INTRUSION PREDICTION SYSTEM FOR CLOUD COMPUTING AND NETWORK BASED SYSTEMS

MOHAMED ABDLRAHEM ABDLHAMED BSc, MSc

A thesis submitted in partial fulfilment of the requirements of

Liverpool John Moores University for the degree of doctor of philosophy

May 2018

This thesis is dedicated to my beloved uncle

Prof. Abdul-Kareem Abdul-Hameed,

who was sadly unable to witness its completion.

Acknowledgements

It is such an honour to thank my Mother and Father, their unconditional endless love and support made me the man I am.

I offer my blessings and regards to my beloved wife who supported me endlessly and took over all my responsibilities to enable me accomplish this work. I hope that one day I can reward you!

My sincere thanks to my director of study Dr. Kashif Kifayat, whose supervision, engagement, and encouragement in every step throughout this work and help and guidance have kept me concentrated and motivated to complete this work successfully.

I would like also to thank Professor Qi Shi and Dr. William Hurst for their support and help.

I would like to thank Prof. Dhiya Al-Jumeily for his appreciated great help and support.

At the finishing of this work, I would like to acknowledge and thank the Department of Computing at LJMU and the wonderful staff. With particular thanks going to Tricia Waterson, Ian Fitzpatrick, Neil Rowe, Steven Thompson, Paul Cartwright.

Lastly, all the thanks and regards to the Ministry of Higher Education and Scholarships department in my beloved country IRAQ and the Iraqi cultural attaché in London who funded this study entirely. Continued thanks to the University of Anbar and College of Imam Al - Adham Abu Hanifa Al - Nu'man, who allowed me this opportunity.

Abstract

Cloud computing offers cost effective computational and storage services with on-demand scalable capacities according to the customers' needs. These properties encourage organisations and individuals to migrate from classical computing to cloud computing from different disciplines.

Although cloud computing is a trendy technology that opens the horizons for many businesses, it is a new paradigm that exploits already existing computing technologies in new framework rather than being a novel technology. This means that cloud computing inherited classical computing problems that are still challenging. Cloud computing security is considered one of the major problems, which require strong security systems to protect the system, and the valuable data stored and processed in it. Intrusion detection systems are one of the important security components and defence layer that detect cyber-attacks and malicious activities in cloud and non-cloud environments. However, there are some limitations such as attacks were detected at the time that the damage of the attack was already done. In recent years, cyber-attacks have increased rapidly in volume and diversity. In 2013, for example, over 552 million customers' identities and crucial information were revealed through data breaches worldwide [3]. These growing threats are further demonstrated in the 50,000 daily attacks on the London Stock Exchange [4]. It has been predicted that the economic impact of cyber-attacks will cost the global economy $3 trillion on aggregate by 2020 [5].

This thesis focused on proposing an Intrusion Prediction System that is capable of sensing an attack before it happens in cloud or non-cloud environments. The proposed solution is based on assessing the host system vulnerabilities and monitoring the network traffic for attacks preparations. It has three main modules. The monitoring module observes the network for any intrusion preparations. This thesis proposes a new dynamic-selective statistical algorithm for detecting scan activities, which is part of reconnaissance that represents an essential step in network attack preparation. The proposed method performs a statistical selective analysis for network traffic searching for an attack or intrusion indications. This is achieved by exploring and applying different statistical and probabilistic methods that deal with scan detection. The second module of the prediction system is vulnerabilities assessment that evaluates the weaknesses and faults of the system and measures the probability of the system to fall victim to cyber-attack. Finally, the third module is the prediction module that combines the output of the two modules and performs risk assessments of the system security from intrusions prediction. The results of the conducted experiments showed that the suggested system outperforms the analogous methods in regards to performance of network scan detection, which means accordingly a significant improvement to the security of the targeted system. The scanning detection algorithm has achieved high detection accuracy with 0% false negative and 50% false positive. In term of performance, the detection algorithm consumed only 23% of the data needed for analysis compared to the best performed rival detection method.

Contents

List of Figures

List of Tables

List of Abbreviations

AC	Attack Complexity
ANN	Artificial Neural Network
API	Application Programming Interface
APT	Advanced Persistent Threats
AV	Attack Vector
C&C , C2	Command and Control
CAMP	Cyber Attacker Model Profile
CGI	Common Gateway Interfaces
CIA	Confidentiality, Integrity, And Availability
CIDS	Cloud Intrusion Detection Service
CNSS	Committee on National Security Systems
CPT	Conditional Probability Table
CPU	Central Processing Unit
CSRF	Cross-Site Request Forgery
CVE	Common Vulnerabilities and Exposures
DAC	Discretionary Access Control
DAG	Directed Acyclic Graph
DARPA	Defence Advanced Research Projects Agency
DCOM	Distributed Component Object Model
DDDAS	Dynamic Data-Driven Application Systems
DDoS	Distributed Denial of Service
DIDS	Distribution Intrusion Detection System
DNS	Domain Name System
DoS	Denial of Service
EC2	Elastic Compute Cloud
ESD	Efficient Scan Detection
EWMA	Exponential Weighted Moving Average
FMEA	Failure Mode & Effects Analysis
FNR	False Negatives Ratio
FPR	False Positives Ratio
FTA	Fault Tree Analysis

GB	Gigabyte
HIDS	Host based intrusion detection system
HMM	Hidden Markov Model
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IAAS	Infrastructure As A Service
IAM	Identity and Access Management
ICMP	Internet Control Message Protocol
IDPS	Intrusion Detection and Prevention System
IDS	Intrusion Detection Systems
IP	Internet Protocol
IPFCC	Intrusion Prediction Framework for Cloud Computing
IPS	Intrusion Prediction System
IPS	Internet Service Providers
IT	Information Technology
JSON	JavaScript Object Notation
KDD	Knowledge Discovery and Data Mining
MAC	Mandatory Access Control
MAE	Cyber Mission Assurance Engineering
MFA	Multi-factor Authentication
MIT	Massachusetts Institute of Technology
NIDS	Network Based Intrusion Detection System
OS	Operating System
PaaS	Platform as a Service
PGA	Partheno Genetic Algorithms
PPP	Program Protection Plan
R2L	Remote-to-Local
RBAC	Role Based Access Control
RPC	Remote Procedure Call
SAAS	Software As A Service
SLA	Service Level Agreement
SN	Sequence Number
SOA	Service-Oriented Architecture

SQL	Structured Query Language
SQUARE	Security Quality Requirements Engineering
SRAM	Static Random-Access Memory
SSE	System Security Engineering
SSL	Secure Sockets Layer
SSO	Single sign-on
SVM	Support Vector Machine
SYN	Synchronize
SYN-ACK	Synchronize Acknowledge
TAPS	Time Access Pattern Scheme
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TNS	Trusted Systems and Networks Analysis
TPM	Trusted Platform Module
TRW	Threshold Random Walk
TTL	Time To Live
TV	Trust Value
U2R	User-to-Root
UDP	User Datagram Protocol
URL	Uniform Resource Locator
UTC	Coordinated Universal Time
VAM	Vulnerability Assessment and Mitigation
VM	Virtual Machine
VMM	Virtual Machine Monitor
XML	Extensible Markup Language
XSS	Cross-site Scripting

List of Publications

A list of the publication consistent in this thesis, some are already published and others still in publication process till the submission of this thesis.

1. M. Abdlhamed, K. Kifayat, Q. Shi, and W. Hurst, “Intrusion Prediction Systems,” in Information Fusion for Cyber-Security Analytics, Springer International Publishing, 2017, pp. 155–174.
2. M. Abdlhamed, K. Kifayat, Q. Shi, and W. Hurst, “A System for Intrusion Prediction in Cloud Computing,” in proceedings of The International Conference on Internet of things and Cloud Computing (ICC 2016), University of Cambridge, Uk, 2016, pp. 1–9.
3. M. Abdlhamed, K. Kifayat, “Security Challenges in Cloud Computing Domains”, Proceedings of the 15th Annual Post Graduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting (PGNet 2014), Liverpool : Liverpool John Moores University.
4. M. Abdlhamed, K. Kifayat, Q. Shi, and W. Hurst, “A Selective Dynamic Port Scan Detection Algorithm”, Accepted at International Conference on Developments in eSystems Engineering (DeSE), 2nd – 5th September 2018, Cambridge, England, UK. (Under publication).

Chapter 1

Introduction

1.1 Cloud Computing

Recent era of human existence is characterised by technology revolution. Computer invention and advances in electronics field have led the transformation of computerise of many daily human jobs and services. This increase reliance on computers urges to more advances and developments in computer field that results in hardware and software that are more sophisticated. This advances resulted in the appearance of new technologies such as internet technologies, large volume of data storage and processing, prevalence of virtualization. The existence of such advanced technologies and the need for using resources remotely are factors contributed in the emergence of cloud computing. Cloud computing is a new way of delivering services and provision resources over the Internet [1]. It brought many advantages; economically by reducing expenditure on hardware and software for the companies and personals, the flexibility of developing and deploying applications rapidly regardless the hosted hardware and software, and the ability to scale resources up and down according to customers need [2]. On the other hand, cyber-based systems inherited the security problems of the classical computing and the Internet. Figure 1.1 shows basic architecture of cloud as following:

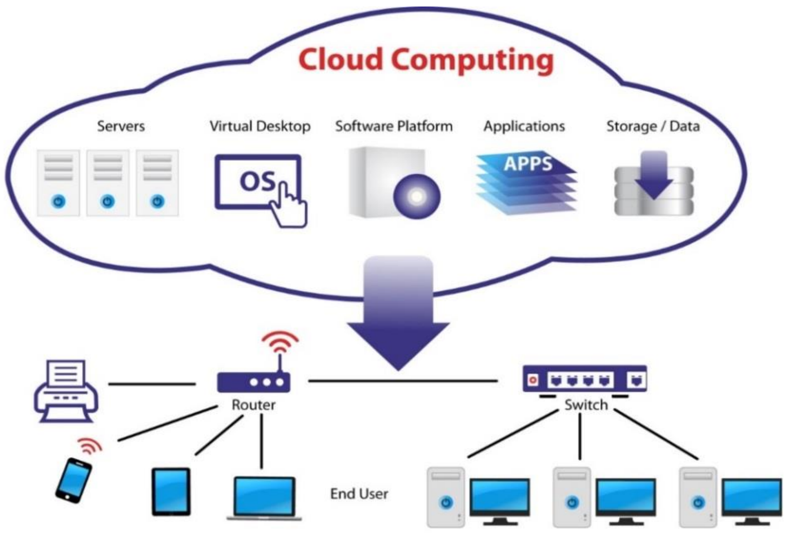


Figure 1.1. Cloud Computing Architecture [3]

Figure 1.1 shows different services offered by cloud computing such as Infrastructure As A Service (IAAS) (Servers, Storage), Software As A Service (SAAS) (Software Platform, Virtual Desktop, Applications). These services are accessible by the end users via the Internet, where the end user uses different terminal devices such as personal computers, smart phones, and tablets to access the cloud.

1.1.1 Security Challenges and Attacks in Cloud Computing

Many challenges and attacks have faced the cloud computing due to the broadness of both vulnerabilities and attack vectors. Multiple types of challenges to this environment are posed from different technologies and levels of software and hardware. In this section presents an introduction to these attacks and challenges to give the reader a comprehensive view of the state of art of cloud security, and will present a detailed description in the next two chapters.

There are three types of cloud deployment; private that is deployed with an organization for internal use, public that provides different service for various originations, and hybrid that combines the two previous types. These forms of deployment are accompanied by security threats such as data leakage due to cloning and resource pooling [4], and unauthorised access to data residuals that are obsolete in different server of the public cloud [5]. Private clouds produces an ‘elastic perimeter’ concept due to alternate between centralizing and distributing the data according to user need, which results in data loss when storing data in higher level of authorization zones [6]. The multi-tenant aspect of the cloud poses another security threat when an illegal access to data is been tried by an authorised costumer that share the same hardware with the victim [7].

As mentioned previously, cloud computing offers a number of services such as Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) that, in turns, poses different challenges to cloud security according to the types of offered the services. While the types of security threats in cloud deployment models are dependent more on exploiting managerial faults or bugs, the threats at service level are mostly using the technical vulnerabilities to achieve the malicious purposes. This is more obvious in the security threats that are caused by the virtualization concept of the cloud such as Virtual Machine (VM) Hopping, Mobility, and Denial of Service and Service Hijacking. In Virtual Machine Hopping [8] , an attacker could gain access and control another VM on the same server, which impact the confidentiality and the availability of the victim severely. In VM Mobility, the VM and its contents are saved as portable soft copies so it could easily relocated from server to another via portable devices without having a copy on fixed storage [4], this process can lead to serious security threats such as data leakage and loss, and targeted attacks like man in-the-middle attack. VM Denial of Service (DoS) happens when a VM consumes all the available resources that server cannot operate the other VMs resides in the server [9]. Service hijacking is also threat to cloud that occur when an unauthorized users gaining illegal control on certain services [10]. Backups of data may leads to security threats when misconducted that results in data leakage or misused by an unauthorised parties [11] .

Network-based attacks are another type of security threats, whether these attacks targeting the network infrastructure or software. Browsing attacks are such attacks where the browsing software is used to launch the attacks such as sniffing, SQL injection, and XML signature wrapping attacks. In the sniffing attack, the attacker install malware on an intermediary host to steal the victim credentials, which leads to illegal use of legit credentials. SQL injection attacks work by inserting malicious code in a model SQL inputs that leads to granting the attacker an unauthorised access to the database and consequently to other confidential information [12]. The XML Signature Element Wrapping attacks targeting to disturb the cloud service by inserting malicious data in signature part of the massage, this may lead the cloud interface to execute arbitrary methods [12]. Different types of attacks might target the network infrastructure such as flooding attacks, which target the bandwidth of the network in order to affect the service availability [13]. In addition to many network based attacks that targets the conventional networking systems, and applied to the cloud for malicious purposes such as the DoS attack.

To demonstrate the impact of cyber-attacks on cloud computing environment, this section present a recent snapshot of these attacks and their implications on both global economy and people. In recent years, cyber-attacks have increased rapidly in volume and diversity. In 2013, for example, over 552 million customers' identities and crucial information were revealed through data breaches worldwide [3]. These growing threats are further demonstrated in the 50,000 daily attacks on the London Stock Exchange [4]. It has been predicted that the economic impact of cyber-attacks will cost the global economy $3 Trillion on aggregate by 2020 [5]. These immense effects and implications have urged the United States Department of Defence to categorize cyber-attacks as an act of war, meriting physical military force in response [4]. Such a categorization depicts the severe view countries across the globe now have of cyber-threats. The classical cyber-attacks such as the Distributed Denial of Service (DDoS) are continuing to target the cloud such that four in five organizations has been targeted by DDoS attack during 2017 [14]. Furthermore, Massachusetts Institute of Technology (MIT) predicts sophisticated ransomware attacks will target the cloud in 2018 [15]. Based on such cyber security atmosphere, the cloud security market will rise from $1.4 Billion to $12 Billion by 2024 [16]. The consequences of cyber-attacks such as huge economic expenses, business effects, and personal damages encourage security researchers to focus on this area in order to mitigate these consequences.

1.1.2 Intrusion Detection and Prevention Systems

The varieties of systems used to counter intrusions are operated in order to detect, prevent and mitigate attacks and their consequences. Among the existing traditional defence systems, Intrusion Detection Systems (IDSs) are playing a major role. Typically, they are composed of software or hardware that automatically monitors computer systems or networks and performs an analysis function to detect intrusions [2]. IDSs are generally grouped into a number of classes: host-based intrusion-detection systems such as Cisco IDS Host Sensor [3] ; Network-based IDS such as Snort [17] Security Onion [18] and distributed intrusion-detection systems such as myNetWatchman [19] and DShield [20].

Prior to commencing any attack, adversaries collect information about the targets such as identifying and mapping the target network, checking the potential victims for vulnerabilities, and other network scanning activities. The initial stage of any attack is planning and reconnaissance, when it comes to attacks that use the network as an attack medium, the reconnaissance is achieved by network scanning activities. This gives the detection of network scanning activities the importance, as it enables the security administration to be aware of the attacks at early stages. Based on the gathered information, the attacker designs the attack plan and chooses suitable attacks according to the discovered weaknesses in the targeted machine. Therefore, identifying and detecting such hostile network scanning activities will support blocking of further harmful attacks.

Network scanning activities are usually performed to discover and specify the online machines that are hosted in the network, and correlate the offered services on these hosts to open ports. Different types of solutions have been proposed to detect scanning activity including monitoring failed connections [21][22], statistical calculations [23][24], network behaviour abnormalities [25][26] and monitor connections to network unused IPs [27]. The purpose of these approaches is to identify the remote host that performs scanning activity to a local network. The majority of scan detection methods associate the activities related to scanning that originate from the same source, this is called an attribute-based detection scheme. In some cases, attackers might use techniques to escape such type of detection, for example by performing a slow scan or heavily distributed scans to evade been detected by attribute-based detection schemes, which affecting these methods performance.

Network-based intrusion detection systems are generally categorised, based on the detection approach, into signature based (misuse), anomaly and specification-based detection systems. In signature-based detection systems, they compare the monitored data to previously identified malicious behaviours; that is the attack is detected when the data under test matches malicious access patterns. This type of solution does not need to be trained and is simple in implementation and deployment. However, these solutions are only capable of detecting known attacks, which means the system should have a prior knowledge of the malicious patterns, and hence, novel malicious patterns cannot be detected under this approach. The second type of network based intrusion detection is anomaly detection, which compares the behaviour of the network to a model of expected behaviour such as the system proposed in [28] [29]. These systems are first trained with normal attack-free data to build the normal access model using machine-learning techniques, which implies a time and datasets for training. These systems detect the deviation of the network traffic from the normal models, which means they are capable of detecting new attacks. On the other hand, this type of detection approach faces challenges such as defining the normal situation in modern networks, effectiveness of machine learning for huge amounts of data to online detection, etc. The third type of detection approach cannot be classified under the latter two types such as Specification-based detection that develops a legitimate system behaviour that accepts benign behaviour and blocks any other behaviours according to specific security rules of the system.

In the field of cloud computing, intrusion detection and prevention systems are used in different parts and technologies of the cloud, and every used technology of these defence systems has its own limitations and issues [6]. Despite its many advantages like the possibility to use resources remotely and huge volumes of storage, the growth in cloud computing has further exacerbated the threat posed by cyber-attacks, as critical infrastructures and digital service providers now have multiple access points to protect [7] [8]. Hence, intrusion-detection systems in a cloud environment are in need of significant research efforts to enhance performance and overcome the challenges and issues, which will be discussed latter in this thesis.

For example, the attacker may have an account in a cloud environment and launches attacks from inside in order to compromise the hypervisor and gain access to all virtual machines' data [9][10]. Monitoring virtual machines and networks inside the hypervisor and performing intrusion detection is an issue for the conventional IDS [11]. However, security policies and transparency represent a challenge for deploying traditional IDS in the cloud environment. This is because conventional solutions tend to use fixed or static security policies, whereas, in a cloud environment, these policies are dynamically changed and updated according to the users' requirements. Despite the existence of advanced cyber-defence systems, attacks and intrusions still occur. Defence systems tried to block previously known attacks, stop ongoing attacks, and detect occurred attacks. However, often the damage caused by an attack is catastrophic and the recovery process from these threats is highly expensive [1][12].

1.1.3 Vulnerabilities assessment

To launch an attack on a system, attackers target the weak points of the system by exploiting the vulnerabilities in the software, applications, protocols, etc., to launch a successful intrusion. Therefore, for the attackers these vulnerabilities represent the doors to the targeted systems that should be searched for and discovered in order to launch successful attacks. For the security defence of an IT system, these vulnerabilities should be discovered and fixed before they become known to the adversaries. In both cases, the process of discovering these vulnerabilities is vital in which different parties try to take the lead.

Vulnerability assessment is the process of discovering, identifying, and analysing the weaknesses, bugs, and faults of the information system in assets, software, applications, procedures, and protocols. The vulnerabilities occur in information systems due to performing some activities or attaching some assets. Vulnerability assessment results in identifying bugs and linking them to the corresponding processes and assets, which enables the analysts to recognize the weak points in the system. Furthermore, this process also studies the spread of the vulnerabilities in the system, tracking security problems to discover the source of the vulnerability and reporting the weaknesses for the beneficial parties such as stockholders and security administrators.

Different frameworks have been proposed to analyse and evaluate the vulnerabilities in computer systems and score them according to predefined criteria. The goal of using such systems is to give security administration a comprehensive awareness of the system security situation and the weak points that attackers might exploit to haram the system. Different types of vulnerabilities assessment has proposed and used according to the purpose and criteria employed in the system. Information System-focused Frameworks is a type of vulnerability assessment that focuses on assessing the vulnerabilities of the system that relate and affect the broad functionalities of the system such as Vulnerability Assessment and Mitigation (VAM) [30]. Holistic framework is another type of vulnerability assessment systems that evaluates different aspects of security including the vulnerabilities such as System Security Engineering (SSE) [31]. Finally, cybersecurity-focused Framework is the last type of vulnerability assessment of systems are used only for vulnerabilities that are exploitable using the internet such as Cyber Mission Assurance Engineering (MAE) [32], Common Vulnerability Scoring System (CVSS) [33]. We will give a detailed overview of these systems in next two chapters.

1.2 Research Gaps and Motivation

This research project is motivated by the prevailing and evolving technology of cloud computing and the related challenges such as security. Such computing paradigms attracted the attention of researchers because of their benefits on the one hand, and the necessity to address their challenges that enlarge the number and type of customers. The defence systems that are used to secure the cloud environment still need more improvements as attacks continue to occur despite the fact that multiple techniques and systems have been employed for the purpose of cloud security. As these countermeasures are working on preventing the intrusions or on detecting in case the prevention has failed, therefore, providing information about the potential attacks improves the ability to prevent the attacks or complicates the attack launched by the adversaries. The motivation to address intrusion prediction in cloud computing and such paradigms was inspired by the fact prediction systems could significantly enhance the security in such environments and hence more reliability and usage. The motives behind this project and this direction of research can be summarised as follows:

a) **Limited use of Intrusion Prediction Systems**: the literature showed that small number of prediction systems have been proposed for cloud and network-based systems [34] [35]. Moreover, most prediction systems in non-cloud systems were not designed to consider the weaknesses of the system as a major aspect for predictive security evaluation. Such gap motives to study the area and try to suggest a feasible system that could perform effectively and cope with the cloud nature.
b) **Lack of integrated approach:** the absence of solutions that consider multiple aspects of the system in the prediction process, and then integrate these aspects in robust framework. Most of the solutions depend on analysing the targeted aspects separately without meaningful integration among them. For example, analysing intrusion detection results and alerts on one hand, or analysing user behaviour or system calls and processes faults on the other hand. There is no linkage between these aspects in the reviewed solutions, which causes issues in these solutions. For instance, in some cases the damage that occurred is not recoverable like leaking highly sensitive information [36].
c) **Complexity and diversity of recent intrusions**: recent intrusions and attacks showed that adversaries use different techniques to perform multiple types of attacks [37]. These results in defence systems taking a long time to identify the attacks, where the harm has already happened, and in some cases, the harm is irreversible such as data breaching. The attacker in the quest for evading detection modifies the steps of launching attacks so the signature of the attack is changed as well, therefore, previously known attacks cannot be detected by intrusion detection systems and a consequence cannot be predicted in advance.
d) **Dynamic nature of services and applications**: cloud computing featured by being a platform for many third party services that has its own vulnerabilities and security configurations, which means unexpected weakness could happen to the platform that it is not designed to deal with. Web applications are another port for bugs and faults that might affect the whole platform, which the defence systems are not weaponized to defend against.

1.3 Aims and Objectives of the Project

The ultimate aim of this research is to propose and develop a new intrusion prediction system to predict attacks on cyber-based systems in both virtualized environments such as cloud and physical cyber empowered systems.

Monitoring system security and resistance against intrusion and attacks is a notoriously challenging task, particularly in the present cyber world where boundaries and goals of the attackers are constantly changing as well as attacks types, vectors and techniques. Dynamicity and evolution are essential characteristics for the applications in computing environments such as cloud computing, which cause the majority of the challenges for existing security monitoring solutions. To address these challenges, the system should incorporate the use of a multi-methodology approach to achieve an optimal prediction of a cyber-attack, and gather data from different sources. The resultant solution should be able to assess the work of other defence systems by providing early information about the security intrusions to security administrations.

The key objectives of this research required to overcome the existing limitations in intrusion prediction systems are:

1. Develop an intrusion prediction framework to predict the probable attacks for the network-based computing environments, which will be able to operate in real-time while preserving the performance and the efficiency of the host system.
2. Create a technique to analyse network traffic from which IP and Port scans can be identified.
3. Create a technique to quantify detection threshold by analysing network traffic using only relevant data.
4. Present a suitable vulnerability assessment technique that provides an evaluation for the weaknesses in the system.
5. Validate that the developed framework and subsequent techniques are capable of effectively predicting intrusion with an appropriate time gap.

1.4 Research Novelty

1. An integrated solution that can operate on computational lightweight techniques while preserving the performance of the network-based system. The importance of our

framework stems from IDSs/IPSs failure to detect or prevent many attacks. In addition, our framework does not need training data or depend on historical data, although using such data to improve and refine prediction over time. On the other hand, in some cases the harm for the system has already happened when an attack detection is conducted. Currently, literature survey has highlighted small number of prediction systems proposed for such environments.

2. A novel use of vulnerabilities assessment and network sensing in an integral approach to produce predictions about the target system security. To the best of our knowledge, this the first time that a solution combines the vulnerability assessment with scan detection to predict potential attacks. As this solution will alarm the security team of probable attacks that are near to occur, which gives this team to act against these hazards and prevents it.
3. In-depth exploration of statistical and probabilistic scan detection methods. In order to better understand the network attacks prerequisites, we conduct an exploration study of the well-known port scan detection methods to test how suitable these methods are to work in a prediction system. Our focus was on probabilistic and statistical methods, as to our minds, it is most suitable for the research direction of the project.
4. A new algorithm for scan detection that uses statistical methods to analyse the network traffic and detect the IP and Port scan. Our solution works on selected attributes that indicate the scanning activity, which means decreasing the amount of data needed to be processed and hence lowering the calculations involved. Does not need to historical data, but rather it compares the potential scanner to other sources in the same traffic in order to recognize the abnormal access pattern to the system.
5. A statistical technique that is adaptive in calculating the threshold dynamically, thus overcoming the problem of predefined threshold calculation that might miss intrusive incidents that are under the fixed threshold. This is because it is difficult to predefine a normal behaviour in modern networks that deal with huge amounts of data streamed in and out with variant access patterns.

1.5 Thesis Structure

This section will outline the organization of this thesis, it is organised into seven subsequent chapters as follows:

Chapter Two: Background

This chapter presents in detail the basic information about the three main concepts of this research: intrusion prediction in cloud computing, port scan detection, vulnerabilities and risk assessment. This prepares the reader to form an insight into the research area in order to understand the need for such solutions and the targeted functionality that the solution promises to offer. In addition, this chapter provides an overview of the prediction systems that work in different computer science fields to give the reader a clear vision of the reality of prediction systems not the hype about it.

Chapter Three: Related Work

This chapter provides an analytical review to the existing literature with emphasis on the benefits and drawbacks of the earlier work, which enthuses the research in this direction of security systems. In addition, due to the lack of prediction systems in cloud computing, a review of intrusion detection systems that is most related to prediction systems has been presented. This review includes a critical focus on the challenges in this field in order to be addressed by the proposed work of this thesis.

Chapter Four: IPFCC Design

This chapter is dedicated to presenting the design of the proposed Intrusion Prediction Framework for Cloud Computing (IPFCC). This chapter provides details of all the novel techniques and algorithms that are specifically proposed for this solution. This includes the designed framework, the scan detection algorithm, the thresholding algorithm, the vulnerability assessments schemes, and the risk matrix for prediction.

Chapter Five: Implementation

This chapter shows the software developments and simulations that have been undertaken to develop this framework. It provides details of implantation techniques and algorithms and how all these parts worked consistently and the way that the framework is evaluated.

Chapter Six: Evaluation Proposed Methods and Framework

This chapter evaluates the IPFCC and the included techniques that are presented in this thesis against the design requirements. In addition to the demonstration of how the proposed work fulfils the aims and objective set out at the beginning of the research.

Chapter Seven: Conclusions and Future Work

This chapter describes the range of the success in overcoming the challenges acknowledged beforehand and summarises the findings of this work. Then the second part of this chapter focuses on the future work, which specifics the potential research direction, extensions, and enhancements that could be carried out based on the results of, or in relation to, this work.

Chapter 2

Background

The research conducted in this project is covering different aspects towards the proposed solution. These aspects are cloud computing, attacks, port scanning, vulnerabilities assessment and prediction. Therefore, this background chapter covers all related aspects.

2.1 Cloud Computing Security Concerns

The beauty of cloud computing is that it is built from known technologies such as distributed systems, Internet and virtualization, which means customers do not need new infrastructures or protocols to benefit from the cloud. On the other hand, cloud computing inherits the challenges and issues that its components have suffered from. Despite the rich legacy of challenges and issues that the cloud computing facing in terms of adoption, security is the major challenge [38]. Demonstrating some significant security issues in cloud computing will draw a picture of security needs in cloud computing:

2.1.1 Virtualization and Multi-Tenancy

Virtualization is the base technology of cloud computing architecture. Virtualization defines the separation of service providing from service-hosting infrastructure by building an interface to virtual resources called Virtual Machines (VM). For example, with virtual memory, the offered memory is more than the physically mounted, and this is achieved by swapping data to discs underlying [39]. The fundamental advantage of virtualization is the capacity to run various operating systems on a the same server and using the same hardware resources [40]. Virtualization benefit Cloud Computing in load balancing by means of dynamic provisioning and relocation of virtual machines among physical servers [41]. Figure 2.1 shows the virtualization architecture.

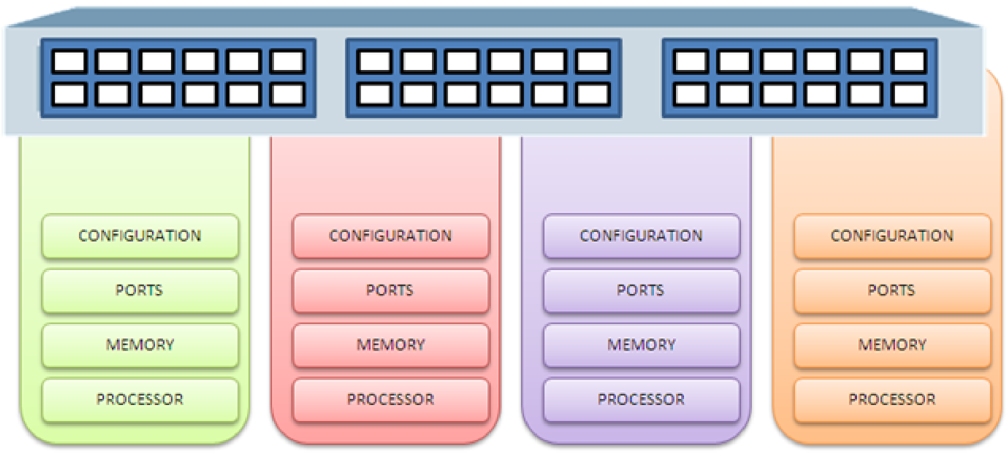


Figure 2.1 Virtualization Architecture in Cloud Computing [42]

Multi-tenancy in cloud computing can be defined as executing multiple VMs that serve different parties on the same physical server [43]. Virtualization has been implemented in cloud computing on three levels; operating system, application and hypervisor. Each of these configurations has its own threats and vulnerabilities. As stated earlier, vulnerability is a weakness point that leads to an error in the system or be exploited by an adversary to harm cloud data, services, or users. For example, CVE-2014-9718 [44] is a vulnerability that discovered in Virtual Machine Monitor (VMM) (which is a software to manage and monitor the VMs in the host) and is the reason behind the conflict in function's return value. This vulnerability causes a Virtual Hardware Logical Errors that leads eventually to expose the system to Denial of service attack (DoS) [45]. Also, Hypervisor (which is a type of VMM) is exposed to these threats in addition to native coding errors that might make Hypervisor and its guests vulnerable to adversaries' attacks such as side-channel and DoS attacks [46]. Different attacks may pose due to virtualization; two examples are presented as following:

a) *VM Hopping*: in which an attacker uses a legitimate VM to access and control another machine (the victim). In VM Hopping, the attacker can observes the victim machine resources usage, adjusts the VM configuration and finally complete control over the data that threaten VM`s confidentiality, integrity and availability [47].

b) *VM Mobility* : this is an aspect of virtualization that implies storing the components of VM on virtual disks in forms of structures that is easy to transfer from one server to another in means of portable storage [48]. This beneficial aspect is used as vulnerability to target the VM with different attacks such as Man in-the-middle [49], this attack might targeting only to steal sensitive data or maximize the damage by controlling the whole operating system. Figure 2.2 present the Man in-the-middle attack process:

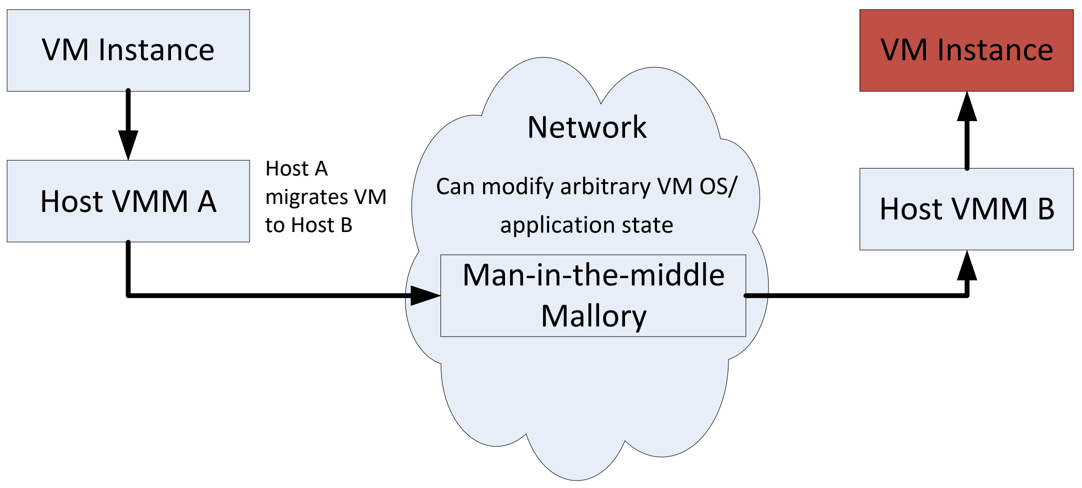

VM Instance
Host VMM A
Host A migrates VM to Host B
Network
Can modify arbitrary VM OS/ application state
Man-in-the-middle Mallory
VM Instance
Host VMM B

Figure 2.2 Man-in-the-middle Attack [49]

2.1.2 Web Application Security

The cloud computing is deployed over the Internet; hence the medium for providing cloud service is the web. The main characteristics that cloud computing offer are the ability to provide network-based access and management for the commercial software from remote central point, which allows the clients to access that software via the web. This characteristic is an essential requirement for cloud to be accessed and managed over the web [50]. Therefore, the security holes in the web browsers and protocols create vulnerabilities to the cloud applications and services. This leads to affect the costumers that uses the cloud and their security. Verizon Business stated in their report [51] that 39% of the hacking activities were targeting the applications, software and services. Such vulnerabilities creates a new type of attacks that evade the classical defence at network level (intrusion detection and intrusion prevention systems) and needs an application level defence [52]. Different attacks listed by the Open Web Application Security Project [53] as follows:

a) Cross-site scripting.

b) Injection flaws like SQL.

c) Broken authentication.

d) Insecure direct object references.

e) Invalidated redirects and forwards.

f) Insufficient transport layer protection.

g) Failure to restrict URL access.

h) Insecure cryptographic storage.

Cross-site scripting (XSS) is an example of such attacks that recently targeted cloud computing service providers and endangered millions of costumers accounts and data [54]. This attack enables the attacker to inject their malicious scripts into an input field in HTML pages that enforce host to executes malicious activities at the host like expose hack-related information [55]. There are two types of this attack; non-persistent (or reflected) cross-site scripting and persistent (or stored) cross-site scripting [56]. In the first type, the user input is reflected directly through webpage by a server side coding with no efficient integrity control. The Persistent or blind is the second type of XSS that inputs malicious code, stored at the server, and then displays at webpage of users who are accessing the page via the infected server. The latter type is the most dangerous type because the malicious code remains in the infected page and any user visit the page will be infected as shown in figure 2.3.

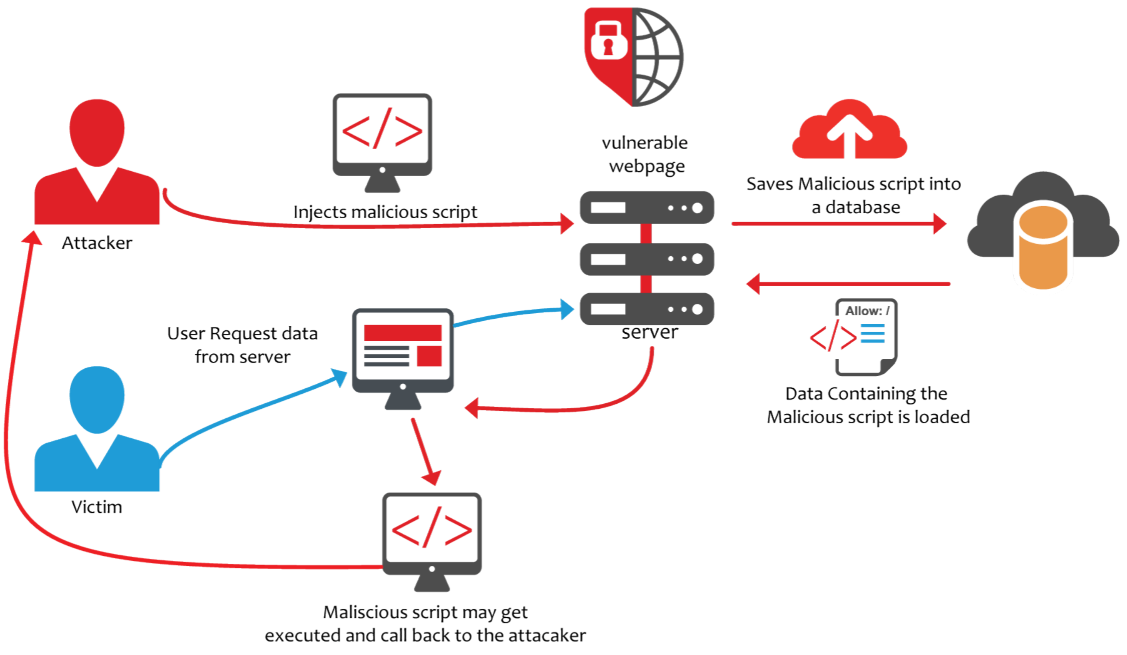

vulnerable webpage
Attacker
Injects malicious script
server
Saves Malicious script into a database
User Request data from server
Allow: /
Data Containing the Malicious script is loaded
Victim
Maliscious script may get executed and call back to the attacaker

Figure 2.3 Blind Cross Site Scripting (XSS) Attack [56]

2.1.3 Abusive Use of Cloud Services

This term refers to the use of the cloud services for illegal actions that serves malicious activities and purposes. Offensive use of cloud resources might be committed by different customers to serve different intentions. The first form of abusive usage of cloud is exploit a vulnerable VM to insert a malware that infects the users and cloud servers for harm or exploit the infected machines. An example of this form is VM-based rootkits [57], which is a software that is employed by other malicious code to evade the detection by antiviruses and security systems on the infected machines. These malicious software types disguise themselves by hiding related operating system objects (such as files and registry entries) from diagnostic programs. Figure 2.4 display an example of VM rootkit.

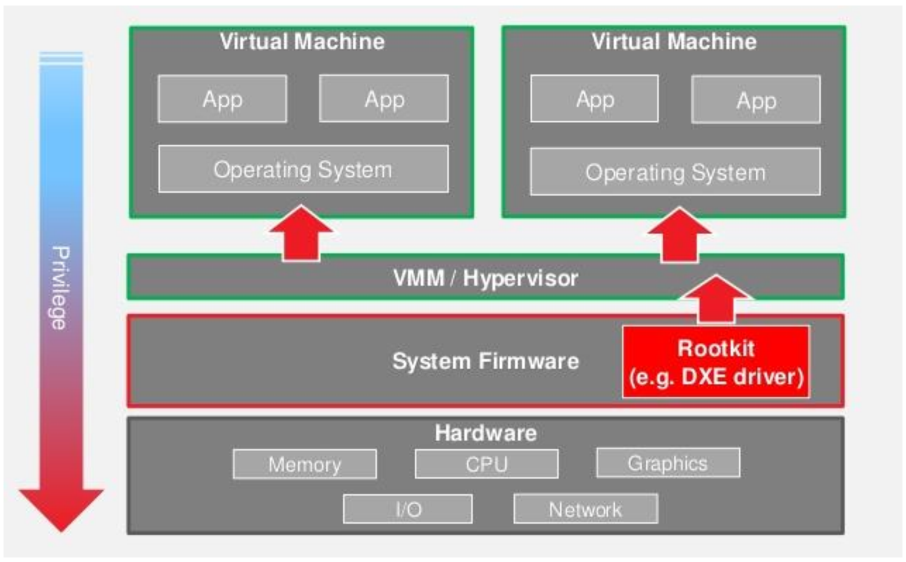


Figure 2.4 Virtual-Machine-Based Rootkit [58]

The second form of abusing the cloud service is by using an infected cloud machine as a base to attack other machines. In this form, the infected machine is not harmed by the malicious but it is used to harm other machines. The attacker achieved this form by exploiting a vulnerability that enables the attackers to install a backdoor that allows them to install malicious codes on victim machines without the owner’s knowledge [59]. The malicious codes that may be installed as Botnet turns the infected machine into a ‘Zombie’ and use it as a remote attack station. The infected machines might be transformed to be a command and control server (C&C) that operates and controls the botnets for nefarious purposes. This form of cloud abuse is used in 2009 to attack Amazon EC2’s cloud services by Zeus bot [59].

2.1.4 APIs and Software Interfaces Insecurity

Cloud computing providers offer the customers a group of programming interfaces and APIs that are used to contact and collaborate with the cloud. The processes of service providing such as provisioning, management, monitoring, and orchestration are executed by these interfaces. Insecure APIs damage cloud services as the security and availability of these services depends on APIs security [60].

Many issues and flaws may affect the security of APIs and interfaces such as reusable token or passwords, fixed access controls, improper authorizations, unknown services or API dependencies [60]. A recent example of such attacks is the security violation at Moonpig [61], where millions of costumers endangered due to using insecure API. As the latter company used an API in their Android application that used a static set of credentials, and by just alternating customer`s ID, the attacker will be able to send millions of requests to company. Different aspects must to observed to secure the APIs in cloud such as Transport security, Authentication and authorization, Code and development practices and Message protection [62]. The cloud should use a Transportation protocol that is secure such as Secure Sockets Layer (SSL), especially when sensitive data are transferred. APIs tend to be lightweight programming pieces, therefore, when APIs are used in authentication and authorization, they should guarantee using a sophisticated practices such as two-way authentication attributes. In Code and development practices, any APIs that uses certain languages such as JSON and XML to pass massages or receive user inputs should be well tested for vulnerabilities such as XSS.

2.1.5 Data Confidentiality

This is an essential security requirement that states that only authorized people or applications have the right to access sensitive data. In cloud, the possibilities of data to face the risk of compromisation due to multiple access points that result from the multiple parties, devices, and software that involve in cloud service providing. As the cloud is responsible for controlling the data, the risk of data jeopardize increases due to the number of participated parties that are authorized to access the data. Different issues that affect data confidentiality are emerged in cloud such as multitenancy (as discussed in section 2.1.1), data remanence, application security and privacy [60]. Data remanence is a form of data that still electronically stored in the cloud although it has logically marked as deleted or removed. As object reusability is key aspect of the cloud, uncontrolled use of this aspect might lead to serious impact on data confidentiality.

The data remanence is occurs due to the lack of physical separation between different users on the same server, which is possible in two situations; the first is when a user unintentionally retrieve old sensitive data; the second situation achieved by malicious attacker who acquires large amount of storage in quest of mining for sensitive data [63].

Protecting the clients' account privacy from theft is vital duty of cloud computing, and Due to the dependency between data confidentiality and authentication method, it is crucial to choose a strong solution because weak authentication leads to major problems such as losing the account or been controlled by an attacker. Authentication is the process of building confidence in the client identity that is provided electronically to the cloud [64]. Different types of authentication are used in the cloud such as [65]:

a) *Username and password*: this is the classical method for granting access to the user by using an identifier and password.

b) Multi-factor authentication (MFA): In this method, a secondary method such as biometric authentication in addition to the username / password method, where these achieved based on steps according to the number of used factors.

c) *Trusted Platform Module (TPM)*: This international standard is hardware security component built into computers, and consists of machine authentication, hardware encryption, signing, secure key storage and attestation [66].

d) *Single sign-on (SSO)*: this type of authentication allows the client to use single set of credential to access all applications and services on the cloud, which is authorized to access. Examples of this type of authentication is lightweight directory access protocol (LDAP) directory, Kerberos and the security assertion mark-up language (SAML) [67].

e) *Biometric authentication*: this method depends on identifying the user using physiological or behavioural characteristics [68].

Figure 2.5 shows an example of authentication process in the cloud.

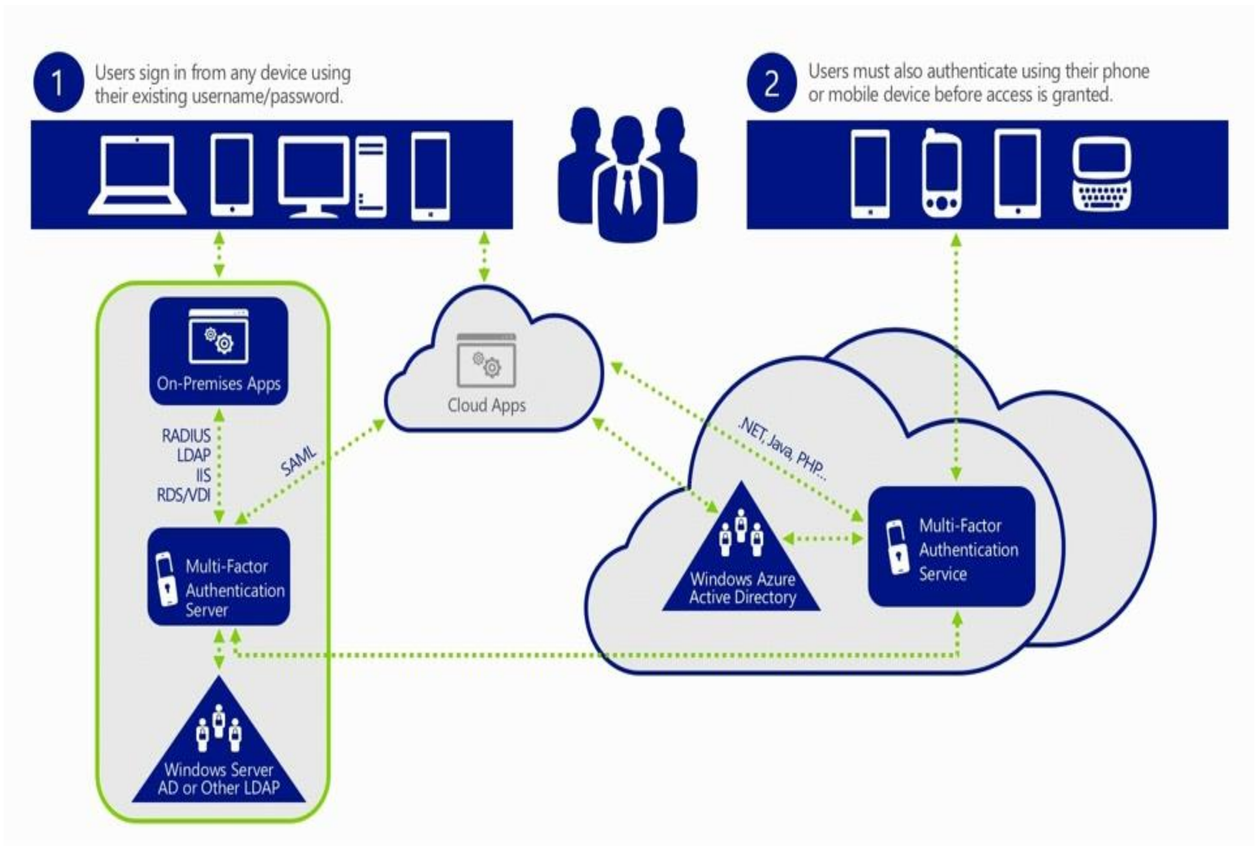

1
Users sign in from any device using their existing username/password.
2
Users must also authenticate using their phone or mobile device before access is granted.
On-Premises Apps
RADIUS
LDAP
IIS
RDS/VDI
SAML
Multi-Factor Authentication Server
Windows Server AD or Other LDAP
Cloud Apps
.NET, Java, PHP...
Windows Azure Active Directory
Multi-Factor Authentication Service

Figure 2.5 Example of Cloud Authentication Method (MS Azure) [69]

2.1.6 Data Integrity

This is an essential characteristic of cloud security and information systems in general. Integrity refers to the modification of data, software, and hardware is only allowed by authorized parties or in authorized ways [70]. This is implies data should be protected from unauthorized insertion, modification and deletion. Authorization is the process of controlling and assigning access levels for each authenticated user to the services it authenticated for, which preserves the security of cloud resources. As the cloud computing deals with multiple services and massive number of users, managing access control become more complex that each cloud service provider has to develop and implement their access control system that comply with security policy and service providing pattern. In conventional computing systems, there are three types of access control approaches; Discretionary Access Control (DAC) that requires to assign permission to users that need access [71]. In DAC, the users could transfer their permissions to other users. The second type is the Mandatory Access Control (MAC), in which the system restricts the ability of a user (subject) to access or perform operation on data (object) or service. This means that the user has no ability to delegate permissions, and only the security administration control the access [71]. Finally, the Role Based Access Control (RBAC) that assigns roles to users, then each role is assigned a set of permissions [72].

In cloud computing, different models and solutions has been proposed for tackle with access control in academia and industry. In business field for example, Amazon Simple Storage Service uses a role based access control to manage access to their storage service [73], as shown in figure 2.6.

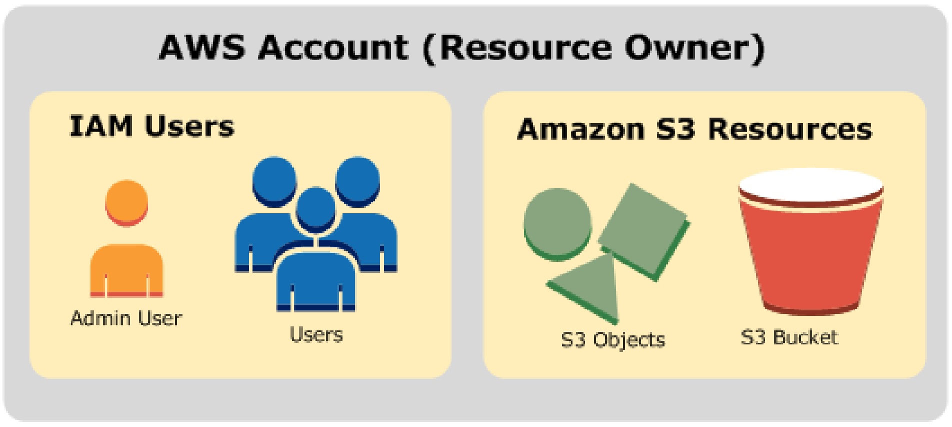


Figure 2.6 Amazon Access Management Overview [73].

Amazon in their approach, are assign roles to the users using Identity and Access Management (IAM), and then assign the permissions to the resources that refers to as Buckets and objects. IAM best practices is a guide that defines the roles for users such as Grant Least Privilege, Enable MFA for Privileged Users and Root User Access Keys.

In academia, different models for access control for cloud computing exist, these solutions are an adaptation and modification of the classical models to suit the cloud nature. To adapt access control to cloud computing, researchers in [74] combined the trust relationship and role based access control system. Others [75] optimize an RBAC model for cloud by support a Distributed RBAC with only one manage role. In [76] a semantic scheme for access control by enriching the RBAC model with semantic web technologies. In [77] they merge between role based and task based systems to produce a Task-Role-Based Access Control, where additional function of activate/deactivate permissions according to process tasks and requirements. Researchers in [78] used security tags that are a RBAC enriched with fine-grain information such as classification, location, timestamp and unique number.

2.1.7 Availability

Availability refers to the capability of authorized entity to use the system`s services on demand [79]. This implies the ability to provide the services even in cases of misbehaviour conducted by malicious parties. The availability is the third pillar of widely applicable security model CIA (confidentiality, Integrity and Availability). It is essential for Cloud computing to provide adequate availability of their services [80]. This poses an obstacle to use the cloud, as many organizations worry about the availability of their service on the cloud that some of these businesses worry about using cloud computing. Recent example of availability problem in cloud is what was announced by Amazon that their cloud storage service is not working due high error rates. This leads to services of different popular businesses such as the travel agent Expedia, the learning site Coursera [81] has completely shut down . The error occurred in Amazon storage service, and as a consequence, different Amazon cloud services has been affected such as Elastic Compute Cloud (EC2) virtual machines, RedShift, Simple Email Service, Workdocs, WorkMail, AWS Auto Scaling, CodeBuild, CodeCommit, Elastic Beanstalk and AWS Lambda [81]. Table 2.1 shows the service outage for major cloud service providers:

Table 2.1. Service Outage In Some Cloud Providers [80]

Provider	**Reason**	**Duration**	**Date**
Amazon S3	Authentication service overload leading to unavailability	2 hours	2/15/08
Amazon	Single bit error leading to gossip protocol explode	6-8 hours	7/20/08
AppEngine	Programming error	5 hours	6/17/08
Gmail	Site unavailable due to outage in contacts system	1.5 hours	8/11/08

Usually, the management of cloud computing services is performed by single provider, which is actually a single point of failure, despite the use of multiple cloud datacentres in different geographical locations and multiple network service providers [80]. Contrarily, to provide a high-availability for computing service, a single point of failure should be avoided [52]. The availability is affected by volumetric attacks that lead to exhaust the service resources limits such as Denial of Service attacks.

2.2 Cloud Computing Attacks

This section will demonstrates some of these attacks as follows:

2.2.1 Distributed / Denial-of-Service (Dos and DDos)

Distributed and Denial-of-service attacks are types of hostile intrusive actions targeting online servers. These attacks are aiming to damage the availability of the targeted host, router or even an entire network. DoS attacks force the victim to perform extensive computations by exploiting vulnerability or flooding with massive loads of network noise. This stops the victim from providing the service for a while, and hence damaging the hosted services [82]. These types of attacks affect and target cloud environment on many levels specifically in data centre operations, incident response, application security, and virtualization [83]. An example of DDoS attacks is the one that targeted Dyn.Inc (a major Internet performance management company) DNS infrastructure that made a huge distribution of services for major technology companies such Twitter, SoundCloud, Spotify, Netflix, Reddit, Pagerduty, Shopify, Disqus, Freshbooks, Vox Media, PayPal, Etsy, Github, etc [84]. This attack started at 11:10 UTC US (Coordinated Universal Time in united states of America) east and targeting the DNS infrastructure that caused DNS query latency and delayed zone propagation during this time. The attacked has been mitigated and the services was restored at 13:20 UTC.

In general, launching D/Dos attacks can be divided into two types of methodologies; vulnerability exploiting attacks and flooding attacks. For example, network-based vulnerability exploiting attacks exploit the weaknesses or bugs in network protocols or applications at the victim machine to cause extreme memory consuming, overload CPU processing, system halt or restart, or general system throughput reducing. On the other hand, network-flooding attacks are executed by sending huge amounts of network packets that cause legitimate traffic to be disrupted and dropped at the victim network [85]. Flooding attacks are the most common forms of D/DoS that target the cloud. These attacks can be categorised into three categories; volume-based attacks, application-based attacks, and low-rate DoS attacks [86]. In addition there are new forms of Dos attacks such as network under-provisioning attacks [87] and Resource Exhaustion attacks [88].

Recently, many types and techniques have been developed and used to launch D/Dos attacks in cloud computing that have been studied extensively and methods discussed to counter these attacks such as filtering the ingress/egress data, input debugging and hash-based IP trace-back [89] [90] [91]. Volume-based attacks are the most popular and basic DDos attacks [86], where the attacker typically overwhelms the victim with a huge number of packets or connections, flooding the network equipment, servers, or bandwidth resources. Previously, this type of attack was launched by many compromised machines that form a botnet. Typically, an attacker compromised a machine or server to work as a central control unit (command and Control CC) and used this unit to compromise more internet-connected systems (Zombies) that the attacker uses to initiate attack against the goal victim. Attackers use different tactics to build their botnet such as tricking the system user to drive by downloading, exploiting vulnerable web browser, or luring victim to run malicious software. The communication between the Zombies and CC then the attacker run through covert channels mostly achieved using internet relay chat. Figure 2.7 displays the basic bot net architecture.

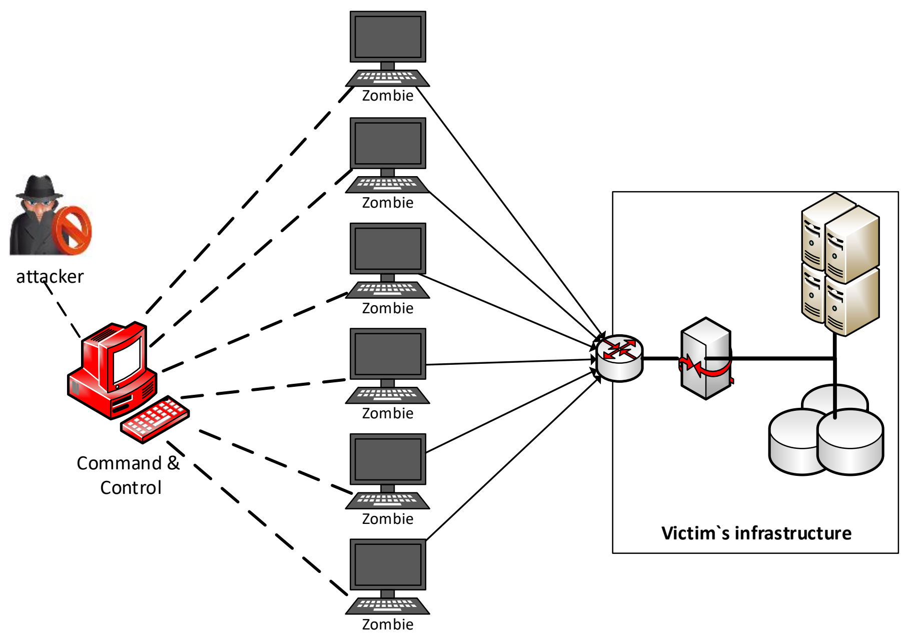


Figure 2.7 Basic Botnet Architecture

With the advent of the cloud, the same technique is used to launch volumetric attacks from datacentres of cloud providers, where the attacker infects cloud-based systems or even rents it legitimately. Cybercriminals take advantage of the massive capabilities that the cloud has to offer, as the cloud represents a great inexpensive powerful platform that these criminals use to launch illegal activities. Figure 2.8 clarifies this idea:

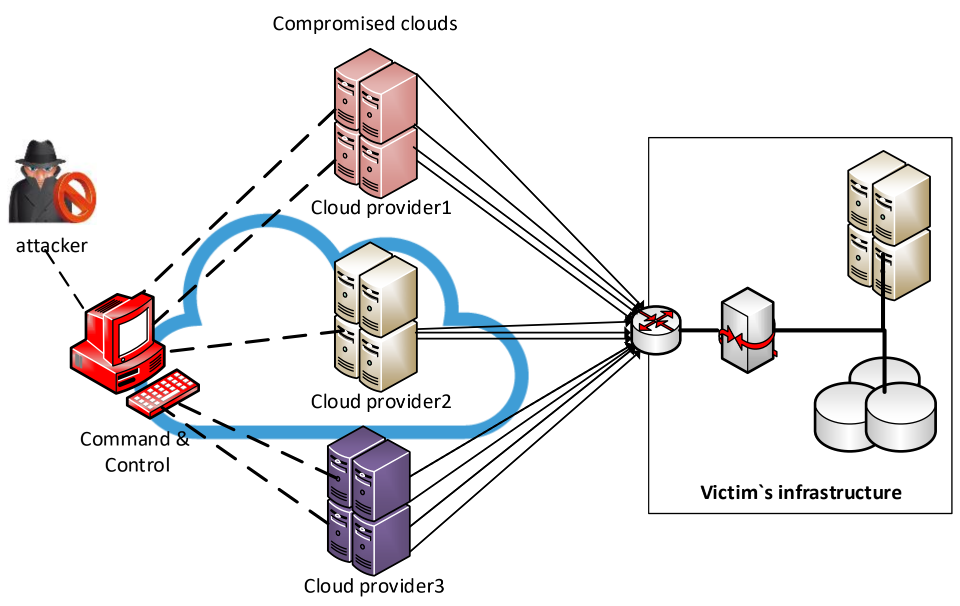


Figure 2.8 Compromised Cloud Datacentres

The main target for D/Dos attacks is to damage the availability of cloud services. In regards to companies and enterprises, this means affecting the business of the attack`s victims and harming them financially and on the reputation level. Moreover, affecting all services that share the same host of the victim and cause them to be out of service.

2.2.2 Malwares

It is an abbreviated term formed from MALicious and softWARE, and refers to any software that is programmed to execute malicious activities. This software can be deployed remotely using attacks that are hard to expose their sources [92]. These attacks are fast spreading that they occur every three minutes in different organizations around the world [93]. Malwares are typically used to steal financial information such as account credentials, numbers of bank cards and accounts, and intellectual property such as private applications, and technical algorithms and secrets [92]. The impact of malwares may be more severe in some advanced types of malwares such as Droppers, since in this type of Malwares they download additional malicious files soon after they have been installed. To achieve this task, Droppers disable security software at the victim system as well as the updates, and make a list of all files and folders in the system [94].

In the cloud environment, many problems and issues are caused by malwares. A malware attack can take the form of injecting malicious services or VMs into the cloud; the cloud system's operator is tricked to execute the malicious code, and then legitimate users' requests are redirected to the malicious service/VM, which compromises the security for the cloud and legitimate users [95]. On the virtualization level, a security management requirement is to save a virtual machine image, which is why when, infected by a malware, it will preserve and propagate the malware every time infected images run [48]. Infecting a folder that is synchronized with a cloud-based storage service means infecting all devices that connect to this service and it is hard to stop this malware as the source is out the cloud [94].

2.2.3 Advanced Persistent Threats (APT)

This term refers to a sophisticated multistage attack that uses multiple techniques and vectors of attacking the victim, all these attacks carried out unnoticed to the security systems and evading detection in order to maintain control over target systems as long as is possible [96]. The steps of APT are normally a sequence of sub attacks that serve the ultimate goal of APT. The typical model to describe APT attacks is the kill chain method [97] [98]. Figure 2.9 shows an abstract APT attack structure as follows:

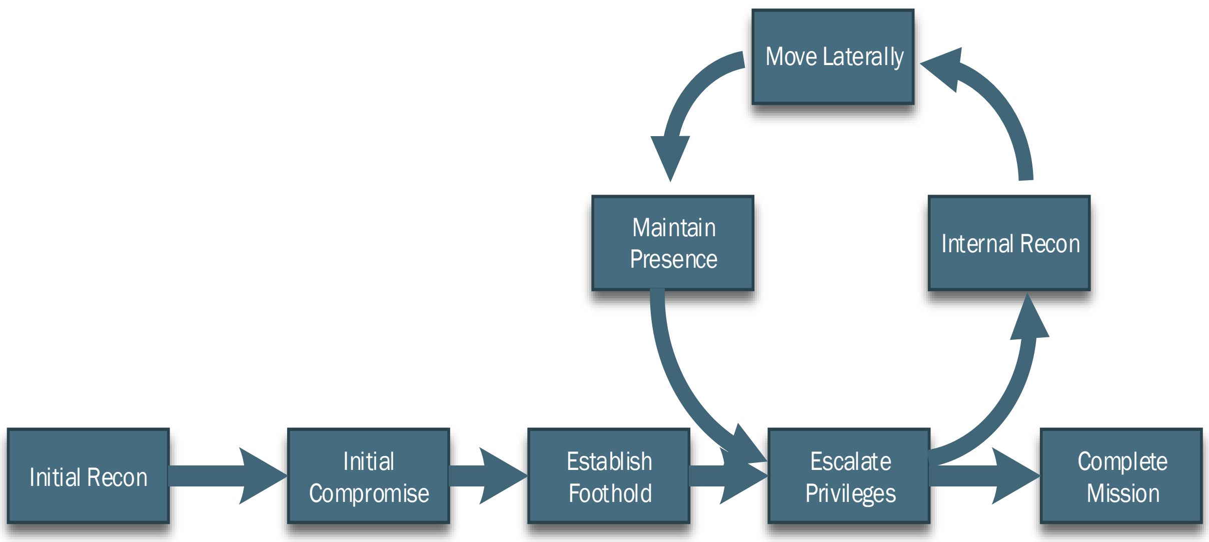


Figure 2.9 APT Structure [99]

In the cloud, some application level vulnerabilities might be exploited by an attacker during the intelligence gathering and lateral movements. These two steps of APT are significant as they are key stages for compromising or further compromising the system. An extensive security analysis for a leading commercial cloud [100] showed some serious vulnerabilities such as sophisticated Malwares that are capable of key logging, process monitoring, and stealing information. Other serious vulnerabilities that have been found are the presence of backdoors and leftover credentials, and unprotected private keys. Attackers can use the latter vulnerabilities not only to hack into the cloud but also to maintain a stealth presence in the infected VM and develop this existence to other VMs in the same host or even the same cloud.

Although other studies such as [101] suggested that using mobility-enabled secure cloud might theoretically limit or diminish the effect of APT. The latter suggestion depends on the idea that the ATP attack surface is restricted to a small impact area that includes the allocated resources only. However, the effectiveness of this solution actually depends on how it is applicable to public commercial clouds, and is yet to be examined against inherited vulnerabilities.

2.2.4 Brute-Force Attacks

Brute-force attacks had been ranked as the top three threats in cloud computing in 2013 [102], while McAfee Labs predicted a massive brute force attack in 2017 [103]. It is the process of sending predetermined values to a server and analysing the response [104]. In other words, an attack when an adversary tries to guess the victim`s sensitive information (especially the password) using a reiterative approach of trial and error [105]. There are three types of brute forces attacks; Dictionary attacks, Search attacks and Rule-based search attacks. In dictionary attacks, an automatic software tries to predict user name and password from a dictionary file, which contains words gathered by the attacker about the user or unique words that are available publicly. Search attack checks likely mixtures of character sets and variety of password length. These attacks take a long time due to the huge number of variations that could be tried. Rule-based search attacks use rules to generate potential passwords depending on partial pieces of information of the victim or from adjusting pre-prepared mask words in the input [105].

2.2.5 Insider Attacks

The insider attacks are have risen from 8% in 2014 to 10% in 2015 of the top causes of data breach [106]. Insider attacks can be defined as a deliberate misuse carried out by a person with a current or former relationship to an organization who owned access to organization`s systems and information, for the purpose of affecting integrity, confidentiality, or availability of these systems and information [107]. Normally, the insider attacks are accomplished on three aims namely Sabotage, theft of intellectual property, and fraud [108]. By sabotage, the insider intends to impose damage to the organization or to a hostile individual. In Theft of intellectual property, an insider might steal and exfiltrate royalty information for the insider`s benefit or to harm the organization by exposing secret information to rivals. With Fraud, the insider performs unauthorized data alteration and updating for personal benefit, or identity disguise such as identity theft and electronic cards fraud. In cloud computing, insider attacks could be launched by three types of insiders [109] namely a hustler cloud administrator, an employee that uses vulnerability in the system to gain unauthorized access, or a cloud user that has a legitimate account but uses it to abuse the cloud infrastructure.

2.3 Intrusion Detection, Prevention and Prediction In Cloud Computing

Although Cloud computing inherits the classical problems of computing, however, the classical solutions for these problems do not suit the cloud. Most reasons for the classical solutions failure stem from the nature of the cloud that is structured according to the new deployment methods and different levels of infrastructure and software that operate to provide the services, unlike the classical solutions that tailored based on defined configurations. In addition, the responsibility of security administration is splitted between the provider and the costumer (service owner); while the cloud provider responsible for securing the infrastructure and platform, the costumer is responsible for securing their services or pay for third party to take this responsibility. This leads to contradictions in policies and defence systems that supplied by different parties. Furthermore, different concepts that contributed into the failure of classical solutions in cloud such as Virtualization, multi-tenancy, service level agreement (SLA), cloud scalability, etc. In this section we will discuss the intrusion related systems and the reason behind the need for intrusion prediction.

2.3.1 Intrusion Detection/Prevention Systems (IDSs, IPDs, IDPSs)

A recently published survey shows that the second main security concern is intrusion detection and prevention in cloud environment [110]. According to [111], a definition of IDS in cloud computing is a software or hardware that automatically detects intrusion and proactively discovers potential intrusive points. An intrusion detection and prevention system IDPS is software or hardware that include all abilities of an intrusion detection system, in addition it can halt possible events as well [111]. Unlike IDSs, IPSs can act upon detected threats and stop them damaging or behaving maliciously in the cloud. IDPS is a mixture of the two types [112] and is capable of altering the configurations of the security environment or removing the malicious part in order to keep working normally [111], in addition it reports the intrusion and action upon it to security administration of the cloud [112].

2.3.2 Types of IDS/IPS in Cloud Computing

There are four main types of IDS/IPS that are deployed in cloud environment [110], these types are differentiated according to the location where the IDPS is located. This section will demonstrates these four types:

a) *Host based intrusion detection system*: this type performs monitoring and analysing of the data gathered from the host machine. The detection intrusion process is done by gathering data like system files, network events, system calls, etc. and according to the abnormal behaviour recognition; a report of attack is issued. Normally HIDS is installed on the host machine, virtual machine or supervisor to detect malicious behaviour by monitoring and analysing log files, access-control rules, and other related information. The effectiveness of HIDS is dependent on the selected host characteristics to be monitored. Different detection methods and solutions has been used in this HIDS such as behaviour based method [113] to detect known attacks and an Artificial Neural Network (ANN) to detect unknown attacks. In addition, algorithmic methodology is used for HIDS by exploiting multi-level IDS that analyses user behaviour and log management in host operating system (OS) [114].

b) *Network Based Intrusion Detection System NIDS*: in this type of IDS is implemented on network traffic to detect intrusive events like DoS attack, port scanning, and cracking into the computer. The mechanism of detection is to compare the present behaviour to an already observed behaviour in real time. NDIS is positioned next to the firewall and the network, and in the cloud, it is placed in the cloud server that deals with external networks. As an example of NIDS the work proposed in [115] that is a signature based IDS to detect the DDoS. The IDS in positioned in virtual switch to log the traffic and drop the packets that match a known signature. Another solution is Cloud Intrusion Detection Service (CIDS) [116] proposed using Snort that is a signature based IDS that deployed at the network level. This solution is intended to work as on demand cloud service that any host desired to benefit from this service should be subscribe to the CIDS.

c) *Distribution Intrusion Detection System (DIDS)*: Distributed IDS is composed of several IDS (e.g. HIDS, NIDS, etc.) across a wide range network that interact mutually or to a server where network observing is active. A component of intrusion detection collects local host data and modify it to be ready for using by a central analyser, which in turn analyses aggregated data using both anomaly and signature based methods for analysing. The DIDSs are characterised by the cooperative approach among the distributed units that inform each other when an intrusion occurs as in [117]. In this solution, an individual NIDS is deployed at each cloud node that act as an agent. These agents share information at the occurrence of the intrusion in a specific area of the cloud such information that contain the attack signature and severity. In [118] a mobile agent based IDS has been proposed to suits the cloud computing environment and fulfil the user`s security demands. In each VM, an agent is deployed that sense the traces of attacks that on detection sends alerts to central Snort IDS that judge the incident and block further attacks.

d) *Hypervisor based Intrusion Detection System*: hypervisor is the platform that runs the virtual machine (s), hypervisor based intrusion detection system performs monitoring and analysing communications in different directions among virtual machines, with hypervisor, and inside hypervisor based virtual network [119]. An example of this type of IDS is what proposed in [120] that called VM introspection based IDS. This solution works by monitors the hardware conditions, activities and software situations of host, which provide vigorous view of the system than other types of IDSs. The structure of this solution is shown in figure 2.10.

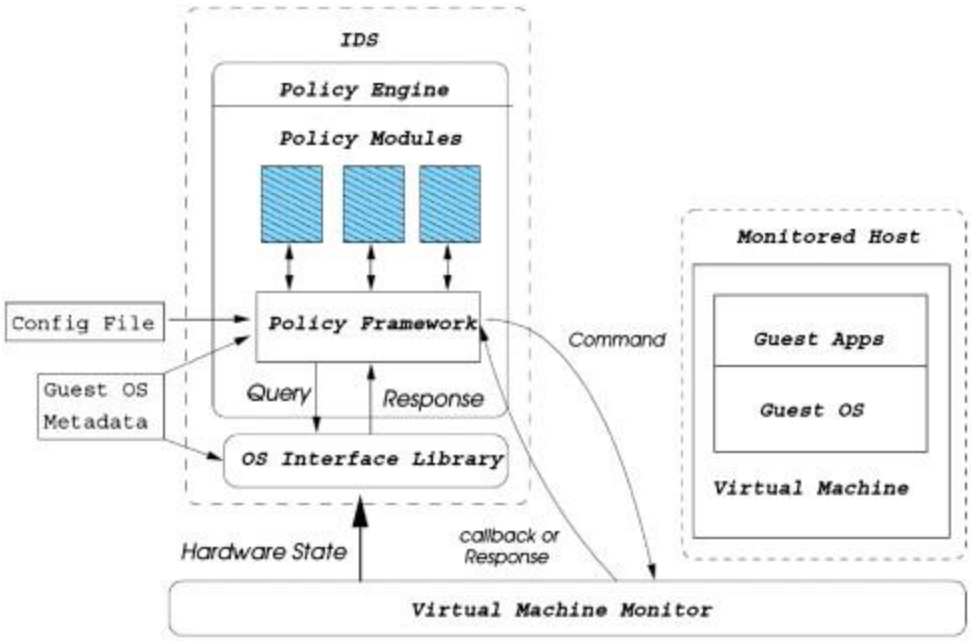


Figure 2.10 A View of VMI-Based IDS Architecture [120]

2.4 Port Scan Detection

Before exploring port scan definition and its methods, it is crucial to demonstrate the importance of port scanning and its role in the launching of a successful attack, and clarify the port scan definition, methods, and detection methods.

2.4.1 Attack Modelling

Although the area of cyber-attacks is well studied and investigated, it is hard to find a formal model for cyber-attacks that can be applied to all cyber-attacks presented in the literature. This is probably because there are different types of attacks comprised of different attack steps, surfaces, platforms, and destinations. In [121] the basic steps that an attacker should follow to achieve a successful network attack are stated as follows:

1. ***Information gathering***: In this step, the attacker starts to prepare for attack by collecting information about the vulnerability of the victim side via the network. The attacker searches the collected information for the vulnerabilities that serve the goal of the attack.

2. ***Assessing vulnerability***: Based on the collected information and what vulnerabilities are found in the victim side, the attacker starts to compromise some nodes in the network as an initial step for launching the attack.

3. ***Launching the attack***: At this step, the attacker performs the attack on the targeted machines with the aid of the compromised nodes.

4. ***Cleaning up***: This is the final step when the attacker completes the attack and tries to remove the attack history by eliminating the registry entries and log files from the attacked machines.

The latter steps are basic and focus on networks, and only cover very basic attacks. However, to cope with the sophisticated attacks that have recently been launched, a general model derived from the military terms called ‘Kill chain’ proposed by Lockheed Martin Corporation [122] can be a considered a general model for remote cyber-attacks that all attacks adhere to a number of attack kill chains [123][124][125][126].

The kill chain phases [122] shown in figure 2.11 are proposed to characterise a computer network attack as follows:

a) ***Reconnaissance***: Investigating and exploring the internet to find targets to be attacked. Normally, gathering information about the target from different sources to collect information such as email, IP, network mapping, social relationships, etc.

b) ***Weaponization***: A crafted Trojan injected with an exploit forged into a deliverable payload using a vulnerable application (weaponizer). Most recently, text files such as Adobe PDFs and MS words files have been used as weaponized deliverables.

c) ***Delivery***: Is the process of transferring the weapon to the target. Some forms of attacks use delivery means such as network traffic, file attachments, infected websites, removable storage devices.

d) ***Exploitation***: Is the process of activating the exploitation in the target system. This includes targeting specific vulnerability in applications or operating systems that an exploit is designed to utilize.

e) ***Installation***: Is the process of establishing a remote connection by perform the exploitation with the victim via backdoor or covered channels that are initiated by the Trojan or the exploit.

f) ***Command and Control (CC)***: A unit plays the role of a commander residing in a compromised host, maintains a connection with victim, and performs the orders requested by the attacker. CC is an intermediate station between the attacker and the victim, used by the attacker to disguise the identity of the attacker and preserve connection with the victim. Further reconnaissance is done to explore the valuable destination inside the network, and perform lateral movements.

g) ***Actions on Objectives***: At this stage, the attacker starts to achieve the original goal of the attack. Such sophisticated attacks are normally aimed at data exfiltration, which involves gathering, encrypting, and extracting information form the target machine. This will include data integrity and availability violations to enable data exfiltration to be achieved successfully. In some cases, the object of the attacker is only use the current victim as a station to intrude into other more valuable targets on the same network or environment.

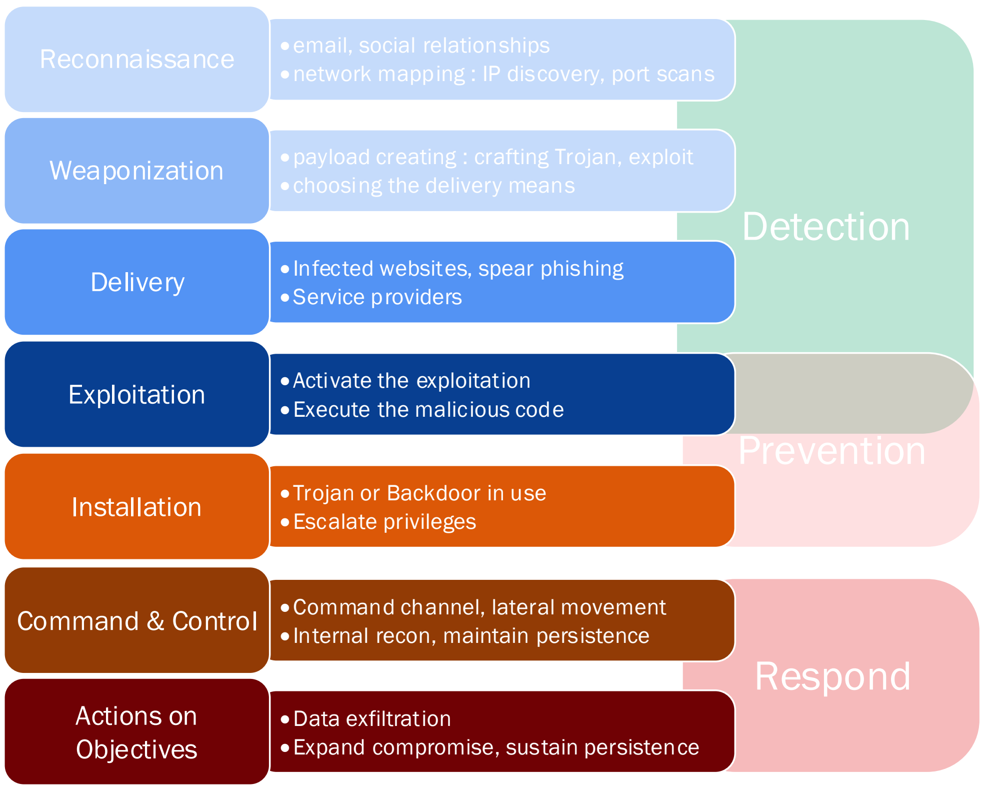

Reconnaissance
• email, social relationships
• network mapping : IP discovery, port scans
Weaponization
• payload creating : crafting Trojan, exploit
• choosing the delivery means
Delivery
• Infected websites, spear phishing
• Service providers
Exploitation
• Activate the exploitation
• Execute the malicious code
Installation
• Trojan or Backdoor in use
• Escalate privileges
Command & Control
• Command channel, lateral movement
• Internal recon, maintain persistence
Actions on Objectives
• Data exfiltration
• Expand compromise, sustain persistence
Detection
Prevention
Respond

Figure 2.11. Attack Kill Chain [98]

We notice from the two attack models that gathering information is the basic step that any attacker intending to launch an attack should perform. Information gathering is crucial because how the attack type will be launched is defined according to the found vulnerabilities, and for identifying the target availability. Therefore, in the next section we will investigate the port scanning.

2.4.2 Port Scanning Definition

Port scanning is the first step of reconnaissance that attackers use to locate the target network or machines and most prevalent approach to discover the open ports and the associated vulnerable services that can be exploited to hack the target system. A port scan is a technique of identifying the available services on a host or a network by testing the connections requests replies [127]. This definition clarifies the purpose of port scan scanning as a tool for initial information gathering. A further understanding of port scanning can be perceived by understanding the mechanism of connecting two machines and the way used by the attacker to exploit this for port scanning. A three-way handshake is a method used to establish a connection between client and server using the Transmission Control Protocol (TCP). The process of three-way handshake is illustrated in figure 2.12.

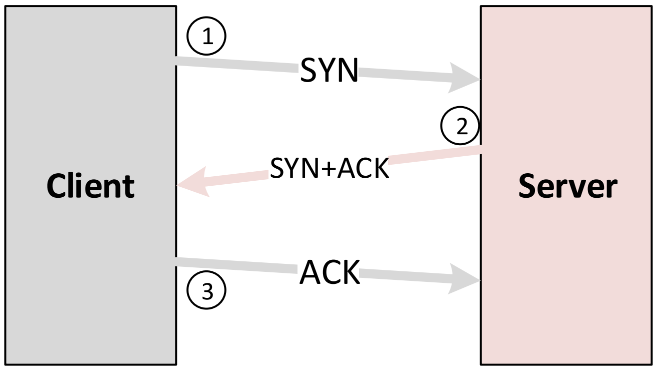


Figure 2.12 Three-way Handshake

To initiate a connection the client sends a TCP packet with SYN set (SYN packet for short) to a specific port on the server machine. The server replies with SYN+ACK packet to the client (by allocating memory space) on the port specified in the SYN packet. In the third step, the client replies with ACK packet, then the connection is set established on both sides. In the simplest types of port scan, the attacker sends SYN packets to all ports in the server (ideally 0-65535) or to well-known ports (0-1024) without completing the third by replying with ACK packets. Any responding port with SYN-ACK packet will inform the attacker that this port is open and more investigation will done by the attacker regarding the services associated with the respondent ports. Next, we will demonstrate the methods and types of port scanning, and the methods used to detect them.

2.4.3 Types of Port Scanning

Although port scanning might be used for benign intentions by system administrators and network engineers, we are interested in port scans that are the initial step of information gathering leading to malicious activities. In this context, attackers launch port scanning in multiple manners according to the type of target and the numbers of ports being investigated. This could be achieved as follows [128]:

a) *Vertical scans*: in this type, the attacker scan several ports in a single destination host. This means the attacker is interested in a specific host and in search of open ports on that specific host.

b) *Horizontal scan*: in this type, the attacker scans the same port on several hosts. This means the attacker is interested in a specific port and searches the hosts that open this port. Normally this type is used when a vulnerability is discovered on a specific port (more precisely in a service listening on this port).

c) *Block scan*: this type is a hybrid scan that searches for multiple ports on multiple hosts.

In order to evade the detection and act in hidden behaviour, the attackers uses different techniques in scanning as in follows:

a) *Stealth scan*: this type scan is executed when attackers desire not to reveal much information about their identity; therefore, they turn to incomplete connections such as (*FIN, Xmas, and Null*). As these scans do not use the three-way handshake they are difficult to log and hence to detect. Other stealthy techniques include modifying the TCP packets, changing the timing of sending scan packets, hiding the scanner IP amongst spoofed IPs, etc. [129]

b) *Distributed scan*: this scan is another technique for evading detection by using multiple sources to perform the scan. That is rather than a single attacker performing the scan, the attacker launches the scan from multiple machines (different IPs) or multiple attackers collaborate to scan subranges of the target host or network.

2.4.4 Port Scan Detection Methods

The detection of port scanning has been tackled as a problem more network based than system based. Many methods are used to detect port scans; these vary according to the aspect of traffic used to indicate the scanning, in addition to the computing method that is most suitable to the solution model. Port scan detection methods [130], can be displayed briefly as follows:

a) *Algorithmic Approaches*: these solutions incorporate different statistical and probabilistic methods to evaluate the incoming traffic, where the use of multiple methods to form the solution algorithm is dependent on modelling of the system and tailored use of these methods. Different algorithms have been proposed to detect port scans that use techniques such as graph-based, probabilistic models, statistical analysis, network anomaly scoring, statistical profiling, heuristic approaches. The criterion of a scan detection varies from one solution to another, as each solution uses a specific mechanism to judge if there is a scan.

b) *Threshold-based Approaches*: these approaches normally examine the traffic and specifically observe traffic attribute values, the detection of a scanning activity is detected when these values exceed a certain threshold. Most solutions of this type use the statistical data analysis to compare with the predefined thresholds. Other approaches might use the validity of packets flag combination, a sequential hypothesis testing, customized traffic filters, and behaviour-based techniques. The criterion in all these solutions is the use of threshold to judge the existence of a scan.

c) *Soft Computing Approaches*: obviously, these approaches uses the fact that port detection does not need to be in definite numbers of ports to be considered as a scan. Therefore, these methods are used because of their ability to approximate reasoning and make decisions from partial truth. Although these algorithms offer reliable solutions, the need for training time and the effectiveness for a huge number of connections might affect their performance. Many soft computing methods have been used in port scan detection such as Artificial Neural Networks (ANN), Partheno Genetic Algorithms (PGA), Fuzzy Logic, and Bayes Kernel Estimators.

d) *Rule-based Approaches*: these approaches work through analysing the incoming network traffic to detect the scan in the traffic that has not complied with predefined rules. This implies building a knowledge base of the normal traffic patterns, which is a challenging task as network traffic is diverse in nature. Rule-based methods include solutions using packet headers information, fuzzy rules, and other signature based intrusion detection systems.

e) *Visual Approaches*: In these approaches, the traffic is represented visually and the scans are detected manually by the security administrators manually by observing anomaly in traffic graphs. Most of these solutions depend on plotting traffic data on packet level or flow level. Such solutions display graphically the relationships between ports and IPs numbers, connection activity, summaries of unique values per protocol, alerts three-dimension projection, traffic clustering, and data patterns visualization.

2.5 Vulnerability Assessments

Vulnerability is a defect in system design or a coding weakness in the developing, operation, controlling of information system or one of its modules that could lead to system breakdown or damage when the system experiences a threat or hazard, or affect the system's ability to recover to a stable work state [131]. This means that the vulnerabilities represent the spots that adversaries exploit to attack the system. Vulnerabilities exist in systems for different reasons such as; flaws in coding, faults in protocol designs, incompatibility among the system modules, etc,; vulnerabilities might arise when an intentionally created backdoor is exploited maliciously.

2.5.1 Vulnerability Types

As computer technology becomes more sophisticated and advanced, vulnerabilities being discovered are grow in their quantity and severity level. National Vulnerability Database (NVD) [132] published statistics related to vulnerabilities showing that the vulnerabilities' severity has fluctuated since 2001, however statistics displayed an incremental pattern of severity that reaches the peak in 2017 as shown in figure 3.13.

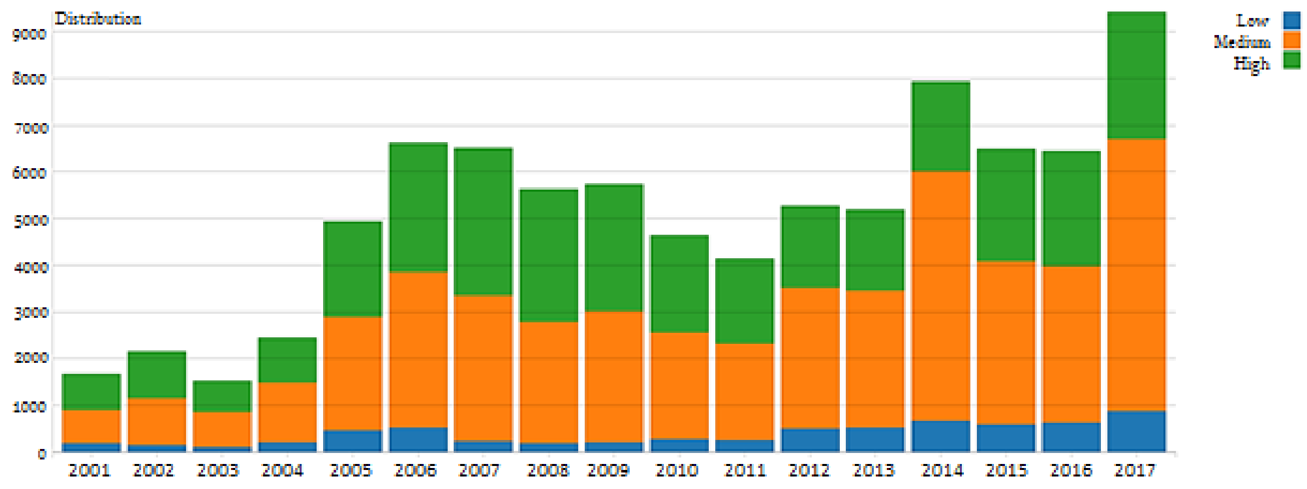


Figure 2.13 Vulnerability Severity Distribution [132] .

Vulnerability types' numbers vary according to what types are more prevalent, for example in 2007 the miscellaneous vulnerabilities were more than 3,500 and all other types were below 500. While in 2016 vulnerabilities related to miscellaneous vulnerabilities were below 500 and vulnerabilities such as input validation, permission-privilege-access control, and insufficient information exceeded 500 and buffer error vulnerability exceeded 1,000.

Network vulnerabilities is related to weakness that could be exploited remotely via the network of the target. Mainly these vulnerabilities originate from a bug in protocols used to connect the computers or a malfunction in network devices and configuration. Most network security problems stem from the fact that network protocols have been designed without considering the security aspect. There are many types of network vulnerabilities such as *Authentication Bypass By Spoofing* , *Stack-Based Buffer Overflows* , *Insufficient Entropy in code segments* , etc [133].

Operating System vulnerabilities are such type of weakness stems from a fault in the coding of OS software, or a procedural defect in the interaction among OS components that leads to the formation of a vulnerability. These vulnerabilities mostly affect the reliability and the confidentiality of the computer system. Although the majority of these vulnerabilities are not caused intentionally, they can still lead to a security breach that might disrupt the performance of the system or threaten the integrity of stored data. Several types of OS weaknesses are exist such as *Backdoors, Buffer Overflow, Installation vulnerabilities*, etc [134].

2.5.2 Vulnerability Scan Tools

It is a software application that searches for vulnerabilities in and maps the information systems such as in network, operating system, and software applications [135]. These scanners are able to identify the vulnerabilities caused by software bugs, incorrect system configurations, or misuse of a user. The main goal of using a vulnerability scanner is the detection of known security holes in the system. As vulnerabilities are discovered continuously, a regular vulnerability scan for the system will facilitate the remedy for these problems that could be used internally or externally. Scanners also map the system network that benefits in two points; first, it discovers devices or systems that connect to the system without permission. This helps to defend the system against intrusive devices or systems that might compromise the whole system security. Second, vulnerability scanners produce network mapping that identifies all network devices and their details such as device types, working OS version and update levels, and other information that is used in security management and monitoring. Despite the important role of vulnerability scanners in security assessment, however, it can only provide a description of the current system state. Therefore, a regular scan should be conducted for maintaining control over security vulnerabilities. Another shortcoming of the scanners is that they are only capable of reporting vulnerabilities according to the database of the scanner, and a human verdict is needed for decision-making, and to determine if the reported vulnerabilities are actually exist or the scanner failed to identify a present weakness.

Network-based scanners are used to discover the vulnerabilities that exist in the network devices, protocols, and administration risks such as Nessus and NSS. These scanners are deployed in single machines to scan other machines in the network, and normally deal with critical security weaknesses such as in firewalls configuration, web servers, network industrial software. There are many types of Network scanners such as Port scanners [136] Web server scanners [137] and Web applications vulnerability scanners [138].

Host-based scanners are designed to discover the vulnerabilities of the host operating system. Therefore, host-based scanners are allowed to test the low-level data such as processes and configurations of the OS. These scanners are able to present an insight of suspicious user behaviour; an indication of system compromising that appears in suspicious file names, unusual system files and privileged programs. Unlike network-based scanners, these types of scanner are able to perform system baseline tests. There are two types of host-based vulnerability scanners namely; Host vulnerability scanners [139] and Database scanners [138].

2.5.3 Vulnerability Assessments Processes

Vulnerability assessments is the process of reviewing the system components and processes in order to disclose weaknesses in the system. In broader definition it is the search for vulnerability in the system. Among the security tools, the vulnerability assessment is more focused on revealing the areas that are exposed to attacks. This means more dealing with the system infrastructure in terms of efficiency, effectiveness, and compliance in the shape of evaluating the level of security in the current system state situation.Vulnerability assessments comprise of four steps namely; vulnerabilities discovery, vulnerabilities identification, vulnerabilities analysis , and reporting. Next, we give an overview of these components as follows:

Vulnerabilities discovery: This process aims at identifying and revealing the number of vulnerabilities existing in the system. This process implies actors that participate in vulnerability discovery including a detector that finds the vulnerability in the system at the owner organization, software vendor detectors that are responsible for finding vulnerabilities as a maintenance or quality assurance, or public vulnerabilities disclosure sources such as Computer Emergency Response Team (CERT) announcements [140]. The research efforts to discover the vulnerabilities have depended on software reliability models, which focus on the dependence among fault existence, fault elimination, and work environment. Many models have been proposed in the literature such as *Anderson Thermodynamic Model (AT)* [141], *Alhazmi-Malaiya Logistic (AML)*[142], etc.

Vulnerabilities Classification : An important part of the vulnerabilities assessment is the process of identifying the vulnerability and categorizing it. Different methods used to classify and rank the vulnerabilities; the most common vulnerabilities taxonomy approaches are classification according to aspects such as cause, type, severity, impact. As it is diverse, some taxonomy tried to describe the vulnerabilities in such a way that is correct but unpractical. On the other hand, other types of taxonomy are flexible to the extent that a single vulnerability might be fitted in different categories [143].

Vulnerability scanning: This step can be defined as the attempt to find the vulnerabilities in the system software and network [144]. This means the actual process of finding the weakness of the system using software called scanners that search for the vulnerabilities in the system. Normally these scans are achieved by using a different computer to scan and check the targeted system [145]. Many software packages are proposed as a scanner such as Nessus [146], SAINT [147], Core Impact [148], AVDS [148], etc. to automate the process of vulnerability scanning, although the automated scan was not able to accurately identify all the vulnerabilities in the tested system, as well as the difference depending on the used operating system[149]. Vulnerability scanning is normally associated with penetration testing, which is actually complementing the work of scanning, although the vulnerability scanning is only focused on searching and identifying the vulnerabilities on the contrary Penetration test that focuses additionally on weaknesses exploitation and remediation. The vulnerability scanning is a knowledge-based process, where the search operation is performed against the content of the scanner vulnerabilities database. This explains the difference in results when using different scanners, as each scanner has its own database and updates [144].

Vulnerability analysis: this is not really an independent step of assessment, as analysis and assessment are frequently used to express the same process. Therefore; analysis encompasses searching, discovering, identifying, classifying the vulnerabilities and ending with reporting to relative persons. Some researchers state that the analysis is finding the relationships between vulnerability category and severity, and between vulnerability and its surrounding conditions [142]. This means, analysis focusing on understating the importance of vulnerability by identifying the severity, as more severe vulnerability leads to higher risk. Another aspect of analysis is to understand the increasing number of vulnerabilities aggregated over time, model the behaviour of these vulnerabilities, and test their impact on system security. Although it is very desirable for assessment tools to own powerful analysis capabilities, most of these tools have not achieved that yet [142].

Vulnerability reporting : this is the final step in the assessment process; a documented description describes the whole state of the system in detail and depicts the results of the assessment process [150]. The report should contain the findings of vulnerability discovery and identification such as vulnerability types, numbers, situation, etc. the importance of vulnerability reporting is shown in two significant points namely: improvement and prediction. Certainly, the direct purpose of vulnerability assessment handling in the current system security is for either securing existed vulnerabilities, or testing the ability of the system to be penetrated by adversaries [150]. Another benefit of reporting is with regard to system security in future, which is the predictive information about potential threats and expected number of potential vulnerabilities that might occur in the system depending on the discovery models.

2.5.4 Vulnerability Assessment Requirements

Vulnerability assessment tools share the similar conditions of work with other security systems. Here we will highlight some of most relevant requirements to these systems as follows [151]:

a) *Computational capabilities*: most tools imply minimum process power to find and analyse the vulnerabilities in the system. Although this condition is not that important nowadays with computers with powerful abilities, the system software has become more complicated and the number of vulnerabilities being discovered has increased on a regular basis.

b) *Reporting capability and Knowledge base*: all assessment software is based on the same concept, which is scanning for Vulnerabilities, however, it differs in how powerful these software packages express their findings, as the results of the scanning process depend on the quality of the vulnerability database used in the scan, and how informative, clear, and robust are the reports.

c) *Performance limitation*: Vulnerability assessment software normally produce a security snapshot of the system for a specific time point. As vulnerabilities are discovered on a frequent basis, scanning the system with every update of the vulnerabilities database might affect the system performance.

d) *Autonomous approaches*: Vulnerability assessments have mostly used the approaches and methods of penetration test in scanning the vulnerabilities, and as a part of the penetration process. While penetration test methods only deal with known and documented vulnerabilities, assessment software is focused on these types of vulnerabilities rather than finding unknown or inactive vulnerabilities.

2.6 Summary

This chapter presented and discussed a background information about the cloud computing and the security challenges and concerns about this environment. Such challenges vary from concerns that affect the performance of the cloud to attacks that impact the cloud severely such as stealing information, affect the availability, and violate the data integrity. The second section of this chapter has discussed the attacks that target cloud computing. The third section displays the intrusion detection and prevention systems that are proposed for cloud and non-cloud computing environments.

This chapter has also presented introductory background information that is necessary to understand the main concepts engaged to achieve this research work. This chapter gave a snapshot of the prediction methods in computer science and other areas, to depict the techniques used to obtain information about potential incidents that might occur in the future.

It also outlined the challenges facing the intrusion prediction in the research area. The main problems discussed in this chapter stemming from cloud computing is not an entirely novel scheme but a new paradigm for conventional computing technologies, which is adding new challenges to the inherited ones from the current technologies.

This chapter has presented a background information of port scanning definition, types and detection methods, which is the second essential part of the thesis. Firstly, begins with clarifying the importance of scanning activities for the success of an attack. This background highlighted two important points; the scanning activity step is precedes the harm steps of the attack, and the outcome of the scanning activity will define technique and the tools that are needed to launch a successful attack. Then the scanning methods were defined, and provided information about the Port Scan detection methods.

The last part of this chapter has provided an introductory information about the vulnerability assessments in the computer systems. This section starts with giving a definition of the vulnerability, then the types of the vulnerabilities in computer environment, the vulnerability assessments processes, and the vulnerability scanners, and finishes with the requirements that should be in any efficient vulnerability assessment system.

Chapter 3

Related work

3.1 Introduction

Prediction is used in many areas such as the stock market, weather forecasting, Health sector, and many others. Although plenty of research has been produced by academia, however, intrusion prediction systems are still not widely used in the cyber industry. In this section we will clarify the importance of intrusion prediction from related terms namely; detection and prevention. Intrusion detection is ‘the process of monitoring the events occurring in a computer system or network and analysing them for signs of intrusions, defined as attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer or network’[152]. Intrusion prevention is the process of detecting intrusions and trying to stop them [153]. Therefore, according to these definitions, prevention is dependent on detection and both are relying on monitored and identified security incidents, which implies that security incidents have already happened or are in the course of happening. It is important to note that detection and prevention systems need direct information to raise the alarm of security violation or prohibit known attacks. Furthermore, these systems failed to identify multiple step attacks, because the detection concept is to capture a single incident at a time not the whole series. In this context, the need for prediction systems appeared as a tool to support both detection and prevention systems. The basic idea behind the Prediction concept is the attempt to provide information on events that have not happened yet depending on historical information and gained knowledge of similar or identical events, which have happened in the past. Figure 3.1 shows the relationship among the three concepts.

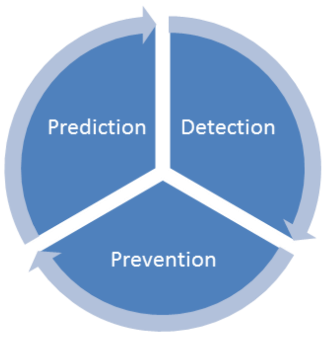


Figure 3.1 the Relationship among Prediction, Detection and Prevention.

Prediction as theory implies a lack of information about what might happen, therefore, security systems use the small amount of information that is available for processing and producing knowledge about potential future incidents. The more data that is available the more accurate prediction can be produced and the certainty of future incidents becomes more realistic. The prediction role stands where there is not enough information to consider incidents as an attack. This is because detection depends on solid information to capture security violations, this information might be signatures or noticeable anomalies in status of the system or network [153]. Therefore, the hierarchy of prediction, detection, and prevention will be in the form as shown in figure 3.2.

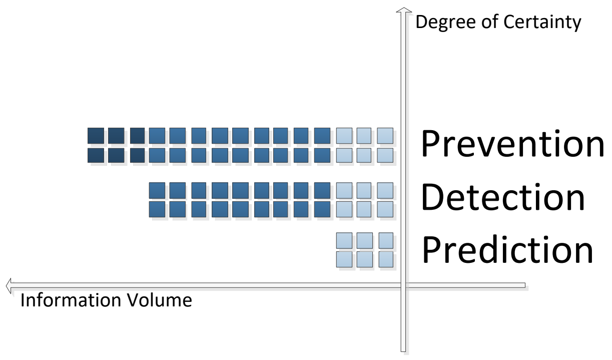


Figure 3.2 Information in Intrusion Related Systems.

Cyber-attacks are normally identified as single and multi-stage attacks, according to the number of steps that are required for achieving the attack successfully. In case of multi-stage intrusions, prediction is playing a significant role, as the probability of producing correct predictions is far higher than other attack vectors. This because multi-stage intrusions involve many steps and attacks in order to achieve the intrusion goal, so there are more likely chances to predict the ultimate attack by observing the initial steps or prior attacks before the attack achieves its goals and harms the system.

3.2 Intrusion Prediction Methodologies in Computer Security

Research into the prediction of cyber-attacks and intrusions area has been enriched by many solutions and models over the last few years [35], [154]–[167]. In this section, we will review some of such solutions to draw a picture of the possible methods, which could be exploited. Prediction systems use different methodologies to infer future potential threats. Methodology means the main concept of work that the system is built upon to achieve prediction. Methodologies are varying according to the type of data and source of data. Most prediction methodologies can be classified as follows:

3.2.1 Alerts Correlation

This methodology can be defined as 'interpretation of multiple alarms so as to increase the semantic information content associated with a reduced set of messages'[168]. The goal of alert correlation is identify the causal relationships between alerts [169]. Alert correlation is achieved in the context of grouping alerts from intrusion detection systems, processing them to put in unified form, and preforming correlations to find causal relationships between them. In the literature, alert correlation is used to build different types of models such as Alert Optimization [167], Attack Scenario [170] [159], Attack Strategies through finding connections between causes and consequences of attack [171], Attack Track using suffix tree [172], and Attack Plan [159].

3.2.2 Sequence of Actions

This methodology depends on detecting the abnormality in ordered sequence of elements. In such case, security incidents that are reported by IDSs or system calls that monitored by calls monitor are put into an ordered set to reflect the causal or temporal relationship among them. The serial sort of the set will make prediction relatively clear, as breaking the chain or the emergence of malicious acts in the series will be considered as intrusion. This concept is used in models such as System call sequences [158] [173], network packet sequences [174]. Although the alert correlation concept can be classified as a sequence of actions, however, this subsection is dedicated to the methods other than alert correlation.

3.2.3 Statistical Methods

The literature showed that statistical methods are widely used for prediction and forecasting in computing and other areas. Statistical analysis tries to investigate and present the collected data to reveal the underlying patterns and trends [175]. Although statistical analysis might not be suitable for all cyber related problems because of the linear nature of statistical analysis [176], however, these methods have been used effectively in many fields of cyber security such as in data mining [177]. The criteria for using statistical models are data nature, attack mechanism, and system architecture. Therefore, the literature showed the use of different statistical models such as time series analysis, linear regression, moving average, weighted moving average, exponential smoothing, reliability analysis, etc. [178] [179].

3.2.4 Probabilistic Methods

Normally this methodology is used when the way of processing the collected data does not lead to clear predictions, so probabilities are assigned and calculated for processing outputs. Another use of this method is to compound with other models in the same system in order to improve the prediction result. Literature showed that Hidden Markov model (HMM) and Bayesian network are the most probabilistic methods being used. HMM is 'a tool for representing probability distributions over sequences of observations' [180], and has been used in many solutions such as in [181] [167] [182],etc. Bayesian network is ' a graphical model for representing conditional independencies between a set of random variables'[180], and been used in many solutions such as in [155] [173] [183].

3.2.5 Feature Extraction

This method is used when the system focuses on a specific piece of information out of the gathered data. Feature Extraction is 'a set of techniques that transform and simplify data so as to make data mining tasks easier' [184]. Feature extraction can be decomposed into two parts; feature constructions that standardize, normalize, filter, and extract local features from data. The second part is feature selection that selects relevant and informative features [185]. Although this methodology is embedded in all data mining based solution, it has been used as a key methodology such as to build Cyber Attacker Model Profile (CAMP) [186], and real time intrusion prediction in [157].

3.3 Intrusion Prediction Systems

Cyber-attacks have the ability to disrupt or destroy computer systems and networks, or the information stored on them [187]. Using various degrees of sophistication, an attacker can access a system remotely by the Internet to gain unauthorised privileges or misuse its privileges. This is known as an intrusion and is the attempt to manipulate the confidentiality, integrity or availability, of security methods [188].

In recent years, cyber-attacks have increased rapidly in volume and diversity. In 2013 for example, over 552 million customers' identities and crucial information were revealed through data breaches world-wide [189]. This growing threat is further demonstrated in the 50,000 daily attacks on the London Stock Exchange [190]. It is predicted that the economic impact of cyber-attacks will cost the global economy $3 trillion on aggregate by 2020 [191]. These immense effects and implications have urged the United States Department of Defence to categorise cyber-attacks as an act of war, meriting physical military force in response [190]. Such a categorisation depicts the severe view countries around the globe now have of cyber-threats.

Two of the main methodologies applied in IDS include; signature-based and anomaly detection. Signature-detection functions by identifying intrusions through correlating real-time data with a known malicious behaviour database. The anomaly-based approach detects intrusions by comparing the current behaviour to a profiled normal behaviour [192].

In cloud computing for example; intrusion detection and prevention systems are used in different levels and technologies, where each of these technologies has its own limitations and issues [110]. Despite the advantages like the possibility to use resources remotely and huge volumes of storage, the growth in cloud computing has further exacerbated the threat posed by cyber-attacks, as critical infrastructures and digital service providers now have multiple access points to protect [193]. Hence, intrusion detection systems in a cloud environment are in need of significant research efforts to enhance performance and overcome the challenges and issues.

Regardless of the existence of advanced cyber-defence systems; attacks and intrusions still occur. Defence systems try to block previously known attacks, stop ongoing attacks, and detect occurred attacks, however, often the damage caused by an attack is catastrophic [187] [194]. Consequently, the need for improving intrusion detection systems is more urgent these days.

We argue that proposing a robust prediction system will help improve detection and prevention capabilities of defence systems.

The next subsections will demonstrate the research done to predict intrusions classified according to the method used as main concept of the work.

3.3.1 Hidden Markov Model

HMM is a probabilistic method that is widely used in many applications such as voice recognition and medical diagnostics. HMM offers a solid mathematical structure that could effectively shape a robust theoretical foundation for problem modelling. In HMM, it is easy to translate theoretical ideas to a mathematical model, which is an advantageous point over other modelling methods such as neural networks. This is reflected in practice through successful applications that use HHM [181].

In their research, Haslum et al., [195], propose a distributed intrusion prevention system. This system is composed of many IPSs spread over networks, which communicate through a central controller. The system uses an HMM to identify intrusion and prepare for the prediction process. In addition, a Fuzzy Inference System, for online risk assessment, is employed. HMM functions by passing a set of system states and probability indicators to the fuzzy inference system to assess the risk simultaneously. Whereas, a Fuzzy inference system analyses the risk using threat level, vulnerability, and asset values.

Continuing the investigation into prediction methodologies, specifically HMM, we identify that Sendi et al., [167] propose a framework for intrusion prediction, again using the HMM and alerts correlation to predict Distributed Denial of Service (DDoS) attacks. In their research, HMM is used to derive the interactions between the attacker and the network. The alerts correlation is modified as alert severity (using three parameters namely; frequency of alert, acceptable number of alerts per day, and alert effect), to generate prediction alarms of the effective steps of the attack, and to enhance the accuracy. The results showed an accurate prediction of DDoSs and a capability to detect other multi-step attacks.

Lai-Cheng proposed a solution that can predict intrusion based on the Markov Chain [182]. Their research claims to function with high-efficiency in real-time by using a load-balancing algorithm and statistical prediction model. The heavy network traffic is divided into smaller segments using a balance-paralleling architecture to overcome the problem of handling heavy traffic loads. When considering the prediction, a series of states of the traffic is checked if it occurs in the order of the Markov chain model, and the probability of this occurrence is computed using the following equation:

$$P_{i,j} = \frac{N_{i,j}}{N_i - \sum_{k=2}^{n} N_{i,k-1,k}} \tag{1}$$

Where $N_{i,k-1,k}$ is the element number of intersection of the state space, $N_{i,j}$ false negative analysis.

Zhengdao et.al., investigated host-based intrusion prediction system, in this case using the hidden semi-Markov model (HsMM) [158]. Firstly, in their research, they define the intrusion detection using the HsMM. Secondly, a case study using a prediction model using HsMM was put forward. This model depends on a differentiation between the normal system calls sequences from the intruder.

3.3.2 Bayesian Networks

Bayesian networks are powerful graphical methods for modelling and reasoning under uncertainty and complexity of information. It comprises of a Directed Acyclic Graph (DAG) enabling smooth representation of the domain knowledge as an influence network, and conditional probability table (CPT) that allows us to specify the uncertainty related to the relationships among domains variables [196].

In their research, Wu et al., proposed a model to predict Cyber-attacks using a Bayesian network [155]. Specifically, an attack graph is used for vulnerability representation, as well as for detecting possible attack paths. The main aim of their research is to consider environmental factors that influence the probability that an attack might be launched by calculating the vulnerabilities. The factors, which are considered in their research, include the value of the assets in the network, the attack history of the network and the usage condition of the network. Using the attack probability algorithms employed by a Bayesian network, Wu et al., claim to achieve accurate results, although no experiments or results are put forward in their paper.

Continuing the investigation into Bayesian Network for IDS, Feng et.al., propose a method to predict the abnormal events using the plan recognition approach [173]. This approach uses dynamic Bayesian network theory to predict the intrusion by monitoring the system call sequences. The goal of the system call sequences is to classify data into both normal and intrusion with a high level of accuracy. By using this classification technique, system calls are represented by the states of Bayesian network. The main challenge of their work is scalability; in a big data environment, the performance and efficiency is questionable.

Ishida et.al., propose an algorithm which is able to forecast the increment or decrement of attacks levels using Bayesian inference [183]. The idea behind this research is to predict whether the attacks will increase or decrease based on patterns of daily or weekly activities. Their research claims to achieve good prediction results, however, their approach is primitive and is unable to predict attacks' type or time.

3.3.3 Genetic Algorithms

Genetic algorithms are mathematical procedures used in artificial intelligence applications to learn the natural process of things. In this subsection, the focus of our investigation is on the use of genetic algorithms for enhancing intrusion prediction. The aim, again, is to identify techniques that can influence our methodology.

Sindhu et al., proposed a framework for intrusion prediction in networks using an Artificial Neural Network (ANN) and genetic algorithm combination [197]. A genetic algorithm is used to train the ANN, where the ANN is structured according to weight optimization. The results showed that this technique could speed up the learning process, and perform better than traditional back propagation networks in prediction accuracy. However, the proposed system functions by checking alarm rates rather than vulnerabilities, which is still a statistical prediction and not actual vulnerability testing. The framework itself is an intrusion detection system that uses the neural network to classify network traffic as normal or abnormal behaviour. This approach could automatically improve the detection ability. The neural network is trained using the KDD99 data set (which is a widely used dataset for experimental intrusion detection prepared by MIT Lincoln Labs [198]), and the features used in training were selected by the genetic algorithm. The genetic algorithm is used to select the most important features (from the KDD data set such as Duration, Protocol type, Src_byte, etc.) of the network traffic for detection and to adjust the neural network parameters. The results showed an effective outcome of detection.

3.3.4 Artificial Neural Networks

Artificial Neural Networks are defined as parallel and distributed processing systems constructed of a huge number of simple and enormously connected processors [199]. ANNs enhance intrusion prediction systems' ability to generalize data and classify it into normal or abnormal [110]. This means ANNs try to overcome the uncertainty of data by attempting to identify multiple shapes of data.

Zhang and Sun [174] propose a novel network based model to predict intrusions using a fuzzy neural network and coefficient correlation. In their research, the focus is on a back propagation algorithm in the fuzzy neural network, to train the network on KDD99 knowledge dataset. The key focus of their work is on a solution to predict the intrusion by analysing the network traffic information and forecasting the intrusion attempts. In their research, they claim to produce acceptable levels of prediction to detect different attacks such as DoS, probing, U2R and R2L. However, the network traffic features selection (such as duration, service, flag, src_byte, wrong_fragment, num_access_files, etc.) should be more specific, such that, the selected features might not be an indicator of an intrusion, and the prediction probability could be accounted for more specifically.

Continuing the case study, we focus on the work by Tang et al [200], who proposed a method for predicting network security situations based on based on back propagation neural with covariance. The aim this research is dependent on the concept of state sequences and identifying suitable values for quantified parameter self-learning adjustments. The prediction process was done by use of an historic and current values situation of the services and host fed to BP neural network.

3.3.5 Data Mining

Data mining is another approach to identifying data and relationships among them to produce information for prediction approach. Data mining is defined as a method for investigating data from multiple perspectives and summarizing it in useful information [201]. Li et al. [160], proposed an approach for intrusion prediction using attack graphs generated by data mining techniques. The data mining association rule generates several scenarios for the attack graphs, depending on multi-step attack patterns that are built using previous intrusion alerts. This algorithm computes the probability of attack incidence in each attack graph, which indicates all potential attacks. Ranking the attack scenarios according to predictability scores, correlating with real-time intrusion alerts can assess the future attack and predict intrusions.

Kim and Park [170] proposed a model to predict the network based advanced persistence attacks (APT). This model depends on extracting intrusion detection events, analysing the correlation among events, and then predicts the intrusion according to the context. First intrusion data is gathered and treated to extract information about attack threads and sessions. Subsequently, correlation analysis was done by creating sequential rules of detected intrusion events. The correlation is achieved by use of a continuous association-rule mining algorithm (CARMA). At this point, the researchers suggest two equations to predict the attempted time of intrusion and events occurring. By analysing and correlating intrusion detection events, their research uncovered an association between some specific attacks. Therefore, they could predict the ATPs according to the existence of some attacks. The results showed the possibility of predicting ATPs attacks depending on the intrusion detection events.

Cheng-Bin [157], proposed a method for intrusion prediction based on feature extraction. The researcher`s method depends on a proposed algorithm that first filters the relevant data, then a support vector machine (SVM) is used to classify the data into normal or abnormal. The researcher claims the ability of this method to predict intrusion online. The proposed method exploits a machine learning method that is a point of power. However, this method actually detects already existing attacks rather than forecasting potential attacks, in addition to its moderate accuracy.

Continuing the investigation, we identify that Onolaja et.al propose a conceptual framework for monitoring the trust dynamically and the prediction of behaviour of network nodes [161]. Their framework employs the paradigm of Dynamic Data-Driven Application Systems (DDDAS) and a trust-based model. The framework consists of two parts, a physical system representing the actual network and its nodes, and a controller representing a simulation of the entire network. The controller contains components, which collect data relating to nodes' behaviour within the network. In addition, it collects a trust value calculation and performs a data mining and prediction service. A trust value (TV) is attached to each node. Initially this value is neutral and it changed according to node behaviour. Nodes are distributed according to trust value to conceptualise three levels of trust: high-risk, medium-risk and low risk in the network. The prediction was achieved in the controller by comparing the actual behaviour of the node with previous behaviour stored in the controller, which changes the trust value and detects the misbehaviour.

Jayasinghe et.al [154] proposed an approach to predict drive-by download attack in web browser environment using a memory-friendly dynamic analysis. The approach depends on monitoring the bytecode stream produced by the web browser engine to render the web page. This is achieved by extracting features (Opcode return/getthisprop/not, setlocal/moreiter/ifne, forlocal/getarg/getlocal, etc) from the stream and then predicting the attack by a data-mining algorithm. The process of capturing and analysing the data stream depends on extracting the intermediate call trace by using Opcode function (a java built-in function) into n-grams keywords. These n-grams represent an individual feature and are fed into a data mining algorithm, which is a support vector machine (SVM), to predict the attacks. The prediction is performed by the SVM to detect new traces that do not exist in the vector space (hyper-plane). Results showed this approach's efficiency and effectiveness. However, this approach is not a universal solution since it is limited to a java library and is affected by the code complexity, in addition to training time used for training the classifier.

3.3.6 Algorithmic Methods

In this sub–section the focus is on the use of algorithmic methodologies and various research areas, which employ these techniques to enhance intrusion prediction systems methodologies.

For example, Kannadiga [163] proposed an event-based system for predicting network intrusion. This system depends on the idea that some attacks could lead to or be the start of a more severe intrusion attempt. The system they propose operates by distributing the attacks into categories, each representing a penetration level of the network. The proposed system collects information from databases about new and previous attack events, hardware, software events information, and other attack reports based on the external network. An intrusion prediction engine is employed to store information and predicts future attacks by mapping the attack events into relevant categories. A network penetration scenario from correlating attack categories is also built up. The prediction engine calculates the probability of future attacks that belonging to the next category of attacks

Another approach, which uses algorithmic methodologies, is proposed by Feng et.al. [173], who detail an approach to predict the abnormal events using the plan recognition technique. This approach uses dynamic Bayesian network theory to predict the intrusion by monitoring the system call sequences. System call sequences are classified into normal and intrusive, with a clear definition for both. According to this classification, system calls are represented by the states of the Bayesian network.

Pontes et.al [171] proposed a two stages system to forecast cyber-attack in computer systems. First stage comprises of an Event Analysis System (EAS) that performs a multi-correlation process by analysing and correlating alerts and logs of operating system and intrusion detection and prevention system`s logs. Researchers suggested a standard principle of causes and consequences within PC-correlation method, such principle based on the connections between the factors contributing to the attack (causes) and the effects of this attack. The second stage of the proposed system is the forecasting stage, where researchers employed the Exponential Weighted Moving Average (EWMA), which is a probabilistic technique to forecast the attack according to the output of stage1. Researchers present a definition for causes and consequences via the PC-correlation method.

Pontes and Guelfi proposed an architecture for forecasting intrusion using collaborative architecture [202]. Researchers did not mention any forecasting methods or techniques, but reference their other Portuguese written work. The basic idea of this research is to divide the process of analysis into four levels; every level is specialized in a specific part of the network; sensing, analysing, and forecasting all over the network and sharing the forecast results. Theoretically, collaborating among different resources is an effective principle to improve prediction and detection of intrusion. However, doing analysis and forecasting on multilevel is computationally infeasible. In addition to the ambiguity of the collaboration process, this solution depends on sensing multiple network variables that produce multiple formats of data in different system levels, which is an inefficient approach in terms of resources management.

Grunske and Joyce [203] proposed a risk-based approach to forecast attacks on components of the system. The idea of this method is to construct attack trees for every system module, with module vulnerabilities. These attack trees are used to measure the probability of successful system security breach. Within each attack tree, researchers defined constraints and metrics. The predefined metrics are; attacker motivation and ranks, attack risk and cost, and the information to calculate these metrics is collected from the proxy. This approach used the concept of attack profile, where every possible attack is predefined along with system status and environment factors that enable the attack. Prediction is done by comparing the calculated probability with the attack profile to decide whether the risk of an attack is at an acceptable level or considered intrusive.

Park et al [179] proposed a mechanism (FORE) for forecasting the cyber weather. FORE predicts worms attacks by analysing the randomness in incoming network traffic. The random connections are those connections that made to non-frequent or not normally used ports. The basic idea behind this mechanism is that the incremental nature of worm propagation produces more randomness in the network traffic, which is intrusive if exceeding a certain threshold. Researchers showed that this mechanism is faster than the previous method [204].

Fachkha et al [178] proposed a model for forecasting the short term future impact of the current distributed denial of service attack and its features. This model aimed to predict the intensity of the attack (number of packets) on rate (period in seconds) from what number of compromised machines (size of the attack). The basic idea is collecting backscattered data and session flows from darknet traffic [205], then applying DDoS detection parameters on collected data to anticipate the DDoS attacks. The anticipation result determined if the DDoS data is predictable and hence to apply the prediction methods.

Abdlhamed et al [206] proposed a system for intrusion prediction in cloud computing. Researchers employed the concept of using multiple sources of data to produce the prediction. They incorporated multiple powerful techniques that form an efficient system such as game theory concepts, behaviour profiling, risk assessment, and statistical methods. These concepts were integrated in component based framework system to perform prediction. Each concept represented one or more component in the framework for example components of data acquisition, game based behaviour builder, dynamic risk builder, historian.

3.4 Port Scan Detection

In [207] the researchers tried to propose the best method to detect the port scanning using a minimum amount of analyzed data. The research studied the use of IP header data only in the analysis process, and based their algorithm on using the network layer header data. As most port scan methods use the concept of fail connection to test if the port is open, where the connection has not been acknowledged by the initiator (the scanner), the research suggests network traffic will contain a high amount of fail connection data. The method proposed in this solution is a statistical method that focuses on detecting horizontal scans by observing the amount of fail connection attempts. The traffic monitoring is focused on packets or packet segments of specific size, although the researcher concluded that the size of packet or segment cannot be used as an indicator to detect port scans. However, they stated that there is a linear relation between number of ports and size of packets in traffic that contains a scan.

Jung et al [21] has developed a probabilistic solution based on the sequential probability ratio test called Threshold Random Walk algorithm (TRW) to detect a port scan activities local network. The algorithm basically depends on the history of connections to a destination according to the hypothesis that previously connected is more likely to be benign, and hence novice connection is more likely to be malicious. The algorithm starts by categorizing the source IPs in the traffic into two groups; malicious addresses, and unknown addresses. A further detailed classification is performed upon the data by splitting the malicious into Scanner, HTTP worms, and other. To identify the malicious IPs the TRW exploits an observation accompanied with Horizontal port scans that is the connection to a closed or idle port, which is an intuitive logic. The TRW examine each source probability using the sequential hypothesis testing proposed by Wald [208] in conjunction with the two thresholds; upper threshold and lower threshold. All these calculations will be updated for repeated source IP till the upper threshold, in the case of scanner, is exceeded. The researchers examined the performance of the algorithm against SNORT and BRO, and they were confident of the performance and the efficiency of TRW.

Yousra et al [209] proposed an algorithm to enhance the on-line port scan detection based on Bloom filters. This algorithm depends on a concept that a flow (stream from specific source) from an attacker will be abnormally larger than normal flows that have not exceeded the predefined threshold. The detection process is achieved in two steps; the counting step which uses bloom filter to analyse traffic, and the decision step that detects suspicious behaviour in the filtered data. These two steps are performed for a specific time window that, as well as the threshold, is set manually. Here the flow is considered a stream of data sent from the single source, therefore, the flow size is considered then the number of different destination ports contained in that flow. At the end of each time window the counter will reset itself to zero to prevent data overflow. The algorithm has been successfully tested against real traffic trace with no false positives. Because of the manually assigning time window length there is a tradeoff between the size of the window and the accuracy such that if the time window is set too long, it may give a lot of false positive outcomes because it combines small flows together. On the other hand, if it was set too short, it may not detect the attack because it divides the big flow into smaller flows by the time window's counter reset.

Xu et al [210] proposed a technique to remotely identify a local host in an internal network that is protected by a firewall and restricted to local access only. The primary objective of this research is to find machines that have hidden behind firewalls without causing an attack to the target. To do this, the researcher had developed a technique using side channel in zombie machine to learn information about the network. Because a Firewall normally blocks outside IP addresses from sending packets to the internal network, the idea is to send packets from the internal network into the same network to avoid firewalls. Therefore, they tried to use zombie in the same subnet where the hidden machine is located. As a principal requirement researchers had to study the way to avoid ingress filtering because it will block incoming packet if it has source address from inside the network. Researchers proposed a novel TCP/IP SYN backlog side channel technique called backlog scan. The SYN backlog is a buffer to hold uncompleted connections (destination replied with SYN-ACK but the source did not reply with ACK) till they are completed with the three way handshake. By using a third machine (zombie) the researchers measure the size of the blockage by sending multiple SYN packets. Then by sending different TCP segments they call it canaries and probes with spoofed SYN packets they could infer how many hosts are alive. The result of the scanning show that their own direct scan found more hidden machines than Nmap.

Matthew et al [211] studied various types of detection models to find the best model to detect the anomaly traffic that an attacker might produce. Packet Header Anomaly Detection (PHAD) has been proposed for identifying hostile network traffic by learning the normal ranges of value for each packet header field only, without the need for IP addresses and ports values. Researchers stated that intrusion detection systems suffered a shortcoming in their inability to discover the attacker intent. PHAD proposed to cover this point based on the hypothesis that the probability of event occurrence has a reverse relationship with the level of anomaly of the event. In other word, the lower the occurrence of an event the higher probability this event is anomalous, and vice versa. PHAD depends on calculations learned from training data to estimate the probabilities to detect the anomalies in online data. Because this solution depends on a learning phase, not only normal states of the packet header values should be learned, but also PHAD should have a-priori knowledge of potential attacks and a certain amount of a-priori knowledge of protocols. Researchers stated that this method performed better than the compared method in detecting four complex attacks, although they also mention that this method is not a standalone intrusion detection but rather is a support technique for existing IDS to enhance their performance.

In [212] the researchers aimed to detect port scan attempts and reveal important information about the scanner such as geographical location. Researchers stated that most IDSs focus on detecting port scanning only, therefore, took a step ahead and tried to identify the attempts of scan that would help to trace an attacker in future. Their experiment is divided into two parts: scan detection and information gathering. For scan detection, researchers searched the packets sniffing information for access patterns that match the stealth port scans in the incoming packets and classified them according to the threshold. The important information gathered from the captured packets such as the IP address of source, operating system, windows size, maximum fragment sizes, timestamp, etc. Other ways to gather information are finding probable locations from IP addresses from the public databases. The results from their experiment show success in the first step, which is scan detection at different time intervals. They can detect both brute force scanners and stealth scanners based on the types of flag of incoming packets. The result from another part of experiment gives the crucial information that can identify the operating system by using TTL parameter.

The researchers in[213] proposed a new method for detecting slow port scanning. Their method depends on analysing the traffic and selecting features of IP addresses that help to classify the IPs' three groups: normal IPs, suspicious IPs, and scanner IPs. Researchers monitored the traffic on short time windows, so the collected data amount will not affect the performance of the system. For classification, they depended on predefined thresholds rather than machine learning algorithms to evade the training time needed and choosing a representative training dataset. The researcher showed the correctness of their method in scanning detection heuristically.

In [214], the proposed scanning detection method depends on exploiting the concept of the dynamic bit sharing. The later concept is a novel storage optimization method that combine probabilistic sampling with bit-sharing by evenly divide the available bits among the sources randomly such that the memory space is optimally used. The main objective of proposing this solution is to increase the speed of a scan detection that consume small memory in static RAM or SRAM through dynamic bit sharing, and to ensure the false positives/negatives ratios are bounded. Their problem statement depending on report any sources whose spread is larger or smaller than threshold by using probabilistic performance that they defined. Then, they presented the new method called Efficient Scan Detection (ESD) to meet the objectives above. ESD is the combination of probabilistic sampling that stores the incoming contacts, and bit-sharing storage that estimates the number of contacts depending on the counts of remaining zeros. To find scanners in this method, it states that if an estimated value of the true spread of source exceeded a threshold value, then ESD identify that source to as a scanner. The results from their experiments compared to other existing works states that ESD performs less than 1 bit for each source, while other solutions require much more than that. Moreover, in terms of memory requirement, ESD does consume a small area in memory since it does not store the source and destination addresses in the contacts. In terms of false positives/negatives ratio, there is only their method that can provide low rates of both FNR and FPR.

Gates et al [215] proposed an novel method for scan detection for very large networks that only provide information on unidirectional flows using logistic regression modelling. Researchers focused on such networks such as internet service providers (IPS) where it is difficult to comprehend of all their characteristics, or packet level information is hard to provide due to the huge amount transmitted. Therefore, the researcher found logistic regression suitable for such a setup because the model will return a probability of a scan that security administration can select limits for defining the scan rather absolute decision, and uses multiple sources of knowledge such as expert point of view and monitored data. The model depends on analytical variables calculated from the flow information that fed into an equation that produces major value that fed into the logistic regression, where variables are weighted according to expert opinion. This method has been evaluated against adjusted threshold random walk (TRW) [21], they stated that TRW outperformed the proposed method, however, the results were comparable they stated.

Mai et al [216], studied the effect of packet sampling on the performance of port scans detection approaches, which data sampling uses as a network monitoring optimization technique. The researchers clearly address the question to achieve their objectives of the research that "*Does packet sampling distort or lose pertinent information from the original traffic profile that affects the effectiveness of existing anomaly detection techniques? If so, by how much?*", since it is unclear that how packet sampling impacts anomaly detection. They use three well-known algorithms which are threshold random walk (TRW) [21], time access pattern scheme (TAPS) [217], and entropy-based method [218] to evaluate the impact of sampling. The evaluations conducted by this study conclude that packet sampling has negative effects in both changing traffic and the performance of detection algorithms. Researchers observe that packet-sampling impacts the inference from single-packet flows and the invariance in address range distribution, therefore, they proposed a hybrid method based on TAPS to detect port scanning called TAPS-SYN. The proposed method depends on the hypotheses being tested on network flows that consist of single SYN-packet during specific time windows. Researchers stated that TAPS-SYN evaluation results showed the lower false positive rates than original methods and preserving sufficiently high success rate for detection in sampled data.

Joanne Treurniet [219] proposed a port scan detection method depending on the use of packet level information to detect arbitrarily slow port scans, that is hard-to-detect scans. The method does not need prior knowledge or training data to operate. The method comprises of two steps; 1) a connection identification called “session” using a finite state machine that models the TCP connection, 2) activity pattern that is a classification of network activities into four categories: productive, scanning, unproductive, and ambiguous. This method is an extended work proposed previously by adding UDP and ICMP in addition to the original solution for TCP protocol. The results showed that the system was able to classify the connections in the traffic successfully according to suggested patterns at a ratio of 98.6% of the whole traffic. Surprisingly, a minimum of 78.0% of traffic came from reconnaissance activities, and most of these activities, about 80.9%, were slow scans. Results also showed that this method is capable of detecting distributed scans but works better with single source scans. The researcher assumed that most of the reconnaissance activities caused by worms spread, as it normally sends huge number of probes to random IPs and ports at random periods.

In the work which appeared in [23] a probabilistic solution has been proposed to detect network scans in real time. In this research the detection depends not only on TCP sockets, but also whether the sources of connections are frequently connected to the network or not. The method starts with classifying the access patterns of remote sources to the targeted destinations; this process is called access indexing. Then, the sources rating process is initiated based on two assumptions, the first assumption is: it is unusual for a source to connect to a destination that has rarely been accessed. By monitoring access patterns, a prior probability for a given destination to be accessed randomly by a source, can be estimated. The second assumption: how suspicious a source is, is defined by the number of destinations and ports that are communicated by it. A consequence of this research suggests that a source that contacts few destinations is more likely to be normal, and the source that contacts a large number of destinations is more likely to be an attacker. The researchers have demonstrated the effectiveness of the solution based on qualitative analysis, also they state that the approach obtained feasible false positive rates and throughput.

Sridharan et al [217] suggested a new method for scan detection called Time-based Access Pattern Sequential hypothesis testing (TAPS). This algorithm combines Sequential Hypothesis Testing technique with access patterns to detect sources that showed malicious access to destination. The algorithm studies the access behaviour for an IP address through a time slice and focuses on the proportion of unique destinations and ports accessed. Then it performs the sequential hypothesis test on the collected access patterns during different periods to detect the port scanning. Researchers stated that TAPS achieved maximum detection rates and minimum false positive rates compared to SNORT [129] and TRW [21].

In [220] the researchers use the concept of anomaly detection in port scan detection. Their main principle of detection is to use the statistical techniques to calculate the changing point of normal traffic into an anomalous. For volumetric scans such as SYN flood researchers used some statistical tests such as the Newton method and z-test for traffic analysis, while for slow port scans, they used two dynamic chi-square tests. The results have been demonstrated as numerical results without commenting on the method performance.

3.5 Vulnerability Assessment

Cyber-attacks are mostly based on exploiting the weaknesses in computing systems and as those attacks become more sophisticated continuously, the need for assessing these weaknesses becomes more urgent. The Committee on National Security Systems (CNSS) defined the vulnerability assessment as “Systematic examination of an information system or product to determine the adequacy of security measures, identify security deficiencies, provide data from which to predict the effectiveness of proposed security measures, and confirm the adequacy of such measures after implementation” [221].

A number of frameworks have been proposed in the literature to tackle the emerging field of Vulnerability assessment, and new methods will be proposed in order to improve the understanding for the security situation and hence improve the system security. Most of the methods applied in vulnerability assessment are inspired by the related field of risk analysis such as Fault Tree Analysis (FTA), and Failure Mode & Effects Analysis (FMEA) [222]. In this section, we investigate the work accomplished for assessing vulnerability in computer systems as follows:

3.5.1 Holistic Frameworks

In this type of solution, there is a comprehensive framework for assessing the security of the system including the vulnerability. Such solutions are normally proposed and developed by an expert team belonging to governmental agencies. For example, System Security Engineering (SSE) that is a specialised engineering branch of systems engineering applies the methods, concepts and measurements of mathematics and engineering in producing an integrated evaluation of system vulnerabilities [223]. This framework is employed by major government agencies to protect the systems, mission-critical jobs, and improve the cyber technologies. SSE has the ability to identify sophisticated cyber threats, a variety of critical security risks, and presents a significant taxonomy of threats.

3.5.2 Information System-focused Frameworks

These frameworks are proposed to be more focused on the information systems and the core critical functions and processes, which differ from the previous frameworks that focus more on the system as a whole and on system interaction with the environment. For example, Vulnerability Assessment & Mitigation (VAM) provides an understanding of the interactions among the system functions and objectives, simplifies the identification of vulnerabilities, and suggests appropriate mitigation practices. VAM classifies the system weaknesses based on four system objects categories; cyber, physical, human/social, and infrastructure.

3.5.3 Cybersecurity-focused Frameworks

The Internet is becoming more integrated into the daily life of humans on governmental, organizational, and personal domains, therefore, the exposure of information systems used in these domains to the cyber world has increased continuously. Consequently, the more exposure to the cyber world the wider threat space for the system. Therefore, a vulnerability assessment methodology that is dedicated to measuring how vulnerable a system is to the cyber work, is a security requirement these days. The MITRE Corporation has proposed an approach to assess vulnerabilities in the cyber-security of information systems [32] that is called the mission assurance engineering (MAE) approach to address the cyber challenges. Cyber MAE is a risk-based system that concentrates on risks to system missions that originate from cyber resources, and threat-informed that collaborates and integrates threat information from different sources.

3.5.4 Service-oriented Architecture-focused Frameworks

In service-oriented architecture (SOA), the business processes are designed to provide a flexibility in composition and interoperability among remote and local services. Consequently, vulnerability discovery and assessment become more difficult and urgent. ATLIST [224] was proposed as an answer to the urgency of a vulnerability assessment methodology that is specialized to work in service-oriented architecture. ATLIST was designed to exploit the SOA standards and essential concepts such as reusability and flexibility. As such, this method enables the detection of known vulnerabilities and facilitates the construction of vulnerability patterns for application support.

3.6 Discussions

As stated earlier, modern system environments needs to be equipped in advance for attacks and intrusions, which have increased and become more sophisticated in techniques. Intrusion prediction systems should be the solid base that any proactive defence plan can rely on. This is because of the promising performance that prediction systems achieved on both directions; the number of attacks that have been predicted successfully and the false alarm rate that has been decreased [196], [197], [167].

Variety of methods are used to implement the prediction process, as seen in the literature, and this because of many factors such as which attributes and variables of the system used to make the prediction. Another factor is the degree of certainty, where a high degree of certainty means that there is a lot of information available and a low degree of certainty means there is a lack of information. In the case of a low degree of certainty, we notice that prediction systems depend on Hidden Markov Models, while with a high degree of certainty they tend to use Bayesian Networks. Furthermore, the type of prediction is an important factor in whether to use one particular method rather than others. For example, solutions dedicated to predict attacks normally use HHMs, while those devoted to forecasting intentions or abnormal events mostly exploit the Bayesian networks. Other factors might be the simplicity of the theoretical system, and HHMs' ease in translating the well-planned system into a mathematical model.

Performance is one important factor, where any defence system (including prediction) should not decrease the performance of the protected system and the usage of its resources. There is an efficiency trade-off dilemma; in other words, the main goal of prediction system is be as efficient as possible, whilst preserving the high level of performance. To achieve an efficient solution, a complex prediction method should be used along with analysing historical data and extensive calculations to produce optimal predictions, which decreases the system performance. Vice versa, to achieve high performance, the prediction solution tend to use inaccurate methods that affect the prediction efficiency. Therefore, preserve the availability and performance of the protected system is a condition when choosing the optimal prediction method that suite the target system.

To enhance the predictability of prediction systems, these systems should be able to recognize attacks in their multiple shapes and variations. At the same time, these systems should have the ability to classify the data into normal and abnormal (intrusive). This takes the research interest into a different area of computer and mathematical techniques and methods such as data mining and soft computing techniques.

Prediction philosophy is based on work in the situation of uncertainty of intrusion. This means there is a lack of information that prediction systems can use and rely on. The basic idea behind the prediction concept is the attempt to provide information on events that have not happened yet, depending on historical information and gained knowledge of the same or events which have happened in the past.

Artificial Neural Networks are used because of the various benefits for the prediction process. ANNs have the ability to generalize data and identify multiple shapes of data, their ability to produce more than one single output; which gives the prediction system a wider space of choices, and finally the ability of ANNs to train and improve its performance over time. Data mining contributes to the prediction systems by its ability to summarize data and find relations among data to produce new information that bridges the gap of information deficiency.

There are many contributions in the field of intrusion prediction and detection systems, whether it is on the methodology or technique levels. However, the literature showed very few solid solutions for a holistic intrusion detection system and hence prediction system. This is because of the limitations and shortcomings of each model; neural networks suffer from the time required for training, data mining and the issue of computational complexity, and so forth. We discussed intrusion detection systems because the vast majority of proposed prediction systems are actually dependent on IDSs. This is why any improvements achieved on IDSs are reflected consequently on the prediction system.

Detection systems also pose problems such as false alarm, which researchers are working on to decrease to the minimum. Anomaly Intrusion detection systems adopt the threshold technique as the unique mechanism when it comes to making decisions, which means that these systems are static, since they rely on a fixed threshold. The process of defining a threshold is ambiguous and most researchers depend on experts to define and measure the threshold. On the other hand, signature based detection systems are not capable of detecting new or unknown attacks, since attacks should be discovered and known, then a signature of the attack details should then be made to prevent future attacks of that signature. In addition, all scan detection systems and methods work on the principle of twofold solution, one part to identify, index and classify the traffic packets, and the other is to quantify the likelihood or give a judgement of what was identified in the first part.

As a result of these discussions, we reach to a comprehension that most suitable the research approach of this project is to use a statistical method for intrusion prediction along with predictive analysis. In this approach, efficient predictions will be made while preserving the performance as an outcome of using lightweight statistical method. This will discussed in detailed in the design chapter of this thesis.

3.7 Summary

This chapter presented the related work proposed to predict the intrusions that targets the computing systems. It started with an introduction that showed the importance of intrusion predictions and justified their usage along with detection systems to secure the computer systems. Then, the literature of intrusion prediction solutions that proposed for securing computer systems has been demonstrated and discussed thoroughly. This part of the chapter also presented a brief classification of the methodologies that has been used in intrusion prediction solutions.

This chapter also presented a detailed demonstration about port scan detection methods that are used primarily for identifying a remote scanner. The last part of this chapter has discussed the vulnerability assessments methods and frameworks that were proposed for identifying the weaknesses of computer systems.

Chapter 4

Intrusion Prediction Framework for Cloud Computing (IPFCC)

4.1 Introduction

Cloud computing poses many challenges to existing security techniques and solutions, which were adopted from the traditional computing solutions. Currently, there is a need for prediction systems to work with massive data volumes and high-speed networks.

Previous chapters clarified that there is a need for new and capable prediction solutions to cope with challenging structure, uncertainty, and complexity of cyber-based systems. The literature review has shown the difficulty of working of an entirely suitable solution for such a challenging environment, and it reviewed the shortcomings of many existing popular techniques used in similar environments. Such problems like the dynamic nature of cloud computing of adding, removing, changing virtual machines that differs from the classical defence systems that normally designed to protect static systems with specific configuration. The structural concepts of cloud computing such as virtualization and multitenancy contributed into the failure of classical defence systems to cope with the cloud environment. Therefore, a novel approach is required to predict potential intrusions and attacks in advance by proposing a new feasible approach to sense the pre-intrusion situation, and analyse system exposure to external attacks.

This chapter proposes a novel framework for intrusion prediction. The framework proposed aims at overcoming the limitations of existing solutions for intrusion prediction in cyber-related systems and provide an efficient and effective prediction system.

4.2 IPFCC Framework

In order to bridge the gap of lack of intrusion prediction systems, a novel solution has been devised in [206]. The framework is a statistical-probabilistic solution that has been specifically developed to monitor system vulnerability and the effect of the network situation on the security of the systems. It focuses primarily on protection against intrusions and attacks that uses the networks as a medium.

The framework proposed in [206] is broader than IPFCC, which incorporates behavioural monitoring and game theory concepts along with historical information analysis to produce the prediction. IPFCC focuses more on producing prediction and minimizing the dependency on historical information to an amount that does not slow the performance of the framework nor delay the decision making of the tested state of the system.

The idea of incorporating a scan detection functionality into IPFCC derived from the role of scanning activities in network based attacks, where scanning represents the initial step of the reconnaissance and information gathering. Scanning techniques in essence are not intended to be attacks as they were originally used by network and security engineers, to map and maintain the network, and included hosts and services. However, the capabilities they offer attract the attacker to use them as scout tools that identify targeted hosts and map networks for launching attacks. Therefore, not all scanning activities found in the network traffic are hostile, but a possibility that scanning activities are intrusive also exists. Moreover, some research indicates that a port scan followed by a vulnerability scan is a good indicator for preparation of an attack to be launched [25], which means detecting scan activities can be used effectively in a prediction system. The use of scan detection for predictive purposes stems from the time gap separating each stage of multistep attacks (which is the nature of network-based attacks).

IPFCC is an autonomous framework that is bases and operates on the host system, but also utilize a collaborative feature by using information about vulnerabilities from trusted sources in the outside world. The automaticity of IPFCC is an important aspect to sustain system independence and security by not disclosing system vulnerabilities. This will eliminate the fear of the extent of trust of a third party control or knowledge of shared information.

IPFCC uses a statistical approach to calculate the score of the vulnerabilities of the host system, utilising data of different levels of host system and outsourced data. The framework is located on the top of the host operating system, yet IPFCC is weaponised with the ability to obtain data from different OS levels, as showed in figure 4.1.

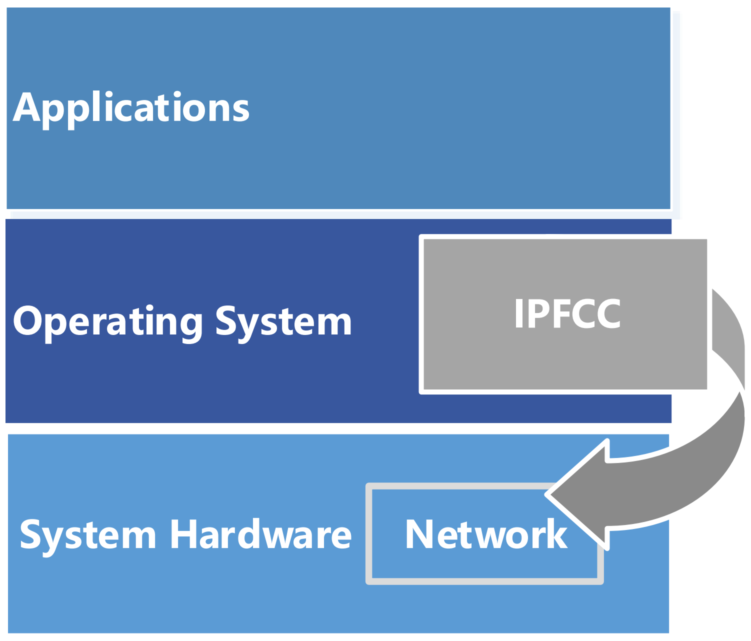


Figure 4.1 IPFCC Location in the Computational Environment.

The IPFCC framework operates by evaluating the situation of the system exposure, and monitoring live traffic to observe the changes and deviations in the incoming network streams. It monitors various OS-level and network-level metrics and makes the prediction of potential intrusions accordingly. These metrics and the mechanisms of both IPFCC`s parts will be discussed in details consecutively.

4.3 IPFCC Framework Design Overview

This section presents an overview of the architecture and information flow among the framework components. Here, this section only demonstrates the interaction between components, the components design and contents will be discussed in detail later on in this chapter. The figure 4.2 demonstrates the components of IPFCC:

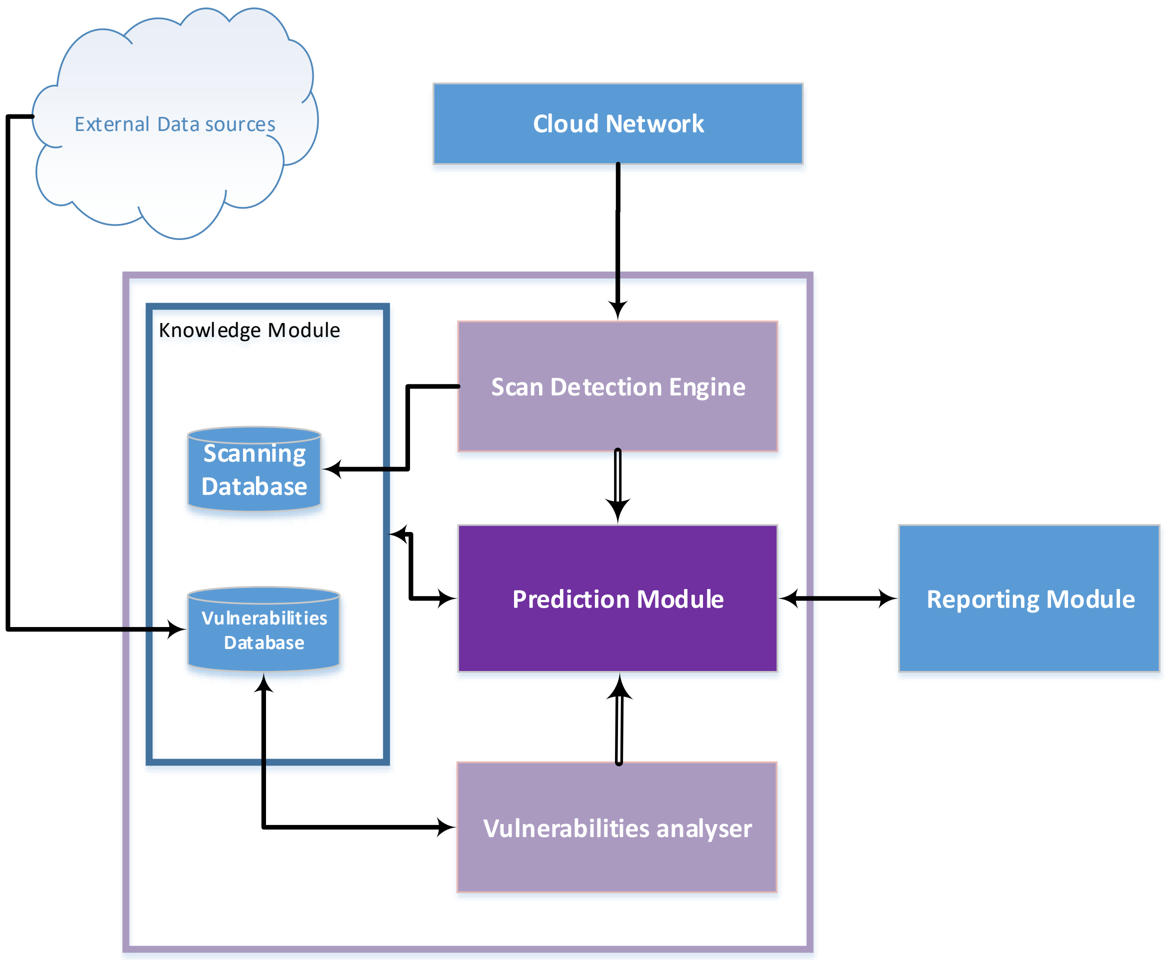


Figure 4.2 IPFCC Framework Architecture

Initially, the information about all vulnerabilities are collected from the system by scanning and from the public databases on the web, then reside in the Knowledge Module databases. Then, this information proceeded to the vulnerabilities analyser that evaluates the information of the weaknesses in the system such as severity and exploitability. The analyser interact in regards to the output in two ways; the first one is with the Prediction module to calculate the risk and make predictions, and the second one is with the knowledge module to save vulnerabilities information for passive prediction mode and for achieving. Scan detection module is responsible of sensing the network for scanning activities that is forwarded directly to the Prediction Module to act accordingly. The second interaction of the Scan Detection Module is with the Knowledge Module to store the scanning activities that launched against the system. The Prediction Module interacts primarily with scan detection and vulnerabilities analysis modules to produce predictions, then output the prediction results as reports and alarms. The other interaction of the Prediction Module is with Knowledge Module for storing prediction results and risk quantifications for archiving. The scan detection engine interact with the network by monitoring the traffic, and interact with Prediction Module by forwarding the monitoring outcomes. The intrusion prediction process preformed in the following flow:

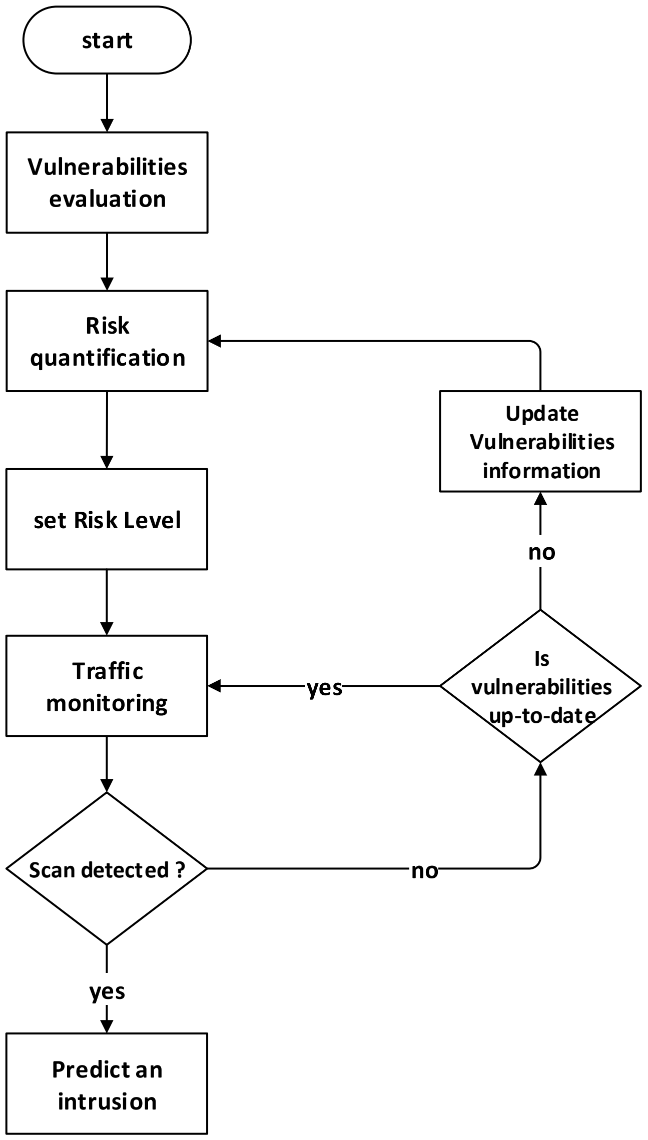


Figure 4.3 Intrusion Prediction Process.

4.4 Scan Detection Engine

This is a dynamic and active part of our solution, which senses the anomaly in the network traffic, and informs the prediction of the results of the network situation. As network traffic contains different types of anomaly, we focus on anomalies that could be used in a predictive manner. Therefore, this module is suggested to be a selective scan detection that monitors the surveillance traffic. The latter type of traffic normally precedes any attack that is launched via the network; therefore, our detection method focused on sensing the scanning traffic. By relating to traffic, this engine is working directly on the network to calculate and analyse the traffic statistically, then send the results of this analysis to the prediction module for decision-making.

The scan detection engine is based on a selective anomaly detection algorithm that uses a counting method to calculate the number of certain packets that relate to scan activities (as discussed in details in 4.4.1). The counting mechanism is applied on the whole traffic (for time window) based on the source of these packets, so the output will result in the number of specific packets sent by each source. After counting the number of packets sent by each source, we then test these numbers to isolate the odd number of packets that is potentially sent by a malicious source. To this point, we will have two sets of sources; normal set and odd set. We calculate the threshold of the normal set and compare the odd set to the threshold then decide if the odd is a scanner or not.

In order to achieve the first part of the algorithm, we count the number of surveillance packets sent from each source (remote hosts). Because the classic counting method does not fit for cloud and modern network traffic that transmit huge amounts of data, we use the data streaming methods for accounting. In data streaming methods, approximations methods are used rather than counting exactly due to performance and memory implications, as approximation methods consume considerably less memory than counting methods. Sketches is a recent technique for unique values estimation for streaming data. Sketches summarises the observed data in order to estimate the frequency or the distinct values in the data set [225]. Therefore, concerning scan detection, the algorithm will count the unique ports accessed (scanned) on a host or the IPs contacted on a specific domain.

After counting the streams flowing into the system, we differentiate between normal and abnormal streams by the amount each stream carries. This implies choosing a mechanism that differentiates the abnormal observations and at the same time preserves the statistical fashion, the way we tackle the whole detection process. We chose to perform the outliers` test (Grubbs` test [226]) that will isolate the streams with abnormally volumetric streams. The output of this process is two sets; the normal set that contains the streams that lie in normal use frequencies, and the other set that contains streams that lie an abnormal distance from the normal set. We then calculate the standard deviation of the normal set and then calculate the detection threshold from the standard deviation.

Here, we would like to refer to the methods of choosing the threshold, as the literature showed that the majority of thresholding is fixed and defined before starting the detection work, which in our opinion, is the reason behind the false positives. On the other hand, setting a threshold for network traffic behaviour is not an easy task as in modern networks, defining normal use behaviours is challenging task, as the modern networks are versatile and interconnected with other broad networks. For example, a predefined malicious pattern of access (hence network traffic) could be a benign access in special events or specific access time such as in peak time. We adaptively select the detection threshold by examining the normal set that is produced after eliminating the outliers, and compute the standard deviation for that specific subset. Then we compute the distance to a central point that represents the centre of the normal set, which will be a reference point. The decision of detection is made if the distance between an outlier and the central point exceeds the threshold.

4.4.1. Selective Dynamic Scan Detection (SASD)

Launching attacks for any reasons is a staged process comprising of a sequence of different steps. The number and type of these steps depend on the type of attack and the intended purpose that motivates launching it. However, surveillance activities are involved in all attacks as a crucial initial step of identifying victims, victims' open ports and vulnerabilities. Based on surveillance outcomes, the attacker can establish an appropriate strategy for attacking the potential target. Therefore, detecting surveillance activities is a crucial step in predicting a forthcoming attack. As surveillance is essentially comprised of scanning activities, this chapter is dedicated to explain our solution for scan detection.

In this section, we give an overview of our detection solution that is meant to work on the system-network interface or network nodes to sense and detect the scanning activities. We detect the scanning activities by counting the increment in specific traffic attributes, then check if this increment is anomalous or not by using the resulting counts. The figure 4.4 shows our proposed solution:

