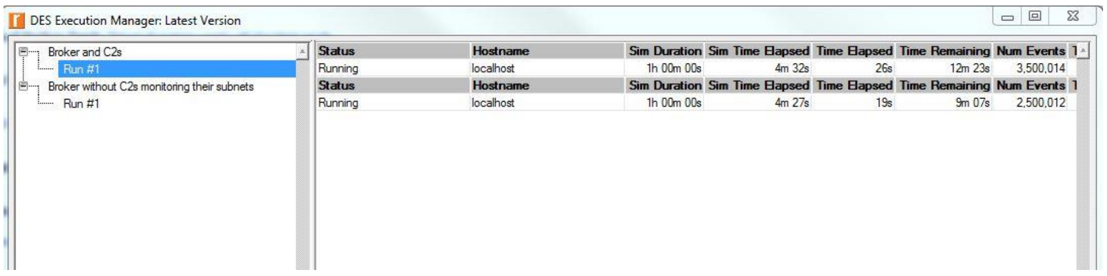
5.2 Co-simulating scenarios

The links between critical infrastructures can be either dependent or interdependent. Dependency is a unidirectional relationship whereas interdependency is a bidirectional relationship where the capabilities of one infrastructure influence the state of another [149]. In the case of Cloud federations, it could be argued that both of these relationships are evident, but for the purpose of simulation the focus is on bidirectional interdependency. Each of the CSPs need to be robust and resilient to random or unplanned faults/attacks, and keep the effects of cascading failure in an interdependent network to a minimum.

In order to show proof of concept, a normal scenario of a federated Cloud environment was implemented, and then compared to the same environment under duress. One of the CSPs within the federation was flooded with traffic, which would cause the associated SN to generate a belief that there is an attack underway.

Figure 5-14 illustrates the co-simulation of the single monitoring entity (Broker without C2s monitoring their subnets) vs our hierarchical structure (Broker and C2s). It is important to analyse the effects of the monitoring entities on the network performance, and also the hierarchical structure that is adopted.

Riverbed Modeler 18.0 facilitates this by performing co-simulation of two scenarios and allows the user to select comparative metrics.


DES Execution Manager: Latest Version
Broker and C2s
Run #1
Broker without C2s monitoring their subnets
Run #1

Status	Hostname	Sim Duration	Sim Time Elapsed	Time Elapsed	Time Remaining	Num Events
Running	localhost	1h 00m 00s	4m 32s	26s	12m 23s	3,500,014

Status	Hostname	Sim Duration	Sim Time Elapsed	Time Elapsed	Time Remaining	Num Events
Running	localhost	1h 00m 00s	4m 27s	19s	9m 07s	2,500,012

Figure 5-14: Co simulation of single monitoring entity vs hierarchical structure

Within Figure 5-14, the DES manager shows how there are two scenarios that will be run under the same simulation conditions but have been adjusted to test differing architectures. Two scenarios, run in parallel, will compare how they react and are affected by the attacker within the network, and react to the attack via scanning/cleaning/communicating with the monitoring entities. The co-simulation measured the effects of having one single monitoring entity on the network (one central C2 with the responsibility of monitoring the whole network), compared to having a more hierarchical based approach whereby there were C2s throughout the federation in adjacent domains with defined roles and responsibilities.

Figure 5-15 shows the Ethernet delay in comparison to the two scenarios within the simulated environment. With the communication structure adopted, the Ethernet delay was greatly affected by the Broker and C2s due to the increased communications, in comparison to the single monitoring entity.

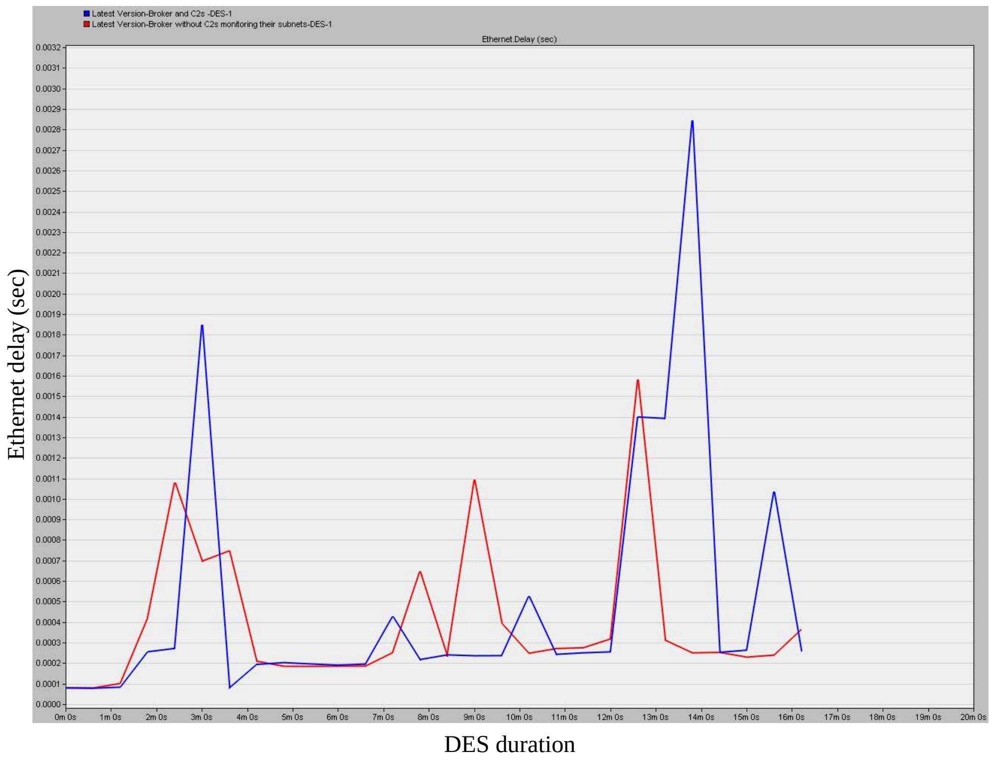


Figure 5-15: Ethernet delay comparison of co-simulation of single monitoring entity vs hierarchical structure

As the parameters of the attack were the same and how the monitoring structure was affected was a focus, it was evident how the attack propagated and targeted the server. The threshold for the scenarios was determined by running the scenario number of runs necessary to achieve a good estimate of a confidence level and the set of parameters that reflected the behaviour of the system. Device infection probability was set to 80%, and the infected devices were set to flood IP 192.102.100.1 (which is the server in the Server domain in both scenarios). The server is set to an availability percentage of 90%, which corresponds to a SLA level of 90 % uptime/availability.

Figure 5-16 shows how the traffic against the server increased with the Broker and C2s architecture, as opposed to one C2, due to the increase in entities within the scenario.

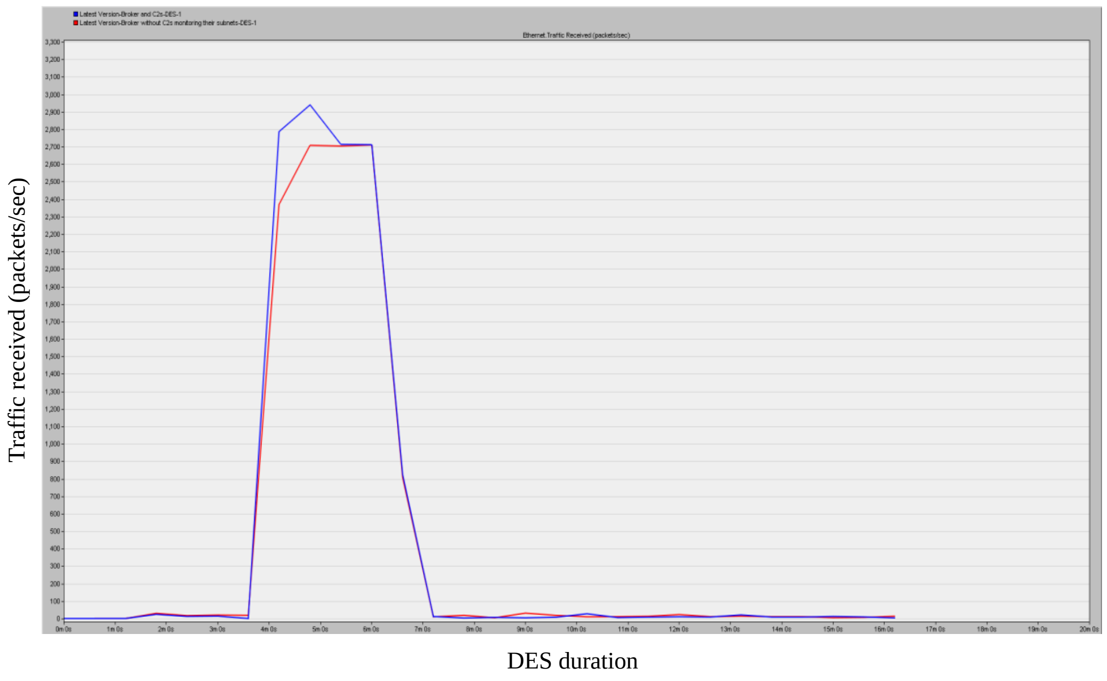


Figure 5-16: Ethernet traffic against Server

In contrast, as shown in Figure 5-16, the Server under attack receives less traffic with the single monitoring entity compared to our hierarchical structure. The statistics from the co-simulation are detailed in Table 5.2. Introducing the hierarchical monitoring via splitting the C2s into the differing domains allows the C2s to scan and remedy the device infections as they would notice these deviations quicker as they are closer to the sources.

Table 5.2: Comparison of Server statistics in co-simulation

	Latest Version-Broker and C2s-DES-1: Server Domain.Server Realm.Server.Ethernet.Traffic Received (packets/sec)	Latest Version-Broker without C2s monitoring their subnets-DES-1: Server Domain.Server Realm.Server.Ethernet.Traffic Received (packets/sec)
Length	29	29
Number of values	29	29
Horizontal, min:	0	0
Horizontal, max:	1008	1008
Vertical, min:	0.083	0.083
Vertical, max:	2,939.88	2,710.11
Initial value:	0.08	0.08
Final value:	3.67	13.06
Expected value:	435.07	414.02

Sample mean:	435.07	414.02
Variance:	947,150.32	838,386.17
Standard deviation:	973.22	915.64

Looking at the comparison of the server statistics in Table 5.2, there is a large standard deviation in both scenarios. As the server was under attack it is understandable that the statistics would be intensified but the DES provides a rudimentary analysis for the device.

At this stage of the simulation, the main purposes were to analyse:

- The role a broker could have with autonomous sharing of information
- The role of a single monitoring entity on the entire federation vs the C2s monitoring their own sub domains
- How an attack within a Cloud federation could affect the interdependent services present
- The associated throughput and delay
- What the DDoS attack effects are on the server.

Note: determining the role of monitoring entities on the federation is very much dependent on the scale of the federation concerned. The scale of the simulated federation explores this matter, but may not be appropriate in answering the question in general due to resource constraints.

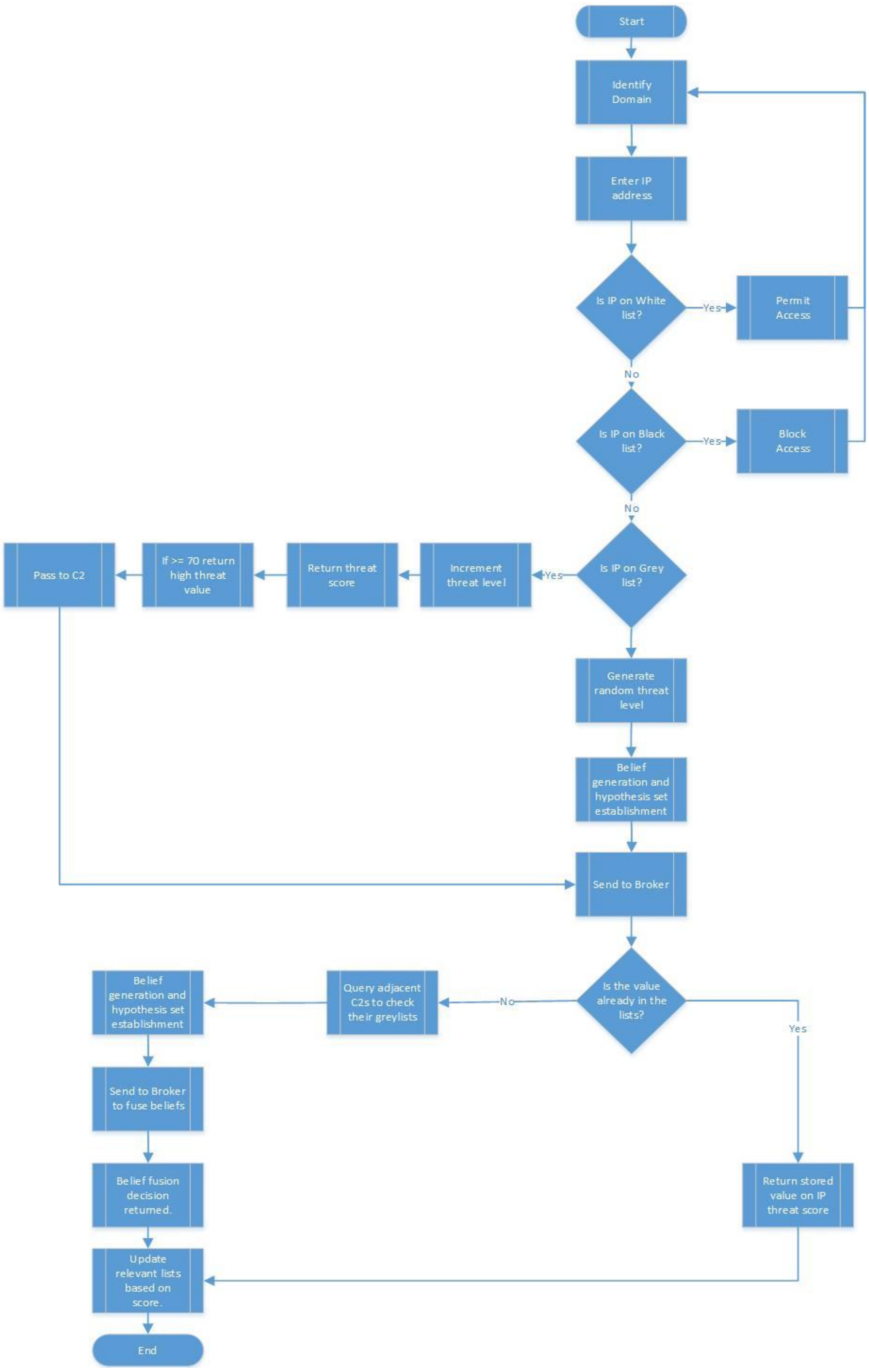
5.3 Collaborative intrusion detection application

Next, the actions to be taken in the simulation, from the point where an intrusion is believed to have been detected are shown. The integration of the decisions coming from different IDSs has emerged as a technique that could strengthen the final decision. Sensor fusion can be defined as the process of collecting information from multiple and possibly heterogeneous sources and combining them to obtain a more descriptive, intuitive and meaningful result [57]. Related work in the field of sensor fusion has been carried out mainly with one of the methods like probability theory, evidence theory, voting fusion theory, fuzzy logic theory, or neural network theory in order to aggregate information.

A major concern on studying information in a distributed system containing autonomous entities is how to model an adversarial threat. Most traditional solutions have a common assumption that all entities are well disciplined to follow the protocol properly, with the only exception that an adversary may keep a record of all intermediate computation. It is difficult to determine if an entity has the capability to change its input database or deviate from the protocol in real world applications. Anomaly-based IDSs detect anomalies beyond a set threshold level in the features they detect, whereas using threshold bounds gives more freedom in steering system properties. Any threshold within the bounds can be chosen depending on the preferred level of trade-off between detection and false alarm rates.

Figure 5-17 shows the runtime flowchart of our D-S fusion proof of concept which runs from the point where a belief is generated about an observation. Implemented in C#, belief generation and data fusion by the broker is conveyed, in addition to hypothesis tables and associated belief information, i.e. belief value, mass, and plausibility. The flowchart is from the perspective of a suspicious observation within a domain. When the program runs you initially identify the domain you currently reside in, then enter an IP address. Based on the entered value the program checks the entered value against the domain white, black, and grey lists to determine if a value is known. If the value is on the white list then access is granted. If the value is on the black list then access is denied. If the value is presently on a grey list then the threat level is incremented, the associated value associated is returned, and if the value is more than or equal to a defined threshold (in our simulation this value is 70), this information is passed to the C2.

The C2 receives a query and associated belief hypothesis from the SN, and queries the Broker with this information. The broker checks if it possess information on the value and returns the stored information to the C2 if present. Otherwise, the broker queries C2s in other domains to check their grey lists and return a BPA based on their stored value, return a BPA based on the queried value. The broker fuses the generated beliefs and makes a system decision – relevant lists are updated based on the score.



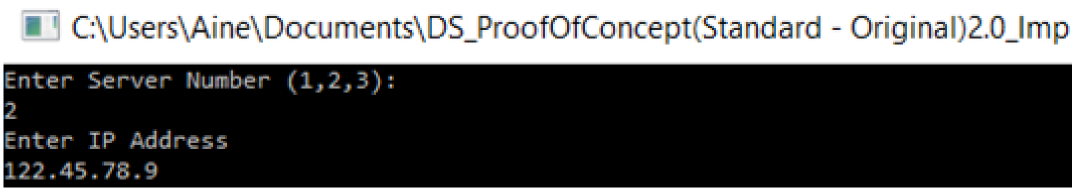
Start
Identify Domain
Enter IP address
Is IP on White list?
Yes
Permit Access
No
Is IP on Black list?
Yes
Block Access
No
Is IP on Grey list?
Yes
Increment threat level
Return threat score
If >= 70 return high threat value
Pass to C2
Generate random threat level
Belief generation and hypothesis set establishment
Send to Broker
Is the value already in the lists?
No
Query adjacent C2s to check their greylists
Belief generation and hypothesis set establishment
Send to Broker to fuse beliefs
Belief fusion decision returned.
Update relevant lists based on score.
End
Yes
Return stored value on IP threat score

Figure 5-17: Runtime flowchart

Our implementation of our Service-based collaborative intrusion detection methodology has three stages. For proof of concept, a lower number of entities are used to convey how communication occurs and the information would be exchanged within the infrastructure; future work would involve expanding this solution to cope with a larger scale.

Stages 1-3

Initially, the user is prompted to identify which C2 domain they are present in. Then an IP address is entered into the program and the value is compared to the Black list, Grey list and White list to see if the value is present. The actions are as below:

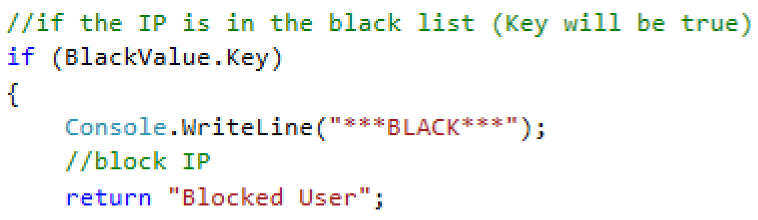


```
C:\Users\Aine\Documents\DS_ProofOfConcept(Standard - Original)2.0_Imp
Enter Server Number (1,2,3):
2
Enter IP Address
122.45.78.9
```

Figure 5-18: Enter Domain and IP address

Stage 1 - Value on Black list

When compared against the lists, if the IP address is in the Black list then the user is ‘Blocked’.

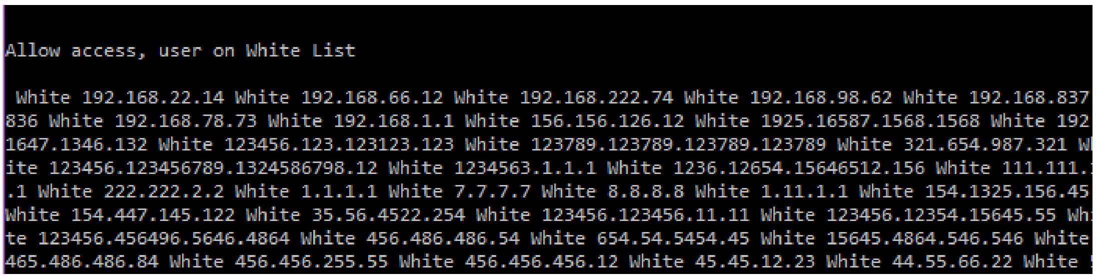


```
//if the IP is in the black list (Key will be true)
if (BlackValue.Key)
{
    Console.WriteLine("***BLACK***");
    //block IP
    return "Blocked User";
```

Figure 5-19: Blocked user key return

Stage 2 - Value on White list

The IP is entered and the value is compared against the lists. If the IP address is present on the White list then the user is ‘Permitted Access’. The console outputs the other values from the white list, and this is also a separate file that can be viewed.

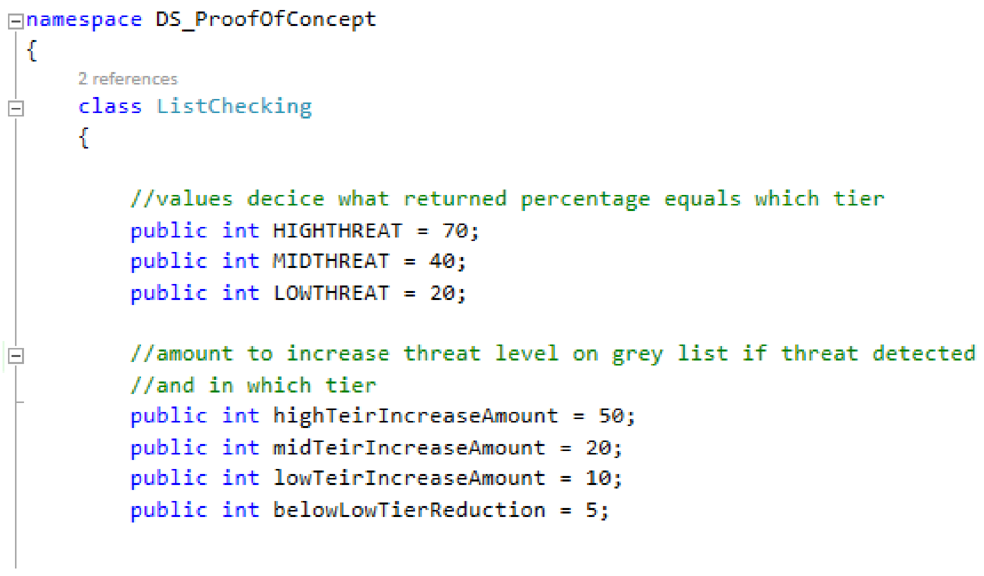


```
Allow access, user on White List

 White 192.168.22.14 White 192.168.66.12 White 192.168.222.74 White 192.168.98.62 White 192.168.837
836 White 192.168.78.73 White 192.168.1.1 White 156.156.126.12 White 1925.16587.1568.1568 White 192
1647.1346.132 White 123456.123.123123.123 White 123789.123789.123789.123789 White 321.654.987.321 Wh
ite 123456.123456789.1324586798.12 White 1234563.1.1.1 White 1236.12654.15646512.156 White 111.111.
.1 White 222.222.2.2 White 1.1.1.1 White 7.7.7.7 White 8.8.8.8 White 1.11.1.1 White 154.1325.156.45
White 154.447.145.122 White 35.56.4522.254 White 123456.123456.11.11 White 123456.12354.15645.55 Wh
te 123456.456496.5646.4864 White 456.486.486.54 White 654.54.5454.45 White 15645.4864.546.546 White
465.486.486.84 White 456.456.255.55 White 456.456.456.12 White 45.45.12.23 White 44.55.66.22 White
```

Figure 5-20: Value on white list

Stage 3 - Value on Grey list

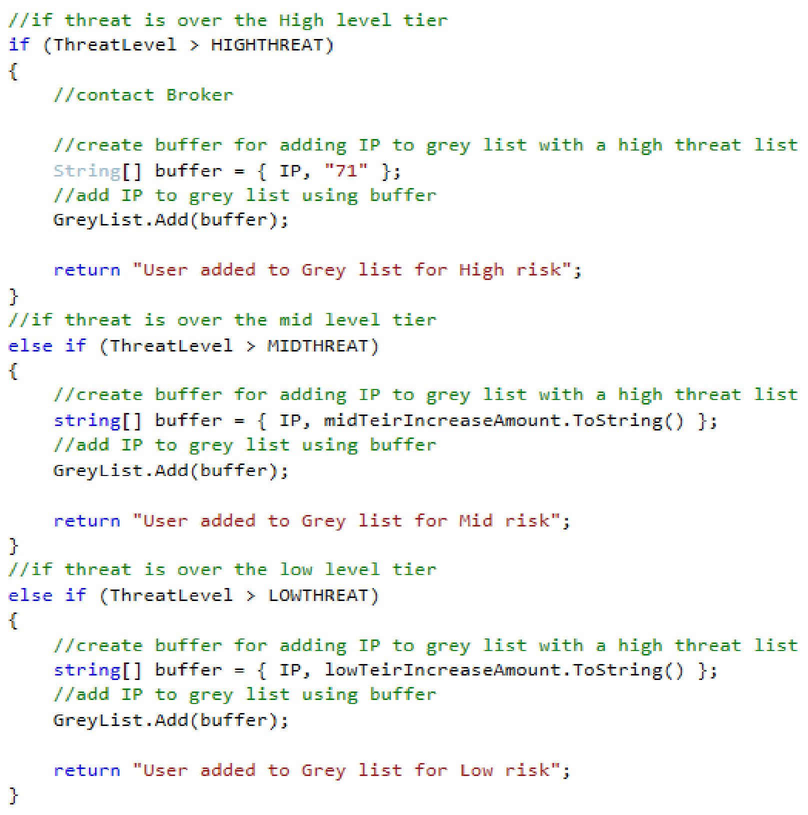
If not present on either list, the value is stored in the Grey list and given a threat value (randomly generated numbers were used to determine this value – then the predefined threshold values to determine its location on the list) which are used to form the belief. Hypothesis sets based on all values between 0 and 1 are included within the program, as well as associated mass values and plausibility functions – these can be found in Table 4.2 in Chapter 4.5.



```
namespace DS_ProofOfConcept
{
    2 references
    class ListChecking
    {

        //values decice what returned percentage equals which tier
        public int HIGHTHREAT = 70;
        public int MIDTHREAT = 40;
        public int LOWTHREAT = 20;

        //amount to increase threat level on grey list if threat detected
        //and in which tier
        public int highTeirIncreaseAmount = 50;
        public int midTeirIncreaseAmount = 20;
        public int lowTeirIncreaseAmount = 10;
        public int belowLowTierReduction = 5;
```

Figure 5-21: Threat value ranges

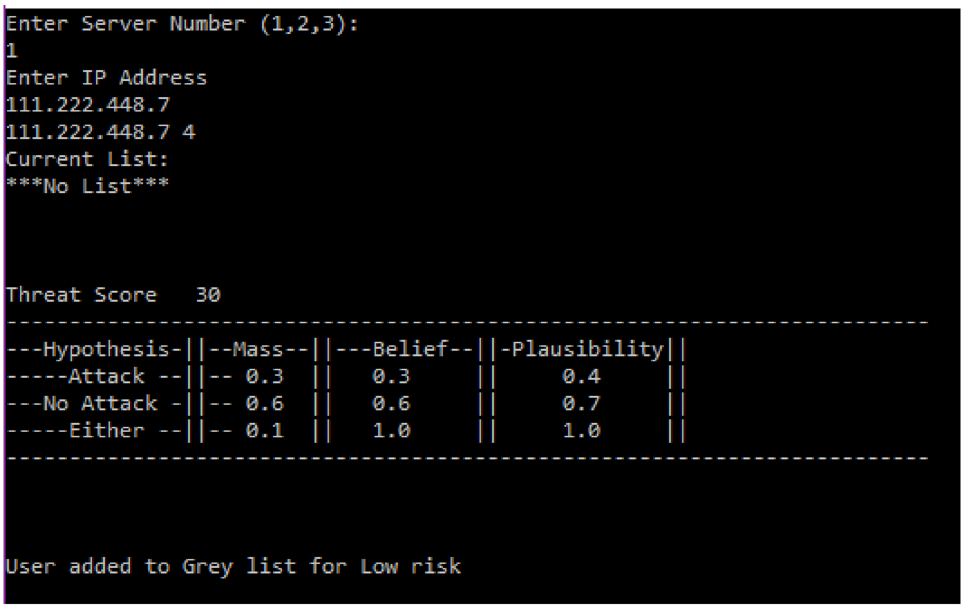
Figure 5-21 shows the threat value ranges used, and the ability to increase/decrease the associated risk due to occurrences on the list is also an option. Figure 5-22 shows the associated outputs to the console depending upon the assigned score. The threat ranges used are based on a maximum value of 100 and a minimum value of 0, and have been defined based on typical grading criteria. Increased occurrences could cause the risk score to increase, e.g. beginning on the white list, moving to the grey list, but then being promoted to the black list.



```
//if threat is over the High level tier
if (ThreatLevel > HIGHTHREAT)
{
    //contact Broker

    //create buffer for adding IP to grey list with a high threat list
    String[] buffer = { IP, "71" };
    //add IP to grey list using buffer
    GreyList.Add(buffer);

    return "User added to Grey list for High risk";
}
//if threat is over the mid level tier
else if (ThreatLevel > MIDTHREAT)
{
    //create buffer for adding IP to grey list with a high threat list
    string[] buffer = { IP, midTeirIncreaseAmount.ToString() };
    //add IP to grey list using buffer
    GreyList.Add(buffer);

    return "User added to Grey list for Mid risk";
}
//if threat is over the low level tier
else if (ThreatLevel > LOWTHREAT)
{
    //create buffer for adding IP to grey list with a high threat list
    string[] buffer = { IP, lowTeirIncreaseAmount.ToString() };
    //add IP to grey list using buffer
    GreyList.Add(buffer);

    return "User added to Grey list for Low risk";
}
```

Figure 5-22: IP assigned to list depending on the range of score

In Figure 5-23, a user has a threat score of 30. This user has been added to the grey list with a low risk assigned to it. If this IP address was to occur more frequently this assigned value could increase and this IP moved up the list.



```
Enter Server Number (1,2,3):
1
Enter IP Address
111.222.448.7
111.222.448.7 4
Current List:
***No List***



Threat Score   30
-------------------------------------------------------------------------
---Hypothesis-||--Mass--||---Belief--||-Plausibility||
-----Attack --||-- 0.3  ||   0.3     ||     0.4     ||
---No Attack -||-- 0.6  ||   0.6     ||     0.7     ||
-----Either --||-- 0.1  ||   1.0     ||     1.0     ||
-------------------------------------------------------------------------



User added to Grey list for Low risk
```

Figure 5-23: User added to grey list with low risk

For testing purposes a threshold of 70 was set to trigger belief generation and the associated hypothesis values output. In Figure 5-24, an entered IP has been assigned a threat score of 80 as it is not presently on a list.

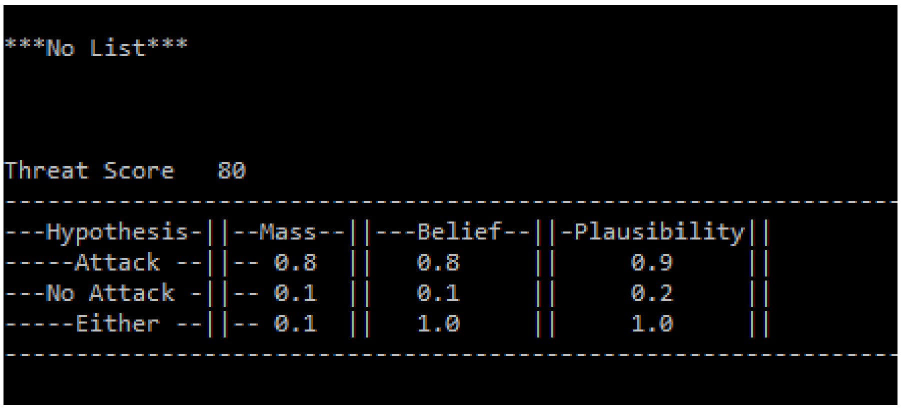

```
***No List***


Threat Score   80
---------------------------------------------------------------
---Hypothesis-||--Mass--||---Belief--||-Plausibility||
-----Attack --||-- 0.8  ||   0.8     ||     0.9     ||
---No Attack -||-- 0.1  ||   0.1     ||     0.2     ||
-----Either --||-- 0.1  ||   1.0     ||     1.0     ||
---------------------------------------------------------------
```

Figure 5-24: Example hypothesis set generation for a threat score of 80

This value is sent to the broker and compared against the Black and White lists, as this information may not have been published to the C2s within the federation. The broker then queries the adjacent monitoring entities and requests they generate a belief based on the original value. Two examples of outputs from C2s are in the Figures 5-25 and 5-26 below.

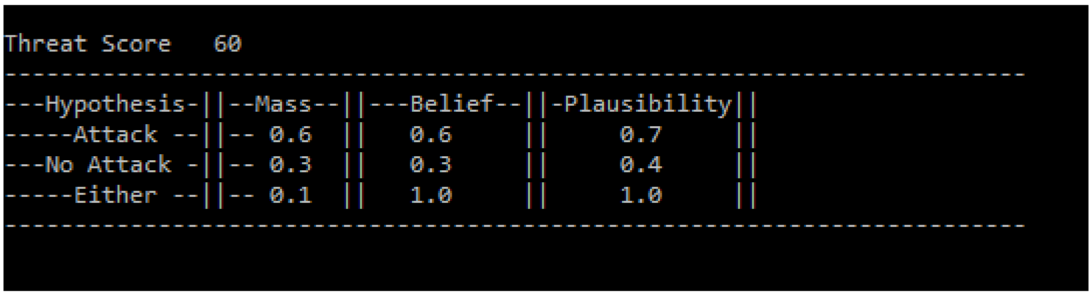

```
Threat Score   60
--------------------------------------------------------------------------
---Hypothesis-||--Mass--||---Belief--||-Plausibility||
-----Attack --||-- 0.6  ||   0.6     ||     0.7     ||
---No Attack -||-- 0.3  ||   0.3     ||     0.4     ||
-----Either --||-- 0.1  ||   1.0     ||     1.0     ||
--------------------------------------------------------------------------
```

Figure 5-25: A belief generation of 0.6

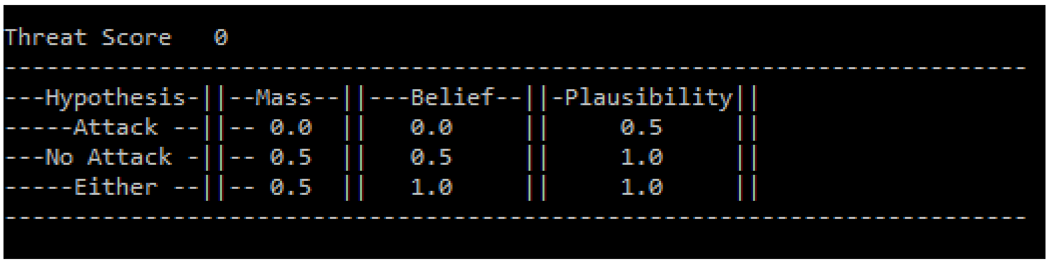

```
Threat Score   0
--------------------------------------------------------------------------
---Hypothesis-||--Mass--||---Belief--||-Plausibility||
-----Attack --||-- 0.0  ||   0.0     ||     0.5     ||
---No Attack -||-- 0.5  ||   0.5     ||     1.0     ||
-----Either --||-- 0.5  ||   1.0     ||     1.0     ||
--------------------------------------------------------------------------
```

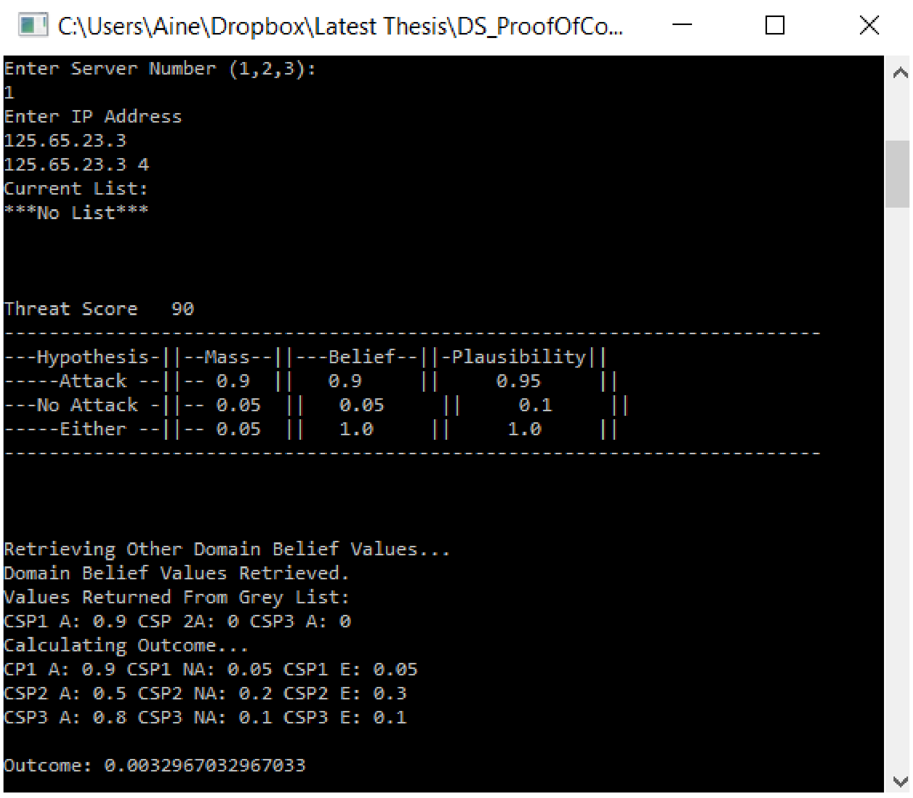
Figure 5-26: A belief generation of 0.0

The broker, using equation 4.25, takes the three belief values and fuses them together to make a system wide decision. Sample code for this is shown in Figure 5-27 below – all D-S proof of concept code is available in the Appendix:

```
public Broker(String IP, double value)
        {
            Console.WriteLine("Retrieving Other Domain Belief Values...");
            //string _filePath =
Path.GetDirectoryName(System.AppDomain.CurrentDomain.BaseDirectory);
            //StreamWriter GreyFile = new System.IO.StreamWriter(_filePath +
@"\CurrentListData\S" + Server + "GreyList.txt");
            GetOtherServerValues(IP);
            if (Program.Server == 1) { S1A = value /100; }
            if (Program.Server == 2) { S2A = value /100; }
            if (Program.Server == 3) { S3A = value /100; }

            Console.WriteLine("Domain Belief Values Retrieved.");
            Console.WriteLine("Values Returned From Grey List:");
            Console.WriteLine("CSP1 A: " + S1A + " CSP 2A: " + S2A + " CSP3 A: " +
S3A);
            Console.WriteLine("Calculating Outcome...");
            CombinationBelief();
            Console.WriteLine("Calculation Complete.");
            Console.WriteLine();
        }

        void CombinationBelief()
        {
            attackToThreeValue();

            double cooefficient = (S1A * S2NA) + (S1NA * S2A) + (S1A * S3NA) +
                            (S1NA * S3A) + (S2A * S3NA) + (S2NA * S3A);
            double final = (1 / (1 - cooefficient)) * S1E * S2E * S3E;

            Console.WriteLine("CP1 A: " + S1A + " CSP1 NA: " + S1NA + " CSP1 E: " +
S1E);
            Console.WriteLine("CSP2 A: " + S2A + " CSP2 NA: " + S2NA + " CSP2 E: " +
S2E);
            Console.WriteLine("CSP3 A: " + S3A + " CSP3 NA: " + S3NA + " CSP3 E: " +
S3E);
            Console.WriteLine();
            Console.WriteLine("Outcome: " + final);
            Console.ReadLine();
        }
```

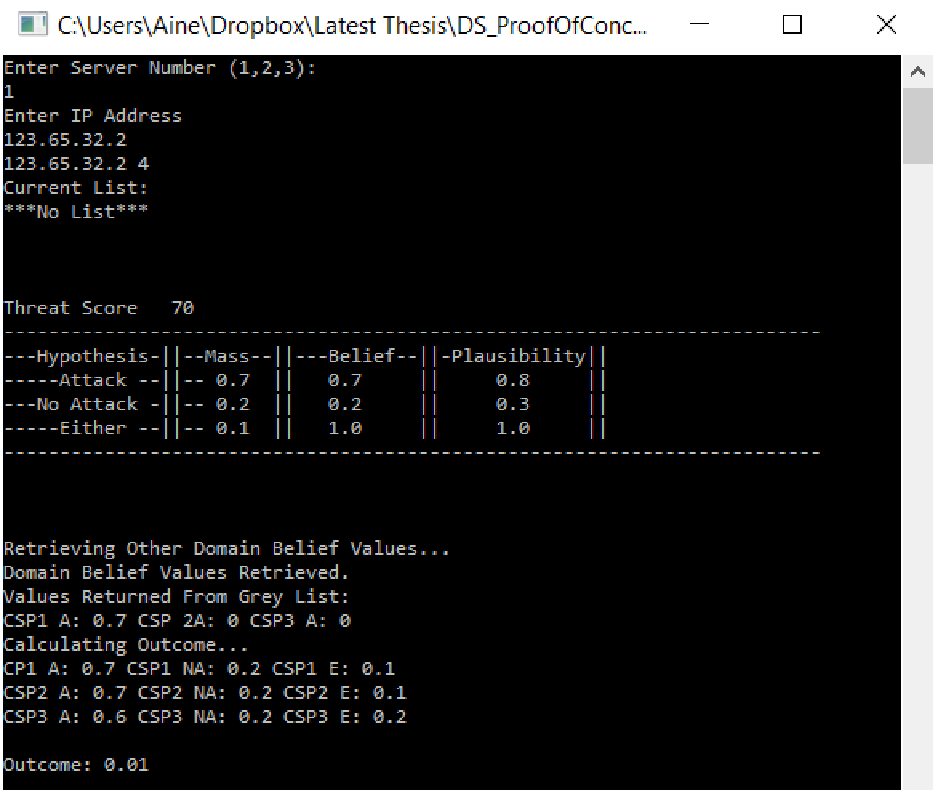
Figure 5-27: D-S belief fusion code

Depending on the threshold established, this returned value would indicate if the belief is an attack or not an attack. This decision would be updated to the lists either white or black for each domain. Figure 5-28 illustrates belief fusion with an initial belief generation of 0.9, and the returned values from the CSPs in other domains. In Figure 5-29, belief combination occurs from an initial belief value of 0.7.



```
Enter Server Number (1,2,3):
1
Enter IP Address
125.65.23.3
125.65.23.3 4
Current List:
***No List***



Threat Score   90
-------------------------------------------------------------------------
---Hypothesis-||--Mass--||---Belief--||-Plausibility||
-----Attack --||-- 0.9  ||   0.9     ||     0.95     ||
---No Attack -||-- 0.05  ||   0.05     ||     0.1     ||
-----Either --||-- 0.05  ||   1.0     ||     1.0     ||
-------------------------------------------------------------------------



Retrieving Other Domain Belief Values...
Domain Belief Values Retrieved.
Values Returned From Grey List:
CSP1 A: 0.9 CSP 2A: 0 CSP3 A: 0
Calculating Outcome...
CP1 A: 0.9 CSP1 NA: 0.05 CSP1 E: 0.05
CSP2 A: 0.5 CSP2 NA: 0.2 CSP2 E: 0.3
CSP3 A: 0.8 CSP3 NA: 0.1 CSP3 E: 0.1

Outcome: 0.0032967032967033
```

Figure 5-28: D-S belief fusion from 3 CSPs



```
Enter Server Number (1,2,3):
1
Enter IP Address
123.65.32.2
123.65.32.2 4
Current List:
***No List***



Threat Score   70
-------------------------------------------------------------------------
---Hypothesis-||--Mass--||---Belief--||-Plausibility||
-----Attack --||-- 0.7  ||   0.7     ||     0.8     ||
---No Attack -||-- 0.2  ||   0.2     ||     0.3     ||
-----Either --||-- 0.1  ||   1.0     ||     1.0     ||
-------------------------------------------------------------------------



Retrieving Other Domain Belief Values...
Domain Belief Values Retrieved.
Values Returned From Grey List:
CSP1 A: 0.7 CSP 2A: 0 CSP3 A: 0
Calculating Outcome...
CP1 A: 0.7 CSP1 NA: 0.2 CSP1 E: 0.1
CSP2 A: 0.7 CSP2 NA: 0.2 CSP2 E: 0.1
CSP3 A: 0.6 CSP3 NA: 0.2 CSP3 E: 0.2

Outcome: 0.01
```

Figure 5-29: D-S belief fusion from 3 CSPs

5.4 Summary

This chapter has detailed the implementation of our Service-based collaborative intrusion detection solution and its subsequent attributes, as well as outlined how the collaborative intrusion detection is implemented. For proof of concept the attributes of our solution were simulated and the benefits and applicability of collaborative intrusion detection in federated Cloud environments illustrated. Application of D-S theory of evidence for collaborative intrusion detection has been implemented and its utilisation highlighted. The integration of the decisions coming from different IDSs has emerged as a technique that could strengthen the final decision. Federated Cloud environments are growing areas in terms of adoption by critical infrastructure vendors, and large corporations, so our "Security as a Service" facilitates this collaborative intrusion detection, and sharing of attack information among these different service providers. For simulation purposes an IP address as the indicator of suspicion was used but this is to demonstrate the application of the D-S algorithm for collaborative intrusion detection. Tailoring the methodology presented in Chapter 4 to include attack signatures or measurable profile information would expand the capabilities of our approach.

Chapter 6

Evaluation

In the Chapter 5, the Service-based collaborative intrusion detection framework was presented, and collaborative intrusion in a federated Cloud environment was demonstrated. CSPs within a Cloud federation are represented as interconnected domains, and based on the monitoring structure, dynamically provision the infrastructure to react and facilitate autonomous sharing of threat information. This architecture relies on a four-tier detection strategy with the elastic Cloud facilitating inter-domain operation, the MNs and SNs reporting possible attacks in the domains, and the C2s collaboratively issuing a response via the involvement of the Cloud broker, and the D-S theory of evidence.

Within this chapter the solution is evaluated against the initial aims and objectives identified in Chapter 1.2, the design requirements established in Chapter 3.5, and compared to other work within this area. Particularly, when assessing the performance of our solution in a federated Cloud environment, it is important to note that within the Cloud, resources are not a problem. Resources can complement the speed of our autonomous information sharing scheme, so it is more suitable looking at the effectiveness and accuracy of our novel approach. However, the associated delay is an important issue that cannot be ignored, particularly when the federation scale/size is large. DDoS attacks aim to starve Cloud resources, so while there are plenty, the designed solution needs to ensure it can still function when resources may be constrained due to an attack – fault tolerance is conceived to be an essential design requirement.

The effectiveness of an IDS is assessed on how capable the detection method is at making correct attack detection. According to the real nature of a given event compared to a prediction, four possible outcomes are shown in Table 6.1. These outcomes are known as the IDPS reaction matrix.

Table 6.1: Possible status for an IDPS reaction

		Predicted	
		Normal	Attack
Actual	Normal	True negative (TN)	False negative (FN)
	Attack	False positive (FP)	True positive (TP)

TNs and TPs correspond to a correct IDS operation; that is, events are successfully labelled as normal and attack. FPs refer to normal events predicted as attacks, while FN are attacks incorrectly predicted as normal events [80]. Most of the systems reviewed used the same evaluation metrics such as the detection rate and false alarm rate. While it is important to note these evaluative metrics, our approach uses the algorithms presented in 4.3 – 4.7 to calculate thresholds, generate pre-alarms and beliefs, and to make a system-wide decision.

Our aims and objectives are to identify the collaborative exchange of information as a key deliverable. Using equation 4.25 a system-wide decision is made about a possible attack, but the group consensus may not reflect the true nature of the domain of origin that generated the belief – in other words, whilst it may be possible to force an FP in one domain, testing for system-wide FP has proved problematic. This may be deemed as an FN locally, but using equation 4.25 the information was fused and this was the collaborative decision. To deal with these conflicting beliefs, our extended D-S fusion process was formulated to reduce the issue with conflicting belief fusion locally.

A high FP rate that seriously affects the system's performance can be detected, and an elevated FN rate leaves the system vulnerable to intrusions. Both FP and FN rates ought to be minimised, together with maximising TP and TN rates simultaneously. Based on Equations 6.1 – 6.6 and the reaction matrix, a possible status for IDPS reaction is shown to quantify IDS performance [80].

$$True\ negative\ rate(TNR) = \frac{TN}{TN+FP} = \frac{no.of\ true\ alerts}{no.of\ alerts} \tag{6.1}$$

$$True\ positive\ rate(TPR) \text{ or Recall (R)} = \frac{TP}{TP+FN} = \frac{no.\ of\ detected\ attacks}{no.\ of\ observable\ attacks} \tag{6.2}$$

$$False\ positive\ rate(FPR): \frac{FP}{TN+FP} = 1 - \frac{TN}{TN+FN} \tag{6.3}$$

$$False\ negative\ rate(FNR): \frac{FN}{TP+FN} \tag{6.4}$$

$$Accuracy = \frac{TN+TP}{TN+TP+FN+FP} \tag{6.5}$$

$$Precision = \frac{TP}{TP+FP} \tag{6.6}$$

There is a trade-off between the two metrics: precision and recall. As the number of detections increases by lowering of the threshold, the recall will increase, while precision is expected to decrease. The recall-precision characterisation of a particular IDS is normally used to analyse the relative and absolute performance of an IDS over a range of operating conditions.

6.1 System evaluation

As the purpose of a monitoring structure is to be non-intrusive the theoretical design requirement aimed to meet this objective. The role of the C2s, belief generation, fusion of beliefs, and facilitation of actions via the broker, and communication exchange between monitoring entities, have been implemented in a proof of concept program written in C# conveying the benefits of collaborative intrusion detection. The system is being evaluated in terms of satisfying the aims and objectives as defined in Chapter 1.2, and through successfully accomplishing the design requirements identified in Chapter 3.5 – which defined the characteristics that collaborative intrusion detection solutions for the Cloud environment must possess.

Summary of aims and objectives fulfilment

Our initial project aim was the design of a collaborative Cloud-based framework that focuses on the monitoring and protection of the critical infrastructure services in the Cloud computing environment having identified the limitations within existing intrusion detection techniques. This was achieved throughout Chapters 2 and 3 during which existing work was reviewed and compared with the recent advancements in technology to define the desired requirements (in Chapter 3.5). These were used to tailor and motivate the research direction which formed the base knowledge for establishing Chapter 4 in the application of our Service-based collaborative intrusion detection framework, providing “Security as a Service” for Cloud federations. This also directed the utilisation and extension of D-S to facilitate collaborative intrusion detection via autonomous sharing of information.

Placement of monitoring solutions within Cloud environments were explored, and while optimal placement was deemed the virtual network, the solution can be tailored to differing Cloud services. The methodological monitoring solution and the associated attributes can be tailored to a diverse range of service models as these may comprise similar functionality but can differ depending on the services needed. Additionally, the Service-based collaborative intrusion detection framework supports robustness to attack via a distributed mitigation architecture, making sure the high value processes continue to function regardless of what is occurring elsewhere.

With regards to our project objectives, achievement of these will be detailed subsequently. Our key objective was the design of a critical infrastructure protection framework that focuses on the monitoring and protection of the critical infrastructure services in the Cloud computing environment. This has been achieved by the development of our Service-based collaborative intrusion detection framework for CSPs within a Cloud federation which is detailed in Chapter 4, outlining architectural components and collaborative decision making techniques for intrusion detection. Through this, the objective for demonstrating a solution and technique that can effectively monitor Cloud domains and reliably help to secure and block threats has been met. This has been achieved by the development of a collaborative intrusion detection scheme whereby attributes can communicate and exchange threat information before it could propagate throughout the network.

A further objective was to create a technique to analyse attack data in multiple domains and to reach a decision on its importance, as to whether an anomaly has occurred. This has been fulfilled with the application of D-S theory of evidence being used for belief generation by monitoring entities, and data fusion by the Cloud broker to facilitate collaborative intrusion detection. Limitations of the application of D-S were identified following assessment of whether the methods proposed, and/or the combinations of selected methods, effectively address the issue of infrastructure service protection in the Cloud environment via collaborative intrusion detection. This assessment allowed us to fulfil our objective and has been achieved via the comparison of our extended D-S fusion process to the works of Lo et al., and a high level assessment using our design requirements in Chapter 3.5.

Summary of design requirements evidence

Table 6.2 details the requirements for collaborative intrusion detection in Cloud environments identified in Chapter 3.5 and highlights evidence of fulfilment for each associated attribute. These requirements define the characteristics that collaborative intrusion detection solutions for the Cloud environment must possess. These requirements can therefore be used to assess the suitability of existing solutions, help to ensure the success of the proposed solution, and provide a useful mechanism to evaluate the solution at a high level.

Table 6.2: Design requirements evidence

Design Requirements	Evidence of fulfilment
Accurate	Evidenced by D-S fusion in Chapter 4.5 and experimentation in Chapter 5.
Adaptable	Evidenced by application of theory developed in Chapter 4 to implementation in 4.3.
Collaborative	Evidenced by implementation in Chapter 5.1-5.3 and supported by architectural design in Chapters 4.1 and 4.2
Configurable	Evidenced by implementation in Chapter 5 and supported by architectural design in Chapters 4.1 and 4.2.
Dynamics	Evidenced by algorithm selection in Chapters 4.3-4.6.
Efficient	Evidenced by experiments in Chapters 5.1-5.3.
Graceful degradation of services	Evidenced by experimental analysis in Chapters 5.1-5.3, and supported by theoretical analysis in Chapter 4.
Low maintenance	Evidenced by experimental analysis in Chapter 5.1-5.3, architectural design in Chapter 4.2, and theoretical analysis in Chapter 4.4.
Lightweight	Evidenced by architectural design in Chapters 4.1-4.2, and demonstrated in experimental analysis in Chapter 5.
No prior knowledge	Evidenced by experimental analysis in Chapter 5.3 and application of D-S in Chapters 4.5-4.7.
Novel threats	Evidenced by theoretical analysis in Chapter 4.2 and supported by algorithm design in Chapters 4.3-4.6.
Reliable	Evidenced by experiment analysis in Chapter 5.2, architectural design in Chapters 4.1 and 4.2, and choice of algorithms from Chapters 4.3-4.6.

Scalable	Evidenced by experimental analysis in Chapters 5.1 and 5.2 architectural design in Chapters 4.1 and 4.2, and choice of algorithms from Chapters 4.4-4.7.

The Service-based collaborative intrusion detection framework – "Security as a Service" was designed with fulfilment of these attributes in mind, in order to achieve the key characteristics identified within the literature. Due to the dynamic nature of the Cloud environment it must be able to adapt to changes that occur within the underlying virtualised infrastructure, with considerations to roles, functionality, and structure – which was evidenced by application of theory developed in Chapter 4 and architectural implementation in Chapters 4.1 and 4.2. Graceful degradation of services, evidenced by experimental analysis in Chapters 5.1 – 5.3, and supported by theoretical analysis in 4.4, ensure that if one analysis component stops working, part of the network may stop being monitored, the rest of the monitoring schema can continue working.

Collaboration was a key element also, as within Cloud federations association among different CSPs with the goal of sharing resources and data is essential. The solution was designed to facilitate the exchange of information and alerts within the infrastructure in an autonomous manner, without the need for human intervention. The entities within can take information and share it among key entities, receive and exchange relevant information when polled. Chapter 4.4 highlights these considerations and details the attainment.

The architectural design in Chapters 4.1 and 4.2 convey how each component may be localised to the set of hosts it monitors, and tuned to its specific tasks or characteristics. Taking into consideration the requirements in Chapter 3.5, the ability to automatically scale alongside the expansion of the underlying virtualised infrastructure, in order to adapt for its constantly changing needs was understood and the architecture designed accordingly. It must be able to cope with the adjusting behaviour and adaptive nature of the underlying virtualised infrastructure in order to formulate an accurate understanding. The solution must operate seamlessly in real-time, and be able to manage and verify a large number of resources, and do it effectively and efficiently - which is evidenced by the experiments in Chapter 5.

The solution must possess the ability to detect novel threats, as due to the interconnected nature of Cloud federations, attacks may propagate from within the federation – which is evidenced by theoretical analysis in Chapter 4.2 and supported by algorithm design in 4.3 – 4.6. The operation of the solution should not depend on prior knowledge relating to the system and behaviour, due to the dynamic nature of the underlying environment and information may become outdated. This is achieved via the application of D-S theory of evidence and contribution examined further in Chapter 6.3.2.

It can be concluded that our Service-based collaborative intrusion detection framework – "Security as a Service" has met all of the design requirements identified in Chapter 3.5. It must be noted that this evaluation reflects a high level evaluation of the attributes and configuration of the framework. Additionally, it meets the project aims and objectives identified in Chapter 1.2.

6.2 Dempster-Shafer

Through simulation of the collaborative intrusion detection process, there have been some issues identified when performing in the evidence aggregation. Highlighted by Chen and Aickelin [126], two problems with the application of D-S for detecting intrusions are the computational complexity associated and the conflicting beliefs management. This study focuses on the computational complexity due to the increase of elements in the frame of discernment, but our issue focuses on non-weighted belief generation and the problem that all beliefs generated should have the same level of trust.

However, when performing the belief calculations using two values, based on equation 4.22, the returned result is quite surprising. When comparing two high belief generations, the assumption is that the combined belief value would also be a high number, however it is a lower value. The correlation between high belief values and low fused outputs suggests that the lower the fused output, the higher the risk. The same is understood for two fused low belief values to generate a high fused output, which would be a low risk.

It is not clear if this is due to our calculations but these metrics have been compared on numerous belief fusions and this is a similar occurrence. The mass value must be between 0 and 1 but not inclusive as this seems to skew the calculations, e.g. using a value of 0 would render the combination calculation ($m_{1,2}(A)$) incalculable as you cannot divide by 0 which would be a pertinent value. A coefficient value of 1 would leave the combination calculation having to divide by 0 (1/1-1) which is an impossible calculation. Also, having a coefficient of 0 would give a negative risk output, which is also an unusable value.

With reference to the hypothesis set in Table 4.2 in Chapter 4.5, further examples can be seen below:

Two monitoring entities, based on observation *A* generate beliefs. Monitor 1s BPA include: Attack = 0.9, No Attack = 0.05, Either = 0.05. Monitor 2s BPAs include: Attack = 0.9, No Attack = 0.5, Either = 0.05.

Based on equation 4.23 these beliefs are fused together and performed as below:

M_{12} (A) = (1/1-0.09) x 0.05 x 0.05 = 0.002747252747252

Where *DSK* is calculated by equation 4.24:

DSK = 0.9 x 0.05 + 0.05 x 0.9 = 0.09

The fused belief gives a value of 0.0027. The two BPAs for an attack were 0.9 so it is assumed that when combined, they would generate a value within a similar range. The normalisation phase distributes the weight and treats all evidences equally, so this is definitely a factor when considering application for detecting intrusions.

A further example of D-S belief fusion is two monitoring entities, based on observation *A* generate beliefs. Monitor 1s BPAs include: Attack = 0.5, No Attack = 0.3, Either = 0.2. Monitor 2s BPAs include: Attack = 0.5, No Attack = 0.3, Either = 0.02.

Based on equation 4.23, these beliefs are fused together and performed as below:

M_{12} (A) = (1/1-0.3) x 0.2 x 0.2 = 0.057142857142857

Where *DSK* is calculated by equation 4.24:

DSK = 0.5 x 0.3 + 0.3 x 0.5 = 0.3

The fused beliefs give a value of 0.057. The two BPAs for an attack were 0.5 so this fused value falls within the range expected based on the presented information.

Two monitoring entities, based on observation *A* generate beliefs. Monitor 1s BPA include: Attack = 0.1, No Attack = 0.6, Either = 0.3. Monitor 2s BPAs include: Attack = 0.1, No Attack = 0.6, Either = 0.3.

Using equation 4.24, beliefs were fused together and performed as below:

M_{12} (A) = (1/1-0.12) x 0.3 x 0.3 = 0.10

Where *DSK* is calculated by equation 4.25:

DSK = 0.1 x 0.6 + 0.6 x 0.1 = 0.12

The fused beliefs give a value of 0.10. The two BPAs for Attack were 0.1, so a low fused belief for attack was expected.

When observing generated beliefs and evidences, various factors for each evidential circumstance should be considered. D-S when applied in an autonomous collaborative environment should apply a weight of confidence when the belief generation occurs. If everyone votes ‘no attack’, but there is clearly evidence of an attack on the domain of origin, there should be a way to overrule the decision based on the strength of the associated trust or confidence value associated with the decision. The algorithm is extended further to take this into consideration, and is detailed in Chapter 4.6.

One further issue identified during our research is that of bandwidth consumption, which the IaaS model facilitates during an attack. While there are evident benefits of the utilisation of Cloud federations for CSPs, this interconnection effects the interdependent services within the federation. In the IaaS model, while under attack, it tries to scale out and share resources from other CSPs within the federation. The applicability of our “Security as a Service” mechanism can help limit the reachability of such an attack, and block the attack before it devastates the other CSPs.

The evidence from our research strongly suggests that D-S theory of evidence has grounds for facilitating autonomous sharing of information but there are limitations that need addressing. Our extended D-S approach aims to tackle the issue of conflicting decisions within interconnected infrastructure, but used in a combination with other computing techniques it could have the ability to improve decision making schemas.

6.3 Comparisons to existing work

Our Service-based collaborative intrusion detection framework – “Security as a Service” solution has been designed with the complexities of dynamic Cloud federations in mind by using the information identified from the background research to define the essential characteristics that potential collaborative intrusion detection solutions for the Cloud environment must possess. These requirements were devised by examining the attributes of IDSs and monitoring schemes and their application to Cloud environments.

In order to highlight the progress made by this project within the field of collaborative intrusion detection for Cloud federations, it is imperative that the work conducted throughout this project is compared to that of our colleagues. Intrusion detection in some distributed environments, e.g. federated systems/services/applications, would share similar characteristics to Cloud federations from the intrusion detection point of view. Due to the constraints of our implementation, our comparison to existing work is qualitative.

6.3.1 Cooperative intrusion detection

A cooperative IDS framework for Cloud computing networks is proposed by Lo et al. [105], where they aim to provide secure and reliable services in a Cloud computing environment. One of the security issues identified is how to reduce the impact of DoS or DDoS attacks in this environment via federated defence. To provide such ability, IDSs are employed in each Cloud region and exchange their alerts with each other. Majority voting is used to determine if the alert is warning of an attack based on the number of alerts received by other IDSs.

The majority vote rule is a decision rule that selects alternatives which have a majority, that is, more than half the votes. It is the binary decision rule used most often in influential decision-making bodies, whereby the decisions of the numerical majority of a group will bind on the whole group [150]. They perform a threshold check, i.e. Threshold = $\mu + \lambda \times \sigma$, to determine if traffic has exceeded a predefined value. Cooperative operation is used to receive alert messages delivered from other IDSs. After receiving these alerts, the cooperative agent makes a judgment by executing the majority vote.

It is essential to understand that whilst the work undertaken in previous years was innovative and ahead of its time, the rapid advancements in technology require regular reviews of the systems designed to protect them – hence, it is imperative that our work is compared to that of preceding work. As such, the design review in Chapter 3.5 takes into consideration the existing approaches available to this research area and identifies their inadequacies for application to the nouveau domain of federated Cloud systems. While Lo et al. [105] support the idea of cooperative detection and sharing of information, their solution is elementary in nature when considered for application to federation defence. Our "Security as a Service" solution has adapted and tailored their ideology to a Cloud federation, taking into consideration the aforementioned design requirements.

The work of Lo et al. [105] conveys a monitoring approach where their distributed IDS aggregates data by individual monitoring entities within their architecture. The proposed IDS is constructed of 3 modules: 'Block' – which blocks/drops bad packets, 'Communication' – to send alerts, and 'Cooperation' – to gather alerts and make a decision. An IDS sends an alert while suffering an attack and the premise of their collaborative aspect is to warn the other domains of the potential attack. However, there does not seem to be an action for the IDS under attack – "If the agent, finally, accepts these alerts, the system adds a new blocking rule into the block table against this type of packet on the Cloud computing regions. Therefore, by this new blocking rule, the Cloud computing regions except the victim one can avoid this type of attack." Our approach in comparison to this has extended the D-S fusion process to include a two stage process to protect the domain of origin, when conflicting belief generation deems the aggregated decision erroneous.

Their use of the adaptive threshold algorithms to detect traffic violations claims accuracy for DoS and DDoS attacks, but this algorithm considers only violations of the threshold which can be quite strict in nature. The Cloud environment is dynamic in nature and more adaptive algorithms have been determined more applicable for such a complex infrastructure. The use of CUSUM and EWMA to adapt to the underlying infrastructure shows how our approach considers the excess volume sent above the normal volume, accounting for the intensity of the violations, and achieving higher accuracy as a combination of both algorithms calculates and re-establishes thresholds.

Additionally, they do not use the resources of the Cloud to protect against these network attacks, but rather focus on their exchange of messages to cross domains. Cooperative agents send/receive alerts from IDSs within other Cloud regions but there is no clear information hierarchy. If there are n sensors detecting intrusions, then $n\ x\ (n-1)$ are exchanged and the associated communications costs would be high [63]. Chapter 4.4 details our hierarchical communication schema that utilises the resources and scale of the Cloud environment for improved communication exchange.

6.3.2 D-S for intrusion detection

Other comparable work within this area proposing the use of D-S within their research does not perform D-S belief fusion, rather they state the application and processes of the fusion stages. Alem et al. [151] in their work on belief functions attempt to improve the efficiency of IDS using D-S. They use D-S in combination with SVM and naïve Bayes classifiers, and fuzzy logic. While they present their results in terms of identifying attacks, their experiments do not detail how they use D-S in belief generation or for data fusion. Their use of training data, and naïve Bayes, is an alternative to the usual stance of no priori knowledge being required when using D-S. Their application of D-S is high level and our mathematical fusion or proof of concept application to Cloud federations cannot be compared with this approach.

Nguyen et al. [152] propose approaches for dealing with conflicting mass values when combining beliefs, however this is mainly an issue when dealing with a large numbers of variables in a hypothesis set. Conflicting ratings occur due to the diversity of users, and this could occur when applying D-S to a wider range of possibility observations. D-S for intrusion detection would utilise BPAs of { }, {attack}, {no attack}, and {attack, no attack}. { } represents an empty subset with a value of 0, that corresponds to “no solution”. Whereas {attack, no attack} represents uncertainty, i.e. it could be either. The associated hypothesis sets for application for intrusion detection would include a smaller number of variables, and conflicting issues could be targeted by the inclusion of SLAs specifying levels of trust or reputation based schemas.

Hu et al. [129] convey applicability of D-S for intrusion detection. Like our reasoning, the choice of D-S is due to the fact that there is no need for state transition matrices and training data to make uncertainty inferences. However, they include “Detection uncertainty” in combination with D-S to accord for uncertainty or ignorance with an event. The objective uncertainty varies with the degree of detection and can affect the observational results in the combination module. While they present some experimental application of their approach, it is unclear how D-S solves the issue of uncertainty. They do conclude that D-S has a ‘great development prospect in the future’. They apply D-S in principle but have no mathematical application to support their methodology, as such our D-S arithmetic and extended D-S fusion process cannot be compared.

Siaterlis et al. [153] propose the use of D-S’s theory of evidence as the underlying data fusion model for creating a DDoS detection engine. They state that the modelling strength of the mathematical notation as well as the ability to take into account knowledge gathered from totally heterogeneous information sources were some of the advantages of using D-S theory. They have demonstrated their idea by developing a prototype that consists of a Snort pre-processor-plugin and a Simple Network Management Protocol (SNMP) data collector that provide the necessary input that through heuristics feeds the D-S inference engine.

In terms of the comparison, it is clear that the application of D-S to the field of intrusion detection has its merits, however for the facilitation of collaboration among monitoring components the complexity is increased. D-S offers an alternative to the traditional probabilistic theory for the mathematical representation of uncertainty, and for collaborative intrusion detection its main attribute is data fusion. The main advantage of D-S is that no priori knowledge of the system is required, thus making it suitable for anomaly detection of previously unseen information, however including heuristics or neural networks or other machine learning techniques in order to improve the accuracy of results have been considered [62].

Experimental results convey the evident benefits of D-S application to Cloud federations, but there are limitations due to weighted belief generation and the normalisation factor when combining the evidences – which can lead to conflicting beliefs. If the fused decision is “No Attack” but the belief of origin has a high confidence value, then the domain of origin would take action against the suspect observation and send the belief value to the broker to store in its local Grey list. Our extended D-S theory of evidence fusion process takes this into consideration and conveys this. The inclusion of trust based schemes, or reputation based scoring inspired by WSN could be utilised also in this case [154], but these are considered for future works.

6.4 Summary

This chapter has evaluated the Service-based collaborative intrusion detection architecture and its fundamental components. As these results demonstrate, our solution facilitates collaborative intrusion detection for federated Cloud services. The overall system was tested for effectiveness and efficiency required by intrusion detection in Cloud environments. Our testing illustrates the benefits of collaborative intrusion detection, and in particular autonomous sharing of information. Our Service-based collaborative intrusion detection framework – "Security as a Service", used in conjunction with D-S and confidence values could be warning of an imminent attack with a Cloud federation.

Additionally, how inter-domain cooperation for holistic security and improved resilience is achievable in our environment is conveyed. Federated Cloud environments are growing areas in terms of adoption by critical infrastructure vendors, and large corporations, so our Service-based framework facilitates this collaborative intrusion detection, and sharing of attack information among this differing populace – which in turn can help improve the Cloud environments' overall resilience to an attack.

Chapter 7

Conclusions and future developments

Cloud computing has emerged as a way to enable content providers to meet their application needs through either Cloud development environments or through outsourced CSPs. Adoption of Cloud computing services allows critical infrastructure vendors to benefit from dynamic resource allocation for managing unpredictable load peaks, storing of historical process data (either on site in a private Cloud, or sharing among other related vendors in a hybrid Cloud), federating into a larger Cloud, and large scale data analytics based on historical data of consumers, to name a few.

Given the public awareness of critical infrastructure and their importance, the public wants to be assured that these systems are built to function in a secure manner and any technological utilisation are secure; maintaining the confidentiality, integrity and availability of their data or services. Cloud outages are unexpected events that occur within a Cloud infrastructure and consequently affect the operations and availability of services placed within the Cloud. Should this be critical infrastructure services, or important historical processes, this would be damaging for the reputation of the infrastructure vendor, or for the CSP. Recognising the signs of an attack quickly, and being able to limit the effect on operation is imperative.

When cyber-attacks and cyber disruptions happen, millions of users are potentially affected. A cyber disruption in this context means a temporary or permanent loss of service, and users of the Cloud service who rely on its continuity are affected. Design of a collaborative Cloud-based framework that focuses on the monitoring and protection of the critical infrastructure services in the Cloud computing environment is the main contribution of our work. Collaboration among CSPs could ensure that they are up to date on different Cloud threats and emerging vulnerabilities, as they are interdependent upon each other at times within the federation, i.e. load balancing at peak times – one drawback from this interconnection is the increase in eDoS attacks. The effects of eDoS/DDoS attacks can span from the loss of some data, to the potential isolation of parts of the federation. Protecting the federated Cloud against cyber-attacks is a key concern, since there are potentially significant economic consequences.

The evidence from our research strongly suggests that using D-S's theory of evidence for belief generation, in combination with the application of confidence values, can help improve the accuracy of generated observations for autonomous sharing of information. The application of D-S to Cloud federations and the inventive D-S extension presented in this thesis are encouraging and show that the area of collaborative intrusion detection in federated Cloud environments is worth pursuing further. The methodology and architectural monitoring structure presented could also be adopted and expanded for application in areas such as WSN, distributed systems, or the Internet of Things (IoT), as autonomous sharing of information is a promising area.

Comparing the solution for distributed systems could involve adapting the hierarchical C2/MN/SN structure to the working of master/slave architecture model of distributed computing architecture where the master node has unidirectional control over one or more slave nodes. In this instance, the task(s) are distributed by the master node to the configured slaves and the results are returned to the master node. Distributed systems and Cloud computing environments slightly refer to different things, however the underlying concept between them is same. However for application to such interconnected domains where local views may vary, machine learning techniques may improve the accuracy of detection.

Observations from different CSPs are correlated autonomously, in order to determine whether similar behaviour that is indicative of an attack or other issues has been observed in their domains. The integration of the decisions coming from different IDSs has emerged as a technique that could strengthen the final decision. Federated Cloud environments are growing areas in terms of adoption by critical infrastructure vendors, and large corporations, so our Service-based collaborative intrusion detection framework facilities this collaborative intrusion detection, and sharing of attack information among these different CSPs. Future efforts to integrate assurance and auditing tools to ensure policy assurance among involved entities are needed also.

7.1 Summary of novel contributions

The research presented in this thesis presents the principles, techniques, and algorithms that can be adapted from other distributed computing paradigms to the development of a collaborative intrusion detection framework that can detect and prevent intrusion in collaborative domains. This features a novel use of the D-S algorithm to detect intrusions and send alerts. To ensure our contribution, D-S evaluates evidence from multiple domains and fuses the beliefs to establish whether an attack is occurring, has occurred or will occur.

The main novel contribution of this project is that it provides the means by which DDoS attacks are detected within a Cloud federation, so as to enable an early propagated response to block the attack, particularly by the interdependent CSPs within the federation. As such, this is effectively inter-domain cooperation as these CSPs will cooperate with each other to offer holistic security, and add to the defence in depth. The D-S theory of evidence is used to facilitate this autonomous sharing of information, and to fuse the generated beliefs to form a system-wide decision.

Providing service-based collaborative intrusion detection – “Security as a Service” in a Cloud federation is achieved via the following novel contributions:

- A collaborative intrusion detection framework that can detect and prevent intrusion in Cloud federations and/or collaborative domains in real-time via the autonomous sharing of information. This features a novel application of the D-S theory of evidence algorithm to detect intrusions and fuse generated beliefs for collaborative intrusion detection, and an extension of D-S to include confidence values for conflicting decisions. There is no current solution that can provide adequate protection for Cloud federations, or identified solution which implements the D-S algorithm and produces collaborative decisions in Cloud federations to improve upon the Cloud security contribution.

- Within related work, the accuracy and efficiency of detection is important, but ensuring the solution is scalable and can deal with large volumes of logs from different sources is problematic. The Cloud Broker coordinates attack responses, both within the domain itself, and with other domains, and is facilitating inter-domain cooperation. D-S is used to fuse the generated beliefs and make a system-wide decision. This cooperation between CSPs ensures that the scalable defence required against DDoS attacks is in an efficient manner; aiming to improve the overall resilience of the interconnected infrastructure.

- CSPs within a Cloud federation are represented as interconnected domains, and can trace attacks back to their domain source by dynamically provisioning the infrastructure to react – CUSUM and EWMA algorithms are used for adaptive threshold calculation. A local propagation mechanism collects statistics at a local level via MNs, and in order to minimise detection delay and reduce the communication overhead, this is propagated among MNs using a gossip protocol. As Cloud environments, and Cloud federations, are large scale, it is essential that any potential solution should scale alongside the environment and have the potential to expand and scale considerably without any issues or performance implications.

7.2 Thesis Summary

In this section, an overview of the thesis and our research for the design of a collaborative Cloud-based framework that focuses on the monitoring and protection of the critical infrastructure services in the Cloud computing environment is presented, in addition to a summary of each of the thesis chapters.

In Chapter 1, the research area and our motivation behind the work being undertaken is introduced. The aims, objectives, and novel contributions of the research are discussed, in addition to a high level overview of the solution. Publications that have resulted from the research undertaken are highlighted also.

In Chapter 2, background on critical infrastructure and Cloud computing is provided, in addition to the evident security vulnerabilities and threats that need to be resolved is highlighted. Critical infrastructure and Cloud utilisation requirements are highlighted, and the need for Cloud-based protection mechanisms for services in Cloud federations.

In Chapter 3, related works in this area are discussed. An overview of the different types of protection and preventative measures in place for intrusion detection in the Cloud environment is detailed. Existing approaches are analysed and an observation of their evident inadequacies presented. This information is used to determine specific requirements for proposing a novel solution for this area.

Chapter 4 outlines the design and architectural details for the novel solution, in addition to the algorithms and methods utilised. The entities present and their functionality are explained, and requirements and considerations for the design are conveyed. The improved D-S two stage intrusion detection method is outlined.

Chapter 5 outlines the implementation details of the intrusion detection framework for federated Cloud environments. Through the use of Riverbed Modeler 18.0, the effects of DDoS attacks to Cloud federations are conveyed and analysed, and collaborative intrusion detection using D-S theory of evidence is demonstrated using a proof of concept developed in C#.

In Chapter 6, the novel solution is evaluated against the metrics we consider to be the most important for determining the efficiency of a collaborative IDS. The solution was evaluated using the requirements established in 3.5 which define the characteristics which collaborative intrusion detection solutions must possess, and the approach is also compared to related work within this research field.

Conclusions and future work are presented in Chapter 7. Within this chapter, a thesis summary is provided, a summary of the project novelties, and some final concluding remarks. This chapter will also highlight how the work can be built on for future research projects.

7.3 Further Work

Further work would involve testing the Service-based collaborative intrusion detection methodology on a real Cloud-based test bed to compare. While the D-S algorithm has been deemed as a scalable algorithm, it would be advantageous to compare the flexibility of the algorithm and the Service-based collaborative intrusion detection framework to see how the effects of the information exchange affect the network. The complex nature of process models within Riverbed Modeler 18.0 meant the effects of the algorithm to the network couldn't be modelled effectively, so using a Cloud-based test bed would be an interesting option.

Adapting the Service-based collaborative intrusion detection architecture to focus on a wider range of attacks would be a further enhancement. For simulation purposes, our application of D-S for collaborative intrusion detection focused on demonstrating the applicability of the algorithm – IP addresses were used as the information that was exchanged. However tailoring the methodology to include attack signatures or measurable profile information would expand the capabilities of the approach. The main advantage of D-S is that no priori knowledge of the system is required, thus making it suitable for anomaly detection of previously unseen information, however including heuristics or neural networks or other machine learning techniques in order to improve the accuracy of results has been considered.

A further consideration for expansion is adapting the solution to make it a dual level IDS, where there is a personalised experience for both CSP and Cloud users. Access control can help tailor the IDS to suit their needs. Cloud consumers should not only have to depend on the CSPs' security infrastructure. They need to be able to monitor and protect their own virtual existence by enforcing additional security methods with other network security technologies. Giving the user the ability to personalise their own security needs could be a beneficial enhancement via SLA negotiation.

The methodology proposed could also be applied to WSN environments, or scaled considerably to tailor to the Internet of Things (IoT). The IoT is a concept coined to cover the interconnected infrastructure and utilities that are increasingly occurring. IoT is the interconnection of uniquely identifiable embedded computing devices within the existing Internet infrastructure, from smart meters in homes, remote sensors for gas and oil utilities, interdependent system-of-systems: but the key issue is the fundamental problem with the interconnection of this "Internet of Things". They are creating a wider attack surface with billions of new devices. IoT inherits the same monitoring requirements from Cloud, but the related challenges are further affected by volume, variety, and velocity characteristics of IoT. Application of our Security of a Service methodology to the IoT environment could provide an interesting testbed for collaborative information exchange.

The IoT does not replace the existing ICT or operational technology networks; rather, it enhances these networks and relies on them in many ways. Recognising all these aspects working together, cyber security and physical security solutions must also work together with a coordinated focus on threats. With an estimated number of 50 billion devices that will be networked by 2020, specific attention must be paid to transportation, storage, access, and processing of the huge amount of data they will produce [155]. Processing large quantities of IoT data will increase as a proportion of workloads of data centres, leaving providers facing new security, capacity and analytics challenges. Handling this data conveniently is a critical challenge, as the overall application performance is highly dependent on the properties of the data management service.

7.4 Concluding Remarks

In this thesis, a novel method for detecting intrusions in a federated Cloud environment has been presented. In the related work section, an extensive survey of existing intrusion detection methods was conducted, identifying the limitations in their scalability and efficiency. From this, a methodology for detecting intrusions in the federated Cloud environment was constructed, improving upon the inefficiencies of current approaches, and tackling the problem with lightweight collaborative decision making. This inspired the design and implementation of the Service-based collaborative intrusion detection architecture where CSPs collaborate for holistic security. The "Security as a Service" entity is present in each CSP's domain "as-a-service" and automatically reacts to deviations from normal behaviour. The design of this solution is small in scale for proof of concept but can embrace the resources and scale of the Cloud environment. Through experimentation, the solution and associated attributes have been validated, demonstrating that they can overcome the challenges for protecting critical infrastructure services in the Cloud environment through collaborative intrusion detection. The evidence from the research strongly suggests that fusion algorithms can play a key role in autonomous decision making schemes, however our experimentation highlights areas upon which improvements are needed before fully applying to federated environments.

References

[1] W. Hurst, M. Merabti, and P. Fergus, “Towards a Framework for Operational Support in Critical Infrastructures,” in The 15th Post Graduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting (PGNet2014), 2014, pp. 1–4.

[2] Department of Homeland Security, “Configuring and Managing Remote Access for Industrial Control Systems - CPNI Centre for the Protection Of National Infrastructure,” Washington DC, USA, 2010.

[3] Trend Micro Incorporated, “Report on Cybersecurity and Critical Infrastructure in the Americas,” Washington DC, USA, 2015.

[4] M. A. C. Dekker and ENISA, “Critical Cloud Computing-A CIIP perspective on cloud computing services,” Greece, 2012.

[5] G. D. C. Rodrigues, G. L. Dos Santos, V. T. Guimaraes, L. Z. Granville, and L. M. R. Tarouco, “An Architecture to Evaluate Scalability, Adaptability and Accuracy in Cloud Monitoring Systems,” in *The International Conference on Information Networking 2014 (ICOIN2014)*, 2014, pp. 46–51.

[6] Á. MacDermott, Q. Shi, M. Merabti, and K. Kifayat, “Hosting critical infrastructure services in the cloud environment considerations,” *International Journal of Critical Infrastructures*, vol. 11, no. 4, pp. 365–381, 2015.

[7] B. Genge, P. Haller, and I. Kiss, “A framework for designing resilient distributed intrusion detection systems for critical infrastructures,” *International Journal of Critical Infrastructure Protection*, vol. 15, pp. 3–11, 2016.

[8] C. B. Westphall *et al.*, “Operation, Management, Security and Sustainability for Cloud Computing,” *Information Systems of the FSMA*, vol. 13, pp. 30–50, 2014.

[9] S. Paudel and M. Tauber, “Security Standards Taxonomy for Cloud Applications in Critical Infrastructure IT,” in *8th International Conference for Internet Technology and*

Secured Transactions (ICITST), 2013, pp. 645–646.

[10] C. W. Ten, G. Manimaran, and C. C. Liu, “Cybersecurity for critical infrastructures: attack and defense modeling,” *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 40, no. 4, pp. 853–865, 2010.

[11] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, “A survey of intrusion detection techniques in Cloud,” *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 42–57, Jun. 2012.

[12] Riverbed Technology Inc., “Riverbed Modeler,” *Riverbed Support*, 2017. .

[13] T. Miyachi, H. Narita, H. Yamada, and H. Furuta, “Myth and reality on control system security revealed by Stuxnet,” in *2011 Proceedings of Society of Instrument and Control Engineers (SICE) Annual Conference*, 2011, pp. 1537–1540.

[14] McAfee Labs, “2012 Threats Predictions,” Santa Clara, CA 95054, 2012.

[15] Á. MacDermott, Q. Shi, M. Merabti, and K. Kifayat, “Detecting Intrusions in the Cloud Environment,” in *14th Annual Postgraduate Symposium on Convergence of Telecommunications, Networking and Broadcasting (PGNet 2013)*, 2013, pp. 336–343.

[16] CPNI, “Critical National Infrastructure,” 2017. [Online]. Available: https://www.cpni.gov.uk/critical-national-infrastructure-0. [Accessed: 18-May-2017].

[17] K. E. Lever, Á. MacDermott, and K. Kifayat, “Evaluating Interdependencies and Cascading Failures Using Distributed Attack Graph Generation Methods for Critical Infrastructure Defence,” in *IEEE 8th International Conference on Developments in eSystems Engineering (DeSE 2015)*, 2015, pp. 47–52.

[18] M. Chander, “Protection of National Critical Infrastructure,” *Defence and Security Alert*, pp. 54–58, 2013.

[19] S. M. Rinaldi, J. P. Peerenboom, and T. K. Kelly, “Identifying, Understanding, and

Analyzing Critical Infrastructure Interdependencies," *IEEE Control Systems Magazine*, vol. 21, no. 6, pp. 11–25, 2001.

[20] R. Brewer, "Protecting critical control systems," *Network Security*, vol. 3, pp. 7–10, Mar. 2012.

[21] E. J. Byres, M. Franz, and D. Miller, "The Use of Attack Trees in Assessing Vulnerabilities in SCADA Systems," in *IEEE International Infrastructure Survivability Workshop (IISW'04)*, 2004, pp. 1–9.

[22] W. Hurst, M. Merabti, and P. Fergus, "Behavioural analysis for supporting critical infrastructure security," in *14th Annual Postgraduate Symposium on Convergence of Telecommunications, Networking and Broadcasting (PGNet 2013)*, 2013.

[23] I. P. Chochliouros and OTE, "Discussion on the Challenges for the Development of a Context for: SEcure Cloud computing for CRitical infrastructure IT ('SECCRIT')," Greece, 2012.

[24] M. Scholler, M. Stiemerling, and A. Ripke, "Resilient deployment of virtual network functions," in *5th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, 2013, pp. 208–214.

[25] M. Florian, S. Paudel, and M. Tauber, "Trustworthy Evidence Gathering Mechanism for Multilayer Cloud Compliance," in *8th International Conference for Internet Technology and Secured Transactions (ICITST)*, 2013, pp. 529–530.

[26] T. Bernard, M. Baruthio, C. Steinmetz, and J.-M. Weber, "Cloud-based event detection platform for water distribution networks using machine-learning algorithms," in *Machine Learning for Cyber Physical Systems (Technologies for Intelligent Automation)*, 2017, pp. 35–43.

[27] S. Iqbal, L. Mat, B. Dhaghighi, and M. Hussain, "On cloud security attacks: A taxonomy and intrusion detection and prevention as a service," *Journal of Network and Computer Applications*, vol. 74, pp. 98–120, 2016.

[28] B. Zhu and S. Sastry, “SCADA-specific intrusion detection/prevention systems: a survey and taxonomy,” *Proceedings of the 1st Workshop on Secure Control Systems (SCS)*, vol. 11, pp. 1–16, 2010.

[29] I. N. Fovino, A. Carcano, T. D. L. Murel, A. Trombetta, and M. Masera, “Modbus/DNP3 State-Based Intrusion Detection System,” in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, 2010, pp. 729–736.

[30] J. Verba, “Idaho national laboratory supervisory control and data acquisition intrusion detection system (SCADA IDS),” in *International Conference on Technologies for Homeland Security (HST)*, 2008, no. 208, pp. 469–473.

[31] L. Briesemeister, S. Cheung, U. Lindqvist, and A. Valdes, “Detection, correlation, and visualization of attacks against critical infrastructure systems,” in *2010 Eighth Annual International Conference on Privacy Security and Trust (PST)*, 2010, pp. 15–22.

[32] A. Carcano, A. Coletta, M. Guglielmi, M. Masera, I. Nai Fovino, and A. Trombetta, “A Multidimensional Critical State Analysis for Detecting Intrusions in SCADA Systems,” *IEEE Transactions on Industrial Informatics*, vol. 7, no. 2, pp. 179–186, 2003.

[33] S. L. Scott, “A Bayesian paradigm for designing intrusion detection systems,” *Computational Statistics & Data Analysis*, vol. 45, no. 1, pp. 69–83, Feb. 2004.

[34] A. Califano, E. Dincelli, and S. Goel, “Using Features of Cloud Computing to Defend Smart Grid against DDoS Attacks,” in *10th Annual Symposium on Information Assurance (ASIA ’15)*, 2015, pp. 44–49.

[35] R. Bhadauria, R. Chaki, N. Chaki, and S. Sanyal, “A Survey on Security Issues in Cloud Computing,” in *IEEE Communications Surveys and Tutorials*, 2011, pp. 1–15.

[36] T. Steiner, H. Khiabani, and SANS Institute, “An Introduction To Securing a Cloud Environment,” Swansea, 2012.

[37] K. Dahbur, B. Mohammad, and A. B. Tarakji, “A Survey of Risks, Threats and

Vulnerabilities in Cloud Computing," in *Proceedings of the 2011 International Conference on Intelligent Semantic Web-Services and Applications - ISWSA '11*, 2011, pp. 1–6.

[38] P. Mell and T. Grance, "The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology," Gaithersburg, MD 20899-8930, 2011.

[39] R. Samani, "Cybercrime Exposed White Paper," Santa Clara, CA 95054, 2013.

[40] K. Hwang and D. Li, "Trusted Cloud Computing with Secure Resources and Data Coloring," *IEEE Internet Computing*, vol. 14, no. 5, pp. 14–22, Sep. 2010.

[41] Á. MacDermott, Q. Shi, M. Merabti, and K. Kifiyat, "Considering an elastic scaling model for cloud security," in *The 8th International Conference for Internet Technology and Secured Transactions (ICITST-2013)*, 2013, pp. 150–155.

[42] R. Bhadauria and S. Sanyal, "Survey on Security Issues in Cloud Computing and Associated Mitigation Techniques," *International Journal of Computer Applications*, vol. 47, no. 18, pp. 47–66, Jun. 2012.

[43] KPMG International, "The Cloud Changing the Business Ecosystem," India, 2011.

[44] M. Ficco, L. Tasquier, and R. Aversa, "Intrusion Detection in Cloud Computing," in *2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, 2013, pp. 276–283.

[45] O. Babaoglu, M. Tamburini, and U. Bologna, "Design and Implementation of a P2P Cloud System," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, 2012, pp. 412–417.

[46] M. Rak *et al.*, "Security Issues in Cloud Federations," in *Achieving Federated and Self-Manageable Cloud Infrastructures: Theory and Practice*, 2012, pp. 176–194.

[47] D. Villegas *et al.*, "Cloud federation in a layered service model," *Journal of Computer and System Sciences*, vol. 78, no. 5, pp. 1330–1344, Sep. 2012.

[48] Á. MacDermott, Q. Shi, and K. Kifayat, "Collaborative Intrusion Detection in Federated Cloud Environments," *Journal of Computer Sciences and Applications on Big Data Analytics in Intelligent Systems*, vol. 3, no. 3A, pp. 10–20, 2015.

[49] V. Chang *et al.*, "Federated cloud resource management: Review and discussion," *Journal of Network and Computer Applications*, vol. 77, no. October 2016, pp. 87–105, 2017.

[50] N. Gruschka and M. Jensen, "Attack Surfaces: A Taxonomy for Attacks on Cloud Services," in *2010 IEEE 3rd International Conference on Cloud Computing*, 2010, pp. 276–279.

[51] N. Kumar and S. Sharma, "Study of Intrusion Detection System for DDoS Attacks in Cloud Computing," in *2013 Tenth International Conference on Wireless and Optical Communications Networks (WOCN)*, 2013, pp. 1–5.

[52] Á. MacDermott, Q. Shi, and K. Kifayat, "Detecting Intrusions in Federated Cloud Environments Using Security as a Service," in *IEEE 8th International Conference on Developments in eSystems Engineering (DeSE 2015)*, 2015, pp. 91–96.

[53] J. Haggerty, Q. Shi, and M. Merabti, "Beyond the perimeter: the need for early detection of denial of service attacks," in *18th Annual Computer Security Applications Conference*, 2002, pp. 413–423.

[54] VM Ware Inc., "Securing the Cloud a Review of Cloud Computing, Security Implications and Best Practices," *Tech Republic, Whitepaper*, 2003. [Online]. Available: http://www.techrepublic.com/resource-library/whitepapers/securing-the-cloud-a-review-of-cloud-computing-security-implications-and-best-practices-copy1/. [Accessed: 25-Jul-2013].

[55] M. Ali, S. U. Khan, and A. V. Vasilakos, "Security in cloud computing: Opportunities

and challenges," *Information Sciences*, vol. 305, pp. 357–383, 2015.

[56] A. Patel, Q. Qassim, and C. Wills, "A survey of intrusion detection and prevention systems," *Information Management & Computer Security*, vol. 18, no. 4, pp. 277–290, 2010.

[57] C. Thomas and B. Narayanaswamy, "Sensor Fusion for Enhancement in Intrusion Detection," in *Sensor Fusion - Foundation and Applications*, 2011, pp. 61–76.

[58] N. Patil, C. Das, S. Patankar, and K. Pol, "Analysis of Distributed Intrusion Detection Systems Using Mobile Agents," in *First International Conference on Emerging Trends in Engineering and Technology*, 2008, pp. 1255–1260.

[59] F. Sabahi and A. Movaghar, "Intrusion Detection: A Survey," in *The Third International Conference on Systems and Networks Communications*, 2008, pp. 23–26.

[60] J. F. C. Joseph, A. Das, B.-C. Seet, and B.-S. Lee, "Opening the Pandora's Box: Exploring the fundamental limitations of designing intrusion detection for MANET routing attacks," *Computer Communications*, vol. 31, no. 14, pp. 3178–3189, Sep. 2008.

[61] J. Peng, K. K. R. Choo, and H. Ashman, "User profiling in intrusion detection: A review," *Journal of Network and Computer Applications*, vol. 72, pp. 14–27, 2016.

[62] J. Bin, X. Huang, R. Liu, and Y. Ma, "A DDoS Attack Detection Method Based on Hybrid Heterogeneous Multiclassifier Ensemble Learning," *Journal of Electrical and Computer Engineering*, vol. 2017, pp. 1–10, 2017.

[63] K. A and C. N. Modi, "An Efficient Security Framework to Detect Intrusions at Virtual Network Layer of Cloud Computing," in *19th International ICIN Conference - Innovations in Clouds, Internet and Networks*, 2016, pp. 133–140.

[64] Cloud Security Alliance, "The Treacherous 12: Cloud Computing Top Threats in 2016," USA, 2016.

[65] W. Xin, H. Ting-lei, and L. Xiao-yu, "Research on the Intrusion detection mechanism based on cloud computing," in *Intelligent Computing and Intelligent Systems (ICIS)*, 2010, pp. 125–128.

[66] T. V. S. Jeganathan and T. A. Prakasam, "Secure the Cloud Computing Environment from Attackers using Intrusion Detection System," *International Journal of Advanced Research in Computer Science & Technology (IJARCST 2014)*, vol. 2, no. 2, pp. 181–186, 2014.

[67] D. Parwani, A. Dutta, P. Kumar Shulka, and M. Tahilyani, "Various Techniques of DDoS Attacks Detection and Prevention at Cloud: A Survey," *Oriental Journal of Computer Science and Technology*, vol. 8, no. 2, pp. 110–120, 2015.

[68] C. H. Ling, W. F. Hsien, and H. Min Shiang, "A Double Circular Chain Intrusion Detection for Cloud Computing Based on AdjointVM approach," *International Journal of Network Security*, vol. 18, no. 2, pp. 397–400, 2016.

[69] S. Gupta, P. Kumar, and A. Abraham, "A Profile Based Network Intrusion Detection and Prevention System for Securing Cloud Environment," *International Journal of Distributed Sensor Networks*, vol. 2013, no. 2, pp. 1–12, 2013.

[70] R. Devi, R. Kumar Jha, A. Gupta, S. Jain, and P. Kumar, "Implementation of Intrusion Detection System using Adaptive Neuro-Fuzzy Inference System for 5G wireless communication network," *AEU-International Journal of Electronics and Communications*, vol. 74, pp. 94–106, 2017.

[71] S. Y. (George) ; Ho, "Intrusion Detection - Systems for Today and Tomorrow," Swansea, 2017.

[72] A. Sahasrabuddhe, S. Naikade, A. Ramaswamy, B. Sadliwala, and P. Futane, "Survey on Intrusion Detection System using Data Mining Techniques," *International Research Journal of Engineering and Technology (IRJET)*, vol. 4, no. 5, pp. 1780–1784, 2017.

[73] A. Kumar Dalai and S. Kumar Jena, "Hybrid Network Intrusion Detection Systems: A

Decade's Perspective," in *Proceedings of the International Conference on Signal, Networks, Computing, and Systems. Lecture Notes in Electrical Engineering*, 2017, pp. 341–349.

[74] H. Hamad and M. Al-Hoby, "Managing Intrusion Detection as a Service in Cloud Networks," *International Journal of Computer Applications*, vol. 41, no. 1, pp. 35–40, Mar. 2012.

[75] T. Bui, D. Hernandez-Lobato, J. Hernandez-Lobato, Y. ; Li, and R. Turner, "Deep Gaussian Processes for Regression using Approximate Expectation Propagation," in *Proceedings of The 33rd International Conference on Machine Learning (PMLR 48)*, 2016, pp. 1472–1481.

[76] S. N. Dhage and B. B. Meshram, "Intrusion detection system in cloud computing environment," *International Journal of Cloud Computing*, vol. 1, no. 2/3, pp. 261–282, 2012.

[77] H. M. Alsafi, W. M. Abduallah, and A. K. Pathan, "IDPS : An Integrated Intrusion Handling Model for Cloud Computing Environment," *International Journal of Computing and Information Technology (IJCIT)*, vol. eprint arX, pp. 1–18, 2012.

[78] G. Hancock, DL, Lamont, "Multi agent system for network attack classification using flow-based intrusion detection," *IEEE Congress on Evolutionary Computation (CEC)*, pp. 1535–1542, 2011.

[79] W. Yu, C. Xiaohui, and W. Sheng, "Anomaly Network Detection Model Based on Mobile Agent," in *2011 Third International Conference on Measuring Technology and Mechatronics Automation*, 2011, pp. 504–507.

[80] S. Shamshirband, N. B. Anuar, M. L. M. Kiah, and A. Patel, "An appraisal and design of a multi-agent system based cooperative wireless intrusion detection computational intelligence technique," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 9, pp. 2105–2127, Oct. 2013.

[81] G. Bourkache, M. Mezghiche, and K. Tamine, “A Distributed Intrusion Detection Model Based on a Society of Intelligent Mobile Agents for Ad Hoc Network,” *2011 Sixth International Conference on Availability, Reliability and Security*, pp. 569–572, Aug. 2011.

[82] W. Jansen, “Intrusion detection with mobile agents,” *Computer Communications*, vol. 25, no. 15, pp. 1392–1401, Sep. 2002.

[83] D. L. Hancock, “Multi agent systems on military networks,” *2011 IEEE Symposium on Computational Intelligence in Cyber Security (CICS)*, pp. 100–107, 2011.

[84] J. Povedano-Molina, J. M. Lopez-Vega, J. M. Lopez-Soler, A. Corradi, and L. Foschini, “DARGOS: A highly adaptable and scalable monitoring architecture for multi-tenant Clouds,” *Future Generation Computer Systems*, vol. 29, no. 8, pp. 2041–2056, 2013.

[85] Á. MacDermott, Q. Shi, M. Merabti, and K. Kifayat, “An elastic scaling method for cloud security,” *Journal of Internet Technology and Secured Transactions (JITST)*, vol. 3, no. 3/4, pp. 254–262, 2014.

[86] A. Patel, M. Taghavi, K. Bakhtiyari, and J. Celestino Júnior, “An intrusion detection and prevention system in cloud computing: A systematic review,” *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 25–41, Jan. 2013.

[87] H. A. Kholidy and F. Baiardi, “CIDD: A Cloud Intrusion Detection Dataset for Cloud Computing and Masquerade Attacks,” in *2012 Ninth International Conference on Information Technology - New Generations*, 2012, pp. 397–402.

[88] R. Shea, S. Member, J. Liu, and S. Member, “Performance of Virtual Machines Under Networked Denial of Service Attacks : Experiments and Analysis,” *IEEE Systems Journal*, vol. 7, no. 2, pp. 335–345, 2013.

[89] A. Chonka and J. Abawajy, “Detecting and Mitigating HX-DoS Attacks against Cloud Web Services,” in *15th International Conference on Network-Based Information Systems*, 2012, pp. 429–434.

[90] M. Mehdi, S. Zair, A. Anou, and M. Bensebti, “A Bayesian Networks in Intrusion Detection Systems,” *Journal of Computer Science*, vol. 3, no. 5, pp. 259–265, 2007.

[91] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A Survey,” *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, Jul. 2009.

[92] C. Turner, R. Jeremiah, D. Richards, and A. Joseph, “A Rule Status Monitoring Algorithm for Rule-Based Intrusion Detection and Prevention Systems,” *Procedia Computer Science*, vol. 95, pp. 361–368, 2016.

[93] T. H.-J. Kim, K. Brancik, D. Dickinson, a. Perrig, and B. Sinopoli, “Cyber–Physical Security of a Smart Grid Infrastructure,” *Proceedings of the IEEE*, vol. 100, no. 1, pp. 195–209, Jan. 2012.

[94] R. Aamir Raza Ashfaq, X.-Z. Wang, J. Zhexue Huang, H. Abbas, and Y.-L. He, “Fuzziness based semi-supervised learning approach for intrusion detection system,” *Information Sciences*, vol. 378, no. 1, pp. 484–497, 2017.

[95] Z. Mahmood and C. Agrawal, “Intrusion Detection in Cloud Computing environment using Neural Network,” *International Journal of Research in Computer Engineering and Electronics*, vol. 1, no. 1, pp. 1–4, 2012.

[96] G. Aceto, A. Botta, W. de Donato, and A. Pescapè, “Cloud monitoring: A survey,” *Computer Networks*, vol. 57, no. 9, pp. 2093–2115, Jun. 2013.

[97] A. Bakshi and Y. B. Dujodwala, “Securing Cloud from DDOS Attacks Using Intrusion Detection System in Virtual Machine,” in *Second International Conference on Communication Software and Networks*, 2010, pp. 260–264.

[98] O. Abouabdalla, H. El-Taj, A. Manasrah, and S. Ramadass, “False positive reduction in intrusion detection system: A survey,” in *2nd IEEE International Conference on Broadband Network & Multimedia Technology*, 2009, pp. 463–466.

[99] C. V. Zhou, C. Leckie, and S. Karunasekera, “A survey of coordinated attacks and

collaborative intrusion detection," *Computers & Security*, vol. 29, no. 1, pp. 124–140, Feb. 2010.

[100] M. Auxilia and D. Tamilselvan, "Anomaly detection using negative security model in web application," *2010 International Conference on Computer Information Systems and Industrial Management Applications (CISIM)*, pp. 481–486, Oct. 2010.

[101] S. Neelakantan and S. Rao, "A Threat-Aware Hybrid Intrusion – Detection Architecture for Dynamic Network Environments," *CSI Journal of Computing*, vol. 1, no. 3, 2012.

[102] S. Anwar *et al.*, "From Intrusion Detection to an Intrusion Response System: Fundamentals, Requirements, and Future Directions," *Algorithms*, vol. 10, no. 2, p. 39, 2017.

[103] N. Stakhanova, S. Basu, and J. Wong, "A taxonomy of intrusion response systems," *International Journal of Information and Computer Security*, vol. 1, no. 1/2, pp. 169–184, 2007.

[104] J. Lee, M. Park, and J. Eom, "Multi-level Intrusion Detection System and log management in Cloud Computing," in *13th International Conference on Advanced Communication Technology (ICACT)*, 2011, no. 1, pp. 552–555.

[105] C.-C. Lo, C.-C. Huang, and J. Ku, "A Cooperative Intrusion Detection System Framework for Cloud Computing Networks," in *39th International Conference on Parallel Processing Workshops*, 2010, pp. 280–284.

[106] R. N. Calheiros, A. N. Toosi, C. Vecchiola, and R. Buyya, "A coordinator for scaling elastic applications across multiple clouds," *Future Generation Computer Systems*, vol. 28, no. 8, pp. 1350–1362, 2012.

[107] Z. Chen, F. Han, J. Cao, X. Jiang, and S. Chen, "Cloud computing-based forensic analysis for collaborative network security management system," *Tsinghua Science and Technology*, vol. 18, no. 1, pp. 40–50, 2013.

[108] Y. Wang, H. Yang, X. Wang, and R. Zhang, “Distributed intrusion detection system based on data fusion method,” in *5th World Congress on Intelligent Control and Automation*, 2004, pp. 4331–4334.

[109] N. Asanka, G. Arachchilage, C. Namiluko, and A. Martin, “A taxonomy for securely sharing information among others in a trust domain,” in *8th International Conference for Internet Technology and Secured Transactions (ICITST)*, 2013, pp. 296–304.

[110] J. Montes, A. Sánchez, B. Memishi, M. S. Pérez, and G. Antoniu, “GMonE: A complete approach to cloud monitoring,” *Future Generation Computer Systems*, vol. 29, no. 8, pp. 2026–2040, 2013.

[111] J. Zhang, “Anomaly based network intrusion detection with unsupervised outlier detection,” in *IEEE International Conference on Communications (ICC 2006)*, 2006, pp. 2388–2393.

[112] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, “How to Enhance Cloud Architectures to Enable Cross-Federation,” in *IEEE 3rd International Conference on Cloud Computing*, 2010, pp. 337–345.

[113] C. J. Fung, D. Y. Lam, and R. Boutaba, “A Decision Making Model for Collaborative Malware Detection Networks,” Technical Report CS-2013-01, 2013.

[114] Cloud Security Alliance, “The Notorious Nine Cloud Computing Top Threats in 2013,” 2013, Cloud Security Alliance, 2013.

[115] A. Shaker Ashoor and S. Gore, “Anomaly Detection Algorithm Using Multi-agents,” *International Journal of Scientific & Technology Research*, vol. 1, no. 3, pp. 54–57, 2012.

[116] V. A. Siris and F. Papagalou, “Application of anomaly detection algorithms for detecting SYN flooding attacks,” *Computer Communications*, vol. 29, no. 9, pp. 1433–1442, May 2006.

[117] A. K. A. Agarwal, S. Johri, A. Agarwal, V. Tyagi, “Multi agent approach for network intrusion detection using data mining concepts,” *Journal of Global Research in Computer Science*, vol. 3, no. 3, pp. 29–32, 2012.

[118] C. Zhang, J. Yin, Z. Cai, and W. Chen, “RRED : Robust RED Algorithm to Counter,” *IEEE Communications Letters*, vol. 14, no. 5, pp. 2–4, 2010.

[119] M. Matos, R. Oliveira, E. Deliot, and P. Murray, “CLON : Overlay Networks and Gossip Protocols for Cloud Environments,” *On the Move to Meaningful Internet Systems: OTM 2009. Springer Berlin Heidelberg*, p. 549–566., 2009.

[120] A. G. Fragkiadakis, V. a. Siris, N. E. Petroulakis, and A. P. Traganitis, “Anomaly-based intrusion detection of jamming attacks, local versus collaborative detection,” *Journal of Wireless Communications and Mobile Computing*, vol. 15, no. 2, pp. 276–294, Jan. 2013.

[121] A. Sharma and N. Kumar, “Comparative Analysis of Low Rate Denial of Service Attack in MANETs,” *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 7, pp. 1166–1175, 2013.

[122] A. A. Cárdenas, S. Amin, and Z. Lin, “Attacks Against Process Control Systems : Risk Assessment , Detection , and Response Categories and Subject Descriptors,” *6th ACM Symposium on Information, Computer and Communications Security*, pp. 355–366, 2011.

[123] A. Patcha and J. Park, “An Adaptive Sampling Algorithm with Applications to Denial-of-Service Attack Detection,” *Proceedings of 15th International Conference on Computer Communications and Networks*, pp. 11–16, Oct. 2006.

[124] G. Latif-Shabgahi, J. M. Bass, and S. Bennett, “A Taxonomy for Software Voting Algorithms Used in Safety-Critical Systems,” *IEEE Transactions on Reliability*, vol. 53, no. 3, pp. 319–328, 2004.

[125] D. Hou, H. He, P. Huang, G. Zhang, and H. Loaiciga, “Detection of water-quality

contamination events based on multi-sensor fusion using an extented Dempster–Shafer method," *Measurement Science and Technology*, vol. 24, no. 5, pp. 1–18, May 2013.

[126] Q. Chen and U. Aickelin, "Anomaly Detection Using the Dempster-Shafer Method," in *Proceedings of the International Conference on Data Mining (DMIN 2006)*, 2006, pp. 232–240.

[127] T. M. Chen and V. Venkataramanan, "Dempster-Shafer theory for intrusion detection in ad hoc networks," *IEEE Internet Computing*, vol. 9, no. 6, pp. 35–41, 2005.

[128] Y.T. Liu, N.R. Pal, A. Marathe, and C.T. Lin, "Weighted Fuzzy Dempster-Shafer Framework for Multi-Modal Information Integration," *IEEE Transactions on Fuzzy Systems*, vol. PP, no. 99, pp. 1–16, 2017.

[129] W. H. Jianhua Li and Q. Gao, "Intrusion Detection Engine Based on Dempster-Shafer's Theory of Evidence," in *International Conference on Communications, Circuits and Systems Proceedings*, 2006, vol. 2, no. 2003, pp. 1627–1631.

[130] R. Zuech, T. M. Khoshgoftaar, and R. Wald, "Intrusion detection and Big Heterogeneous Data: a Survey," *Journal of Big Data*, vol. 2, no. 3, pp. 1–41, 2015.

[131] E. Khalil, S. Enniari, and M. Zbakh, "Cloud computing architectures based multi-tenant IDS," in *2013 National Security Days (JNS3)*, 2013, pp. 1–5.

[132] R. Vanathi and S. Gunasekaran, "Comparison of Network Intrusion Detection Systems in Cloud Computing Environment," in *International Conference on Computer Communication and Informatics (ICCCI -2012)*, 2012, pp. 1–6.

[133] L. Coppolino, S. D'Antonio, L. Romano, and G. Spagnuolo, "An Intrusion Detection System for Critical Information Infrastructures using Wireless Sensor Network technologies," in *2010 5th International Conference on Critical Infrastructure (CRIS)*, 2010, pp. 1–8.

[134] S. Taghavi Zargar, H. Takabi, and J. Joshi, "DCDIDP: A Distributed, Collaborative, and

Data-driven Intrusion Detection and Prevention Framework for Cloud Computing Environments," *Proceedings of the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pp. 332–341, 2011.

[135] Á. Macdermott, Q. Shi, M. Merabti, and K. Kifayat, "Security as a Service for a Cloud Federation," in *The 15th Post Graduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting (PGNet2014)*, 2014, pp. 77–82.

[136] Á. Macdermott, Q. Shi, and K. Kifayat, "Distributed attack prevention using Dempster-Shafer theory of evidence," in *ICIC 2017. Lecture Notes in Computer Science, vol 10363. In: Huang DS., Hussain A., Han K., Gromiha M. (eds) Intelligent Computing Methodologies.*, 2017, pp. 203–212.

[137] B. Elmasri, "Detection of Denial Of Service Attacks on Application Layer Protocol," University of Surrey, UK, 2014.

[138] V. Christodoulou and B. Yaxin, "A Combination of CUSUM-EWMA for Anomaly Detection in Time Series Data," in *IEEE Conference on Data Science and Advanced Analytics (DSAA)*, 2015, pp. 1–8.

[139] C. Ding, Y. Chen, T. Xu, and X. Fu, "CloudGPS : A Scalable and ISP-Friendly Server Selection Scheme in Cloud Computing Environments," in *IEEE 20th International Workshop on Quality of Service.*, 2012, pp. 5–13.

[140] G. Hu, W. P. Tay, and Y. Wen, "Cloud Robotics: Architecture, Challenges and Applications," *IEEE Network*, no. May/June, pp. 21–28, 2012.

[141] S. Meng *et al.*, "Reliable State Monitoring in Cloud Datacenters," in *2012 IEEE Fifth International Conference on Cloud Computing*, 2012, pp. 951–958.

[142] M. Mechtri, D. Zeghlache, E. Zekri, and I. J. Marshall, "Inter and intra Cloud Networking Gateway as a service," in *2013 IEEE 2nd International Conference on Cloud Networking (CloudNet)*, 2013, pp. 156–163.

[143] C.-C. Liu, "Challenges and Opportunities of Electric Energy Systems of the Future," Dublin, 2012.

[144] S. Bharadwaja, W. Sun, M. Niamat, and F. Shen, "Collabra: A Xen Hypervisor Based Collaborative Intrusion Detection System," *2011 Eighth International Conference on Information Technology: New Generations*, pp. 695–700, Apr. 2011.

[145] F. F. Alruwaili and T. A. Gulliver, "CCIPS : A Cooperative Intrusion Detection and Prevention Framework for Cloud Services," *International Journal of Latest Trends in Computing*, vol. 4, no. 4, pp. 151–158, 2013.

[146] C. Siaterlis, B. Maglaris, and C. C. N. General-, "Towards Multisensor Data Fusion for DoS detection," in *2004 ACM symposium on Applied Computing*, 2004, pp. 439–446.

[147] A. Josang and S. Pope, "Dempster's Rule as Seen by Little Coloured Balls," *Computational Intelligence*, vol. 28, no. 4, pp. 453–474, 2012.

[148] D. J. Bucci, S. Acharya, T. J. Pleskac, and M. Kam, "Subjective confidence and source reliability in soft data fusion," in *48th Annual Conference on Information Sciences and Systems, (CISS 2014)*, 2014, pp. 1–6.

[149] W. Hurst and Á. MacDermott, "Evaluating the Effects of Cascading Failures in a Network of Critical Infrastructures," *Inderscience International Journal of System of Systems Engineering*, vol. 6, no. 3, pp. 221–236, 2015.

[150] Boundless, "Winning an Election: Majority, Plurality, and Proportional Representation," *Boundless Political Science Boundless*, 2016. [Online]. Available: https://www.boundless.com/political-science/textbooks/boundless-political-science-textbook/campaigns-and-elections-8/elections-60/winning-an-election-majority-plurality-and-proportional-representation-343-10694/. [Accessed: 01-May-2017].

[151] A. Alem, Y. Dahmani, and A. Hadjali, "On the use of Belief Functions to improve High Performance Intrusion Detection System," in *12th International Conference on Signal-Image Technology & Internet-Based Systems*, 2016, pp. 266–270.

[152] V. Nguyen and V. Huynh, “On Information Fusion in Recommender Systems Based-on Dempster-Shafer Theory,” in *IEEE 28th International Conference on Tools with Artificial Intelligence*, 2016, pp. 78–85.

[153] C. Siaterlis and B. Maglaris, “Towards multisensor data fusion for DoS detection,” in *2004 ACM symposium on Applied Computing*, 2004, pp. 439–446.

[154] K. Hwang, S. Kulkareni, and Y. Hu, “Cloud Security with Virtualized Defense and Reputation-Based Trust Mangement,” in *2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing*, 2009, pp. 717–722.

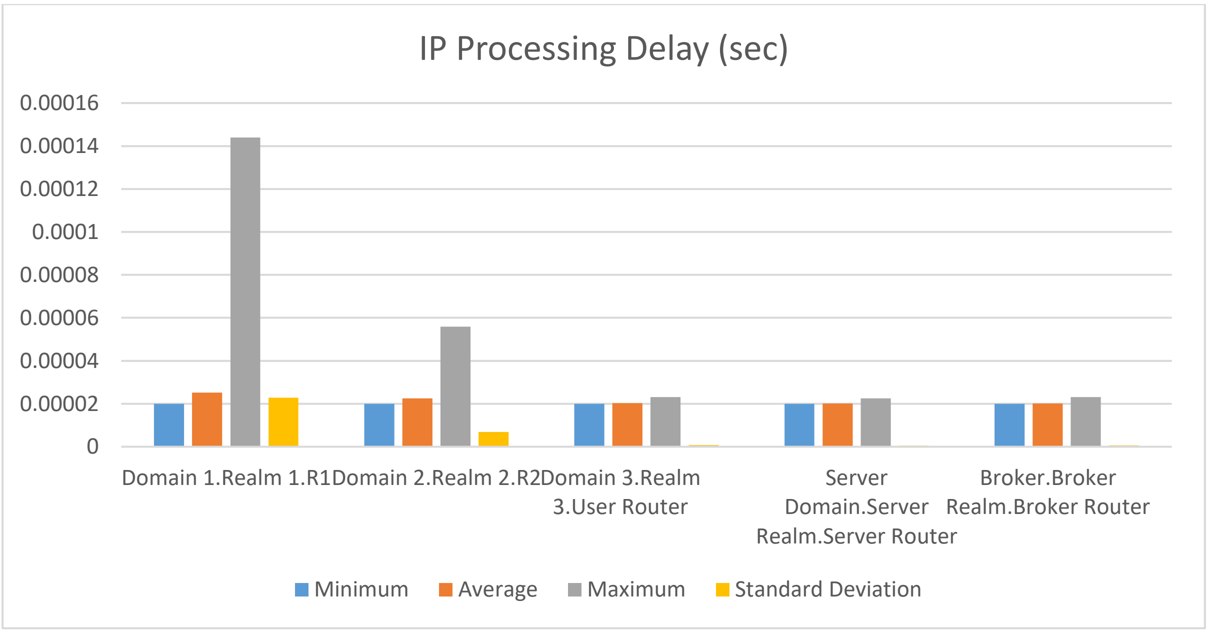
[155] A. Botta, W. Donato, V. Perisco et al., “On the Integration of Cloud Computing and Internet of Things,” in *International Conference on Future Internet of Things and Cloud (FiCloud)*, 2014, pp. 23-30.

Appendix

Further statistics on simulation devices

IP Processing Delay (sec)

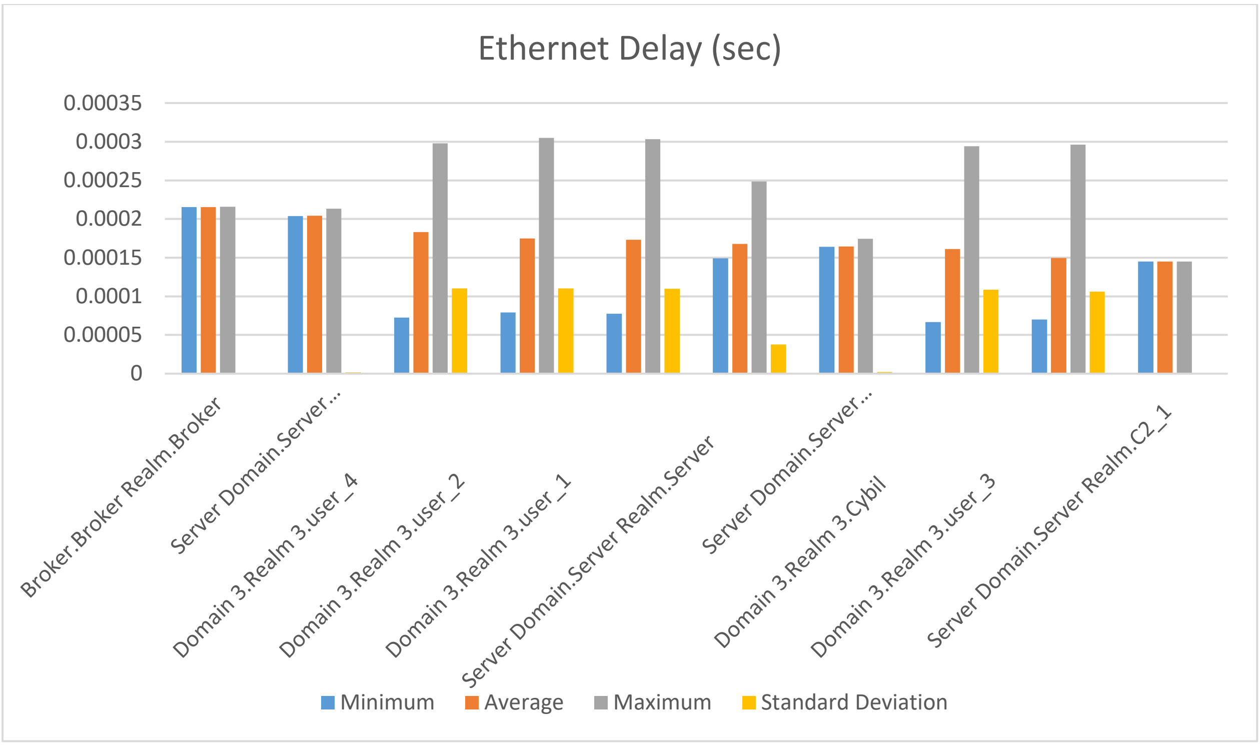
Rank	Object Name	Minimum	Average	Maximum	Standard Deviation
1, 2	Domain 1.Realm 1.R1	0.00002	0.000026	0.00014	0.00002289
3	Domain 2.Realm 2.R2	0.00002	0.000023	0.00005	0.00000690
4	Domain 3.Realm 3.User Router	0.00002	0.00002	0.00002	0.00000077
5	Server Domain.Server Realm.Server Router	0.00002	0.00002	0.00002	0.00000055
6	Broker.Broker Realm.Broker Router	0.00002	0.00002	0.00002	0.00000057



IP processing day (sec)

Ethernet Delay (sec)

Rank	Object Name	Minimum	Average	Maximum	Standard Deviation
1	Broker.Broker Realm.Broker	0.00022	0.00022	0.00022	0.00000001
2	Server Domain.Server Realm.infected_user_4	0.00020	0.00020	0.00021	0.00000173
3	Domain 3.Realm 3.user_4	0.000072	0.00018	0.0003	0.00011022
4	Domain 3.Realm 3.user_2	0.000080	0.00018	0.0003	0.00011021
5	Domain 3.Realm 3.user_1	0.000077	0.00017	0.0003	0.00010979
6	Server Domain.Server Realm.Server	0.00015	0.00017	0.0002	0.00003755
7	Server Domain.Server Realm.infected_user_19	0.00016	0.00016	0.0002	0.00000192
8	Domain 3.Realm 3.Cybil	0.000067	0.00016	0.0003	0.00010845
9	Domain 3.Realm 3.user_3	0.00007	0.00015	0.0003	0.00010604
10	Server Domain.Server Realm.C2_1	0.00015	0.00015	0.0001	0.00000001

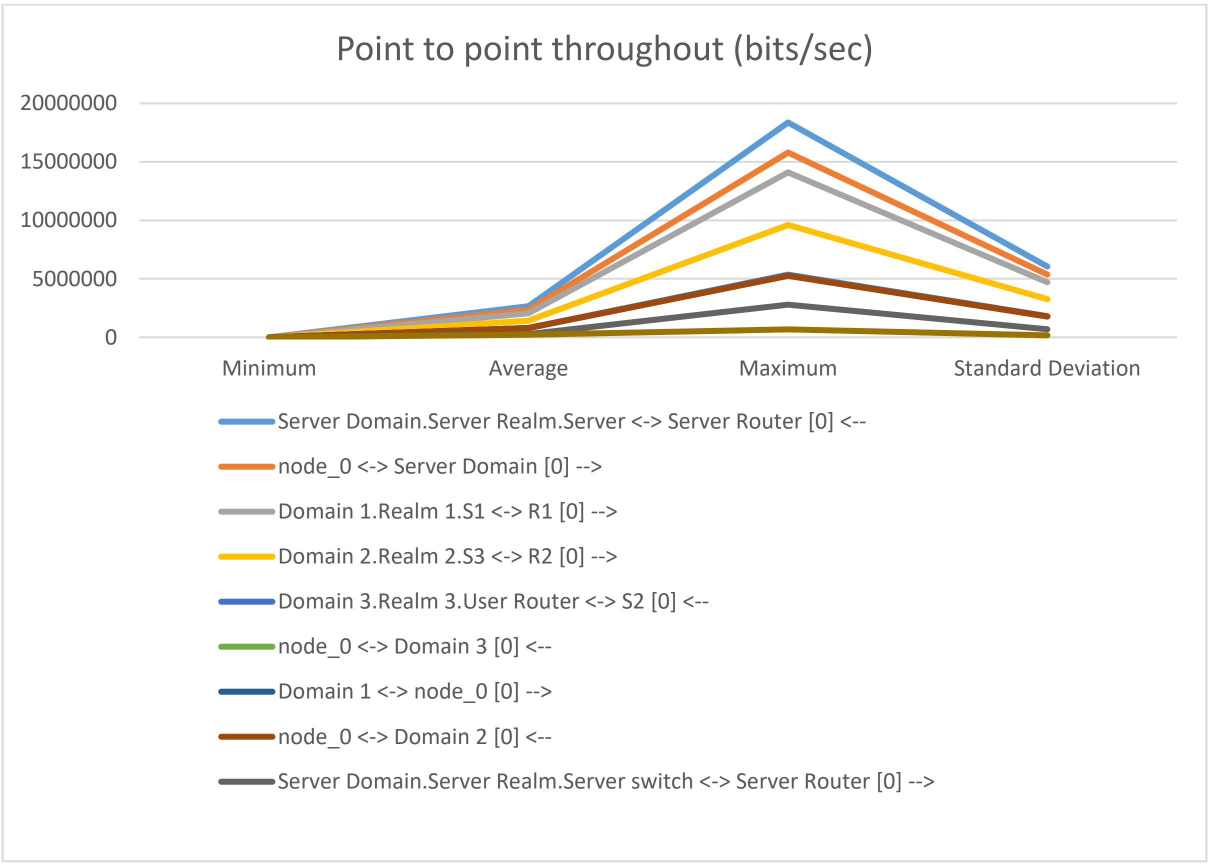


Ethernet delay (sec)

Point to point throughput (bits/sec)

Rank	Object Name	Minimum	Average	Maximum	Standard Deviation
1	Server Domain.Server Realm.Server <-> Server Router [0] <--	60.000	2,661,335	18,366,169	6,059,989
2	node_0 <-> Server Domain [0] -->	66.667	2,355,094	15,793,123	5,356,568
3	Domain 1.Realm 1.S1 <-> R1 [0] -->	0.000	2,051,798	14,099,491	4,698,868
4	Domain 2.Realm 2.S3 <-> R2 [0] -->	0.000	1,429,394	9,604,800	3,260,290
5	Domain 3.Realm 3.User Router <-> S2 [0] <--	0.000	802,289	5,352,178	1,811,553
6	node_0 <-> Domain 3 [0] <--	50.000	789,493	5,273,106	1,785,675
7	Domain 1 <-> node_0 [0] -->	50.000	782,883	5,260,067	1,785,908

8	node_0 <-> Domain 2 [0] <--	33.333	782,863	5,260,067	1,785,471
9	Server Domain.Server Realm.Server switch <-> Server Router [0] -->	0.000	270,774	2,789,839	696,731
10	Server Domain.Server Realm.Server <-> Server Router [0] -->	0.000	217,131	675,362	172,788



Point to point throughout (bits/sec)

Pseudo code for exchange in collaborative intrusion detection schema

Within this section pseudo was written to determine the logic required for implementing the message exchange between the entities in our model. This is a faster way of determining logic prior to creating the final coded model.

```
//IP comes into the network

//if the IP is on black list
if(on black list){
        block user
}
//if the IP is on grey list
else if( on grey list){

        X = AssessActivity(); //determine what is being checked to assess the activity risk
                                                        //return a value which will equate to risk
        //if percentage is over certain threshold - currently set to 70
        if(x = suspiciousActivity){
                //if percentage is at a high threshold
                if(suspiciousActivity = HIGH){
                        if(at HIGH position on grey list){
                                add to black list;
                                contact Broker;
                        }
                        //contact Broker about suspiciousActivity to check against other server
lists
                        call Broker;
                        move to high position on grey list;
                }
                //if percentage is at a mid threshold
                else if(suspiciousActivity = MEDIUM){
                        move to a mid position on grey list;
                }
                //if percentage is at a low threshold
        else if(suspiciousActivity = LOW){
            move to a lower position on grey list;
        }
    }
    if(non suspiciousActivity){
        //lower the IP's position on grey list (as their score is not currently suspicious)
        lower position on grey list;
    }
}

//if the IP is on white list
if(on white list){

    AssessActivity();
    if(non suspiciousActivity){
        //allow IP into network
        Allow access;
    }
    else{
        //do same checks which are in the grey list
        //will be in a function to be easy to call
        grey list checks;
    }
}
//if the IP is not on a list yet
else{
    AssessActivity();
    //if percentage is at a high threshold
    if(suspiciousActivity = HIGH){
            //contact Broker about suspiciousActivity to check against other server
lists
            call Broker;
            add to high position on grey list //while waiting for feedback
    }
    //if percentage is at a mid threshold
```

```
        else if(suspiciousActivity = MEDIUM){
                //contact Broker about suspiciousActivity to check against other server lists
                call Broker;
                add to mid position on grey list; //while waiting for Broker feedback
        }
        //if percentage is at a low threshold
        else if(suspiciousActivity = LOW){
                add to low position on grey list //while waiting for Broker
        }
        else{
                //if the activity is not suspicious, add IP to the white list
                add to white list;
        }
}

///////
//if Broker requests a check of grey list ->receive a check list request

check for IP on grey
//if the ip is on grey list
if(IP exists){
        //return the position on grey list
        //(risk value and any related information on it)
        return risk percentage (position on list)
}
else{
        //if the IP is not on the list
        return no information known
}
```

DS_ProofOfConcept Code

```
Broker.cs

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DS_ProofOfConcept
{
    public class Broker
    {
        List<String[]> GreyList1;
        List<String[]> GreyList2;
        List<String[]> GreyList3;

        double S1A;
        double S1NA;
        double S1E;

        double S2A;
        double S2NA;
        double S2E;

        double S3A;
        double S3NA;
        double S3E;

        public Broker(String IP, double value)
        {
            Console.WriteLine("Retrieving Other Domain Belief Values...");
            //string _filePath = Path.GetDirectoryName(System.AppDomain.Current-
Domain.BaseDirectory);
            //StreamWriter GreyFile = new System.IO.StreamWriter(_filePath + @"\Cur-
rentListData\S" + Server + "GreyList.txt");
            GetOtherServerValues(IP);
            if (Program.Server == 1) { S1A = value /100; }
            if (Program.Server == 2) { S2A = value /100; }
            if (Program.Server == 3) { S3A = value /100; }

            Console.WriteLine("Domain Belief Values Retrieved.");
            Console.WriteLine("Values Returned From Grey List:");
            Console.WriteLine("CSP1 A: " + S1A + " CSP 2A: " + S2A + " CSP3 A: " +
S3A);
            Console.WriteLine("Calculating Outcome...");
            CombinationBelief();
            Console.WriteLine("Calculation Complete.");
            Console.WriteLine();
        }

        void CombinationBelief()
        {
            attackToThreeValue();

            double cooefficient = (S1A * S2NA) + (S1NA * S2A) + (S1A * S3NA) +
                                (S1NA * S3A) + (S2A * S3NA) + (S2NA * S3A);
            double final = (1 / (1 - cooefficient)) * S1E * S2E * S3E;
            Console.WriteLine("CP1 A: " + S1A + " CSP1 NA: " + S1NA + " CSP1 E: " +
S1E);
            Console.WriteLine("CSP2 A: " + S2A + " CSP2 NA: " + S2NA + " CSP2 E: " +
S2E);
            Console.WriteLine("CSP3 A: " + S3A + " CSP3 NA: " + S3NA + " CSP3 E: " +
S3E);
            Console.WriteLine();
            Console.WriteLine("Outcome: " + final);
            Console.ReadLine();
        }

        public void attackToThreeValue()
        {
            if(S1A.ToString() == "0")
            {
                S1A = 0.5;
                S1NA = 0.4;
                S1E = 0.1;
            }
            else if (S1A.ToString() == "1.0")
            {
                S1NA = 0.0;
                S1E = 0.0;

            }
            else if (S1A.ToString() == "0.9")
            {
                S1NA = 0.05;
                S1E = 0.05;
            }

            else if (S1A.ToString() == "0.8")
            {
                S1NA = 0.1;
                S1E = 0.1;
            }

            else if (S1A.ToString() == "0.7")
            {
                S1NA = 0.2;
                S1E = 0.1;
            }

            else if (S1A.ToString() == "0.6")
            {
                S1NA = 0.3;
                S1E = 0.1;
            }
            else if (S1A.ToString() == "0.5")
            {
                S1NA = 0.3;
                S1E = 0.2;
            }
            else if (S1A.ToString() == "0.4")
            {
                S1NA = 0.4;
                S1E = 0.2;
            }
            else if (S1A.ToString() == "0.3")
            {
                S1NA = 0.6;
                S1E = 0.1;
            }
            else if (S1A.ToString() == "0.2")
            {
                S1NA = 0.3;
                S1E = 0.5;
            }
            else if (S1A.ToString() == "0.1")
            {
                S1NA = 0.6;
                S1E = 0.3;
            }
            else if (S1A.ToString() == "0.0")
            {
                S1NA = 0.5;
                S1E = 0.5;
            }
            else
            {
                Console.WriteLine("ERRORRRR ERRORRR");
            }


            if (S2A.ToString() == "0")
            {
                S2A = 0.5;
                S2NA = 0.2;
                S2E = 0.3;
            }
            else if (S2A.ToString() == "1.0")
            {
                S2NA = 0.0;
                S2E = 0.0;

            }
            else if (S2A.ToString() == "0.9")
            {
                S2NA = 0.05;
                S2E = 0.05;
            }

            else if (S2A.ToString() == "0.8")
            {
                S2NA = 0.1;
                S2E = 0.1;
            }

            else if (S2A.ToString() == "0.7")
            {
                S2NA = 0.2;
                S2E = 0.1;
            }

            else if (S2A.ToString() == "0.6")
            {
                S2NA = 0.3;
                S2E = 0.1;
            }
            else if (S2A.ToString() == "0.5")
            {
                S2NA = 0.3;
                S2E = 0.2;
            }
```

```
else if (S2A.ToString() == "0.4")
{
    S2NA = 0.4;
    S2E = 0.2;
}
else if (S2A.ToString() == "0.3")
{
    S2NA = 0.6;
    S2E = 0.1;
}
else if (S2A.ToString() == "0.2")
{
    S2NA = 0.3;
    S2E = 0.5;
}
else if (S2A.ToString() == "0.1")
{
    S2NA = 0.6;
    S2E = 0.3;
}
else if (S2A.ToString() == "0.0")
{
    S2NA = 0.5;
    S2E = 0.5;
}
else
{
    Console.WriteLine("ERRORRR");
}

if (S3A.ToString() == "0")
{
    S3A = 0.5;
    S3NA = 0.3;
    S3E = 0.2;
}
else if (S3A.ToString() == "1.0")
{
    S3NA = 0.0;
    S3E = 0.0;

}
else if (S3A.ToString() == "0.9")
{
    S3NA = 0.05;
    S3E = 0.05;
}

else if (S3A.ToString() == "0.8")
{
    S3NA = 0.1;
    S3E = 0.1;
}

else if (S3A.ToString() == "0.7")
{
    S3NA = 0.2;
    S3E = 0.1;
}

else if (S3A.ToString() == "0.6")
{
        S3NA = 0.3;
        S3E = 0.1;
    }
    else if (S3A.ToString() == "0.5")
    {
        S3NA = 0.3;
        S3E = 0.2;
    }
    else if (S3A.ToString() == "0.4")
    {
        S3NA = 0.4;
        S3E = 0.2;
    }
    else if (S3A.ToString() == "0.3")
    {
        S3NA = 0.6;
        S3E = 0.1;
    }
    else if (S3A.ToString() == "0.2")
    {
        S3NA = 0.3;
        S3E = 0.5;
    }
    else if (S3A.ToString() == "0.1")
    {
        S3NA = 0.6;
        S3E = 0.3;
    }
    else if (S3A.ToString() == "0.0")
    {
        S3NA = 0.5;
        S3E = 0.5;
    }
    else
    {
        Console.WriteLine("ERROR");
    }
}


void GetOtherServerValues(String IP)
{
    if (Program.Server != 3)
    {
        GreyList3 = PopulateCurrentWGBLists.readGreyCurrentInfo(3);
        //print out values of list to console and overwrite file
        for (int i = 0; i < GreyList3.Count; i++)
        {
            if (GreyList3[i][0] == IP)
            {
                S3A = Int32.Parse(GreyList3[i][1]) /100;
            }
        }
    }
    if (Program.Server != 2)
    {
        GreyList2 = PopulateCurrentWGBLists.readGreyCurrentInfo(2);
        //print out values of list to console and overwrite file
        for (int i = 0; i < GreyList2.Count; i++)
        {
            if (GreyList2[i][0] == IP)
            {
```

```
                        S2A = Int32.Parse(GreyList2[i][1]) /100;
                    }
                }
            }
            if (Program.Server != 1)
            {
                GreyList1 = PopulateCurrentWGBLists.readGreyCurrentInfo(1);
                //print out values of list to console and overwrite file
                for (int i = 0; i < GreyList1.Count; i++)
                {
                    if (GreyList1[i][0] == IP)
                    {
                        S1A = Int32.Parse(GreyList1[i][1]) /100;
                    }
                }
            }
        }
    }
}
```

PopulateCurrentWBGLists.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DS_ProofOfConcept
{
    class PopulateCurrentWGBLists
    {


        static String[] WhiteList;
        static List<String[]> GreyList = new List<String[]>();
        static String[] BlackList;



        public static string[] readWhiteCurrentInfo()
        {
            //gets file path position of current program directory
            string _filePath = Path.GetDirectoryName(System.AppDomain.Current-
Domain.BaseDirectory);
            _filePath = Directory.GetParent(Directory.GetPar-
ent(_filePath).FullName).FullName;
            //opens White List for reading from
            string wListRead = System.IO.File.ReadAllText(_filePath + @"\CurrentList-
Data\S1WhiteList.txt");

            //remove all spaces from file
            wListRead.Replace(" ", String.Empty);

            //string[] buffer = wListRead.Split(';');
            //WhiteList = new String[buffer.Length + 5];
            //WhiteList = buffer;

            //split up all IP's into string array
            WhiteList = wListRead.Split(';');

            return WhiteList;
        }

        public static string[] readBlackCurrentInfo()
        {
            //gets file path position of current program directory
            string _filePath = Path.GetDirectoryName(System.AppDomain.Current-
Domain.BaseDirectory);
            _filePath = Directory.GetParent(Directory.GetPar-
ent(_filePath).FullName).FullName;
            //opens Black List for reading
            string bListRead = System.IO.File.ReadAllText(_filePath + @"\CurrentList-
Data\S1BlackList.txt");

            //remove all spaces from file
            bListRead.Replace(" ", String.Empty);

            //split up all IP's into string array
            BlackList = bListRead.Split(';');
            return BlackList;
        }

        public static List<string[]> readGreyCurrentInfo(int Server)
        {
            //gets file path position of current program directory
            string _filePath = Path.GetDirectoryName(System.AppDomain.Current-
Domain.BaseDirectory);
            _filePath = Directory.GetParent(Directory.GetPar-
ent(_filePath).FullName).FullName;
            //opens Grey List for reading from
            string gListRead = System.IO.File.ReadAllText(_filePath + @"\CurrentList-
Data\S" + Server + "GreyList.txt");

            //removed all spaces from file
            gListRead.Replace(" ", null);
            string[] buffer = gListRead.Split(';');

            //for every value in the grey list 'buffer' array
            for (int i = 0; i < buffer.Length - 1; i++)
            {

                populateGreyList(buffer[i], i);

            }

            return GreyList;
        }



        public static void populateGreyList(String item, int i)
        {
            //remove all spaces from current value
            item.Replace(" ", null);
            //create array of the IP and threat level, being split at the ','
            string[] buffer = item.Split(',');
            //add these values to the list array 'GreyList'
            GreyList.Add(buffer);

        }

    }
}
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace DS_ProofOfConcept
{
    public static class Program
    {
        //public variables for all lists
        //Whitelist and Blacklist are arrays, whereas the Greylist is a list which
contain arrays (this will store two values currently).
        static String[] WhiteList;
        static List<String[]> GreyList;
        static String[] BlackList;
        public static int Server = 0;

        //Opens the console
        [DllImport("kernel32.dll", SetLastError = true)]
        [return: MarshalAs(UnmanagedType.Bool)]
        static extern bool AllocConsole();

        [STAThread]
        static void Main()
        {


            //shows console
            AllocConsole();
            while (true)
            {
                Console.WriteLine("Enter Server Number (1,2,3):");
                //Server = Int32.Parse(Console.ReadLine());


                //Populate three Lists in current class
                WhiteList = PopulateCurrentWGBLists.readWhiteCurrentInfo();
                GreyList = PopulateCurrentWGBLists.readGreyCurrentInfo(Server);
                BlackList = PopulateCurrentWGBLists.readBlackCurrentInfo();


                //Output request for user to console
                Console.WriteLine("Enter IP Address");
                //wait for user input
                String inputValue = Console.ReadLine();

                //split up using input at '.' to check if input is correct
                string[] llama = inputValue.Split('.');
                //while the amount of '.' is not equal to 4 (amount needed for accu-
rate IPv4)
                while (llama.Length != 4)
                {
                    //repeat user request
                    Console.WriteLine("Enter IP Address");
                    //wait for user input
                    inputValue = Console.ReadLine();
                    //split user input at '.'
                    llama = inputValue.Split('.');
                }
                //output current input and length for debugging
                Console.WriteLine(inputValue + " " + llama.Length);


                //make new instance for list checking
                ListChecking LC = new ListChecking();
                //check all lists for the ip and place or change position in lists if
needed
                String output = LC.ListCheck(WhiteList, BlackList, GreyList, input-
Value);
                //output the returned string from ListCheck
                Console.WriteLine(output);
                Console.WriteLine();

                //gets file path position of current program directory
                string _filePath = Path.GetDirectoryName(System.AppDomain.Current-
Domain.BaseDirectory);
                _filePath = Directory.GetParent(Directory.GetPar-
ent(_filePath).FullName).FullName;
                //opens White List file for writing
                StreamWriter WhiteFile = new System.IO.StreamWriter(_filePath +
@"\CurrentListData\S1WhiteList.txt");

                //print out values of list to console and overwrite file
                for (int i = 0; i < WhiteList.Length; i++)
                {
                    //Console.Write(" White " + WhiteList[i]);

                    if (i == 0)
                    {
                        WhiteFile.Write(WhiteList[i]);
                    }
                    else
                    {
                        WhiteFile.Write(";" + WhiteList[i]);
                    }
                }
                //close the white file
                WhiteFile.Close();
                //Console.WriteLine();
                //opens Grey List file for writing to
                StreamWriter GreyFile = new System.IO.StreamWriter(_filePath + @"\Cur-
rentListData\S" + Server + "GreyList.txt");

                //print out values of list to console and overwrite file
                for (int i = 0; i < GreyList.Count; i++)
                {
                    //Console.Write(" Grey " + GreyList[i][0]);
                    GreyFile.Write(GreyList[i][0] + "," + GreyList[i][1] + ";");
                }
                //close the grey file
                GreyFile.Close();
                //Console.WriteLine();
                //opens Black List file for writing to
                StreamWriter BlackFile = new System.IO.StreamWriter(_filePath +
@"\CurrentListData\S1BlackList.txt");

                //print out values of list to console and overwrite file
                for (int i = 0; i < BlackList.Length; i++)
                {
                    //Console.Write(" Black " + BlackList[i]);

                    if (i == 0)
                    {
                        BlackFile.Write(BlackList[i]);
                    }
                    else
                    {
                        BlackFile.Write(";" + BlackList[i]);
                    }
                }
                //close the black file
                BlackFile.Close();

                //read line holds the console in place until a key is pressed
                Console.ReadLine();
            }

        }



        //setting values of white list in this class, used when list is changed
        public static void setWhiteList(string[] WL)
        {
            WhiteList = WL;

        }
        //setting values of black list in this class, used when list is changed
        public static void setBlackList(string[] BL)
        {
            BlackList = BL;

        }
        /*public static void setGreyList(List<string[]> GL)
        {
            GreyList = GL;

        }*/
    }
}
```

ListChecking.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DS_ProofOfConcept
{
    class ListChecking
    {

        //values decide what returned percentage equals which tier
        public int HIGHTHREAT = 70;
        public int MIDTHREAT = 40;
        public int LOWTHREAT = 20;

        //amount to increase threat level on grey list if threat detected
        //and in which tier
        public int highTeirIncreaseAmount = 50;
        public int midTeirIncreaseAmount = 20;
        public int lowTeirIncreaseAmount = 10;
        public int belowLowTierReduction = 5;


        //checking white and black lists for IP (done seperatly, use same method)
        public KeyValuePair<Boolean, int> checkthisList(string[] currentList, string
IP)
        {
            for(int i = 0; i< currentList.Length; i++)
            {
                if(currentList[i] == IP || currentList[i].Contains(IP))
                {
                    return new KeyValuePair<Boolean, int>(true, i);
                }
            }
            return new KeyValuePair<Boolean, int>(false, 0);
        }

        //checking grey list for IP
        public KeyValuePair<Boolean,int> checkGreyList(List<string[]> greyList, string
IP)
        {

            for (int i = 0; i < greyList.Count; i++)
            {
                if (greyList[i][0] == IP || greyList[i][0].Contains(IP))
                {
                    return new KeyValuePair<Boolean, int>(true, i);
                }
            }
            return new KeyValuePair<Boolean, int>(false, 0);
        }


        //checking lists trying to find if IP is on list and what threat it holds
        public string ListCheck(string[] WhiteList, string[] BlackList, List<string[]>
GreyList, string IP)
        {
            //scan all lists
            KeyValuePair<Boolean, int> BlackValue = checkthisList(BlackList, IP);
            KeyValuePair<Boolean, int> GreyValue = checkGreyList(GreyList, IP);
            KeyValuePair<Boolean, int> WhiteValue = checkthisList(WhiteList, IP);
            Console.WriteLine("Current List:");
            //if the IP is in the black list (Key will be true)
            if (BlackValue.Key)
            {
                Console.WriteLine("***BLACK***");
                //block IP
                return "Blocked User";

            }
            //if IP is in the grey list (Key will be true)
            else if(GreyValue.Key)
            {
                Console.WriteLine("***GREY***");
                //check for amount of threat detected
                int ThreatLevel = CheckThreat();
                //if the threat is within the high tier bracket for this attack
                if (ThreatLevel >= HIGHTHREAT)
                {
                    //if the threat of IP is already above 70
                    if(Int32.Parse(GreyList[GreyValue.Value][1]) > 70)
                    {
                        //make a new array of the new IP being added
                        String[] array2 = { IP };
                        //make new array the size of the Black List and new array list 
to be the correct length
                        String[] newArray = new String[BlackList.Length + ar-
ray2.Length];
                        //combine black list and new array
                        Array.Copy(BlackList, newArray, BlackList.Length);
                        //combine old Black List and new values into the new array, 
soon to become the black list
                        Array.Copy(BlackList, 0, newArray, BlackList.Length, ar-
ray2.Length);

                        //use method within the Program class to update value of Black 
List to then be written
                        //back to the black list file
                        Program.setBlackList(newArray);



                        //remove add to Black List
                        Console.WriteLine("WARNING " + IP + " Added To Black List");
                        //remove from grey list
                        GreyList.RemoveAt(GreyValue.Value);
                        //contact broker HERE
                        return "Blocking User";
                    }
                    else
                    {
                        //otherwise change to a high threat status on grey list
                        GreyList[GreyValue.Value][1] = Int32.Parse(GreyList[Grey-
Value.Value][1]) + highTeirIncreaseAmount + ""; ;
                        Console.WriteLine(IP + " Threat Level increased to " + 
GreyList[GreyValue.Value][1] + " on Grey List");
                        return "User High Threat";
                    }
                }
                //if the threat is within the Mid tier bracket for this attack
                else if (ThreatLevel >= MIDTHREAT)
                {
                   //increase their threat status by the mid teir amount
                   GreyList[GreyValue.Value][1] = Int32.Parse(GreyList[Grey-
Value.Value][1]) + midTeirIncreaseAmount + "";
                   Console.WriteLine(IP + " Threat Level increased to " + 
GreyList[GreyValue.Value][1] + " on Grey List");
                    return "User Mid Threat";
                }
                //if the threat is within the Mid-tier bracket for this attack
                else if (ThreatLevel >= LOWTHREAT)
                {
                    //increase their threat status by the mid teir amount
                    GreyList[GreyValue.Value][1] = Int32.Parse(GreyList[Grey-
Value.Value][1]) + lowTeirIncreaseAmount + "";
                    Console.WriteLine(IP + " Threat Level increased to " + 
GreyList[GreyValue.Value][1] + " on Grey List");
                    return "User Low Threat";
                }
                //if their threat is low
                else
```

```
                {
                    //lower their threat status
                    GreyList[GreyValue.Value][1] = Int32.Parse(GreyList[Grey-
Value.Value][1]) - belowLowTierReduction + "";
                    Console.WriteLine(IP + " Threat Level decreased to " +
GreyList[GreyValue.Value][1] + " on Grey List");
                    //if their threat  status is below zero, they haven't caused is-
sues for a long time
                    //remove them from grey list, add to white list
                    if (Int32.Parse(GreyList[GreyValue.Value][1]) == 0)
                    {
                        //make a new array of the new IP being added
                        String[] array2 = { IP };
                        //make new array the size of the White List and new array list
to be the correct length
                        String[] newArray = new String[WhiteList.Length + ar-
ray2.Length];
                        //combine White List and new array
                        Array.Copy(WhiteList, newArray, WhiteList.Length);
                        //combine old White List and new values into the new array,
soon to become the White list
                        Array.Copy(array2, 0, newArray, WhiteList.Length, ar-
ray2.Length);

                        //use method within the Program class to update value of White
List to then be written
                        //back to the White List file
                        Program.setWhiteList(newArray);

                        //add to white list
                        //WhiteList[WhiteList.Length-1] = GreyList[Grey-
Value.Value][0].ToString();
                        //remove from grey list
                        GreyList.RemoveAt(GreyValue.Value);
                        Console.WriteLine(IP + " Removed from Grey List");
                        Console.WriteLine(IP + " Added to White List");
                        return "No threat, Removed from Grey List, added to White
List";
                    }
                    return "No threat, User low level on Grey List.";
                }
            }
            //if IP is in the white list (Key will be true)
            else if (WhiteValue.Key)
            {
                Console.WriteLine("***WHITE***");
                //check threat level of current attack
                int ThreatLevel = CheckThreat();

                //if thread level is below the low threshold
                if (ThreatLevel <= LOWTHREAT)
                {
                    //Allow Access
                    return "Allow access, user on White List";
                }
                //otherwise there is an issue
                else
                {
                    //if thread is over the High level tier
                    if (ThreatLevel >= HIGHTHREAT)
                    {
                        //contact Broker
                        //create buffer for adding IP to grey list with a high threat
list
                        String[] buffer = { IP, ThreatLevel.ToString() };
                        //add IP to grey list using buffer
                        GreyList.Add(buffer);
                        //remove IP from white list
                        WhiteList.Except(new string[] {White-
List[WhiteValue.Value]}).ToArray();

                        new Broker(IP, ThreatLevel);
                        return "User added to Grey list for High risk";
                    }
                    //if threat is over the mid level tier
                    else if (ThreatLevel >= MIDTHREAT)
                    {
                        //create buffer for adding IP to grey list with a high threat
list
                        string[] buffer = { IP, midTeirIncreaseAmount.ToString()};
                        //add IP to grey list using buffer
                        GreyList.Add(buffer);
                        //remove IP from white list
                        WhiteList.Except(new string[] { WhiteList[WhiteValue.Value]
}).ToArray();
                        return "User added to Grey list for Mid risk";
                    }
                    //if threat is over the low level tier
                    else if (ThreatLevel >= LOWTHREAT)
                    {
                        //create buffer for adding IP to grey list with a high threat
list
                        string[] buffer = { IP, lowTeirIncreaseAmount.ToString() };
                        //add IP to grey list using buffer
                        GreyList.Add(buffer);
                        //remove IP from white list
                        WhiteList.Except(new string[] { WhiteList[WhiteValue.Value]
}).ToArray();
                        return "User added to Grey list for Low risk";
                    }

                }
            }
            else
            {
                Console.WriteLine("***No List***");
                //check threat level of current attack
                int ThreatLevel = CheckThreat();

                //if threat level is below the low threshold
                if (ThreatLevel <= LOWTHREAT)
                {
                    //make a new array of the new IP being added
                    String[] array2 = { IP };
                    //make new array the size of the White List and new array list to
be the correct length
                    String[] newArray = new String[WhiteList.Length + array2.Length];
                    //combine Whit list and new array
                    Array.Copy(WhiteList, newArray, WhiteList.Length);
                    //combine old White List and new values into the new array, soon
to become the White List
                    Array.Copy(array2, 0, newArray, WhiteList.Length, array2.Length);
                    //use method within the Program class to update value of Black
List to then be written
                    //back to the black list file
                    Program.setWhiteList(newArray);

                    //Allow Access
                    return "Allow access, user on White List";
                }
                //otherwise there is an issue
                else
                {
                    //if threat is over the High level tier
                    if (ThreatLevel >= HIGHTHREAT)
                    {
                        //contact Broker

                        //create buffer for adding IP to grey list with a high threat
list
                        String[] buffer = { IP, ThreatLevel.ToString() };
                        //add IP to grey list using buffer
                        GreyList.Add(buffer);
                        new Broker(IP, ThreatLevel);

                        return "User added to Grey list for High risk";
                    }
                    //if threat is over the mid level tier
                    else if (ThreatLevel >= MIDTHREAT)
                    {
                        //create buffer for adding IP to grey list with a high threat
list
                        string[] buffer = { IP, midTeirIncreaseAmount.ToString() };
                        //add IP to grey list using buffer
                        GreyList.Add(buffer);

                        return "User added to Grey list for Mid risk";
                    }
                    //if threat is over the low level tier
                    else if (ThreatLevel >= LOWTHREAT)
                    {
                        //create buffer for adding IP to grey list with a high threat
list
                        string[] buffer = { IP, lowTeirIncreaseAmount.ToString() };
                        //add IP to grey list using buffer
                        GreyList.Add(buffer);

                        return "User added to Grey list for Low risk";
                    }
                }
            }

            return "Error";
        }


        public int CheckThreat()
        {
            Console.WriteLine();
            Console.WriteLine();
            Console.WriteLine();

            //do calculation!!
```

```
Random rand = new Random();
double random = rand.NextDouble();
double NoAttack = 0;
double Either = 0;

string Attack = string.Format("{0:0.0}", Math.Truncate(random * 10) / 10);
Console.WriteLine("Threat Score    " + double.Parse(Attack) * 100);

if(Attack == "1.0")
{
    NoAttack = 0.0;
    Either = 0.0;

}
else if(Attack == "0.9")
{
    NoAttack = 0.05;
    Either = 0.05;
}

else if (Attack == "0.8")
{
    NoAttack = 0.1;
    Either = 0.1;
}

else if (Attack == "0.7")
{
    NoAttack = 0.2;
    Either = 0.1;
}

else if (Attack == "0.6")
{
    NoAttack = 0.3;
    Either = 0.1;
}
else if (Attack == "0.5")
{
    NoAttack = 0.3;
    Either = 0.2;
}
else if (Attack == "0.4")
{
    NoAttack = 0.4;
    Either = 0.2;
}
else if (Attack == "0.3")
{
    NoAttack = 0.6;
    Either = 0.1;
}
else if (Attack == "0.2")
{
    NoAttack = 0.3;
    Either = 0.5;
}
else if (Attack == "0.1")
{
    NoAttack = 0.6;
    Either = 0.3;
}
```

```
            else if (Attack == "0.0")
            {
                NoAttack = 0.5;
                Either = 0.5;
            }
            else
            {
                Console.WriteLine("ERROR");
            }


            double attackPlausability = 1 - NoAttack;
            double NoAttackPlausability = 1 - double.Parse(Attack);


            //Console.WriteLine("Hmmm!!!!   " + Attack);
            String NoAttackPlausabilityFormatConversion;
            if(NoAttackPlausability == 1)
            {
                NoAttackPlausabilityFormatConversion = "1.0";
            }
            else
            {
                NoAttackPlausabilityFormatConversion = NoAttackPlausabil-
ity.ToString();
            }

            Console.WriteLine("-------------------------------------------------------
------------------");
            Console.WriteLine("---Hypothesis-||--Mass--||---Belief--||-Plausibil-
ity||");
            Console.WriteLine("-----Attack --||-- " + Attack + "  ||   " + Attack + "
||     " + attackPlausability + "     ||");
            Console.WriteLine("---No Attack -||-- " + NoAttack + "  ||   " + NoAttack
+ "     ||     " + NoAttackPlausabilityFormatConversion + "     ||");
            Console.WriteLine("-----Either --||-- " + Either + "  ||   " + "1.0" + "
||     " + "1.0" + "     ||");
            Console.WriteLine("-------------------------------------------------------
------------------");



            Console.WriteLine();
            Console.WriteLine();
            Console.WriteLine();

            return Convert.ToInt32(Double.Parse(Attack)*100);
        }
```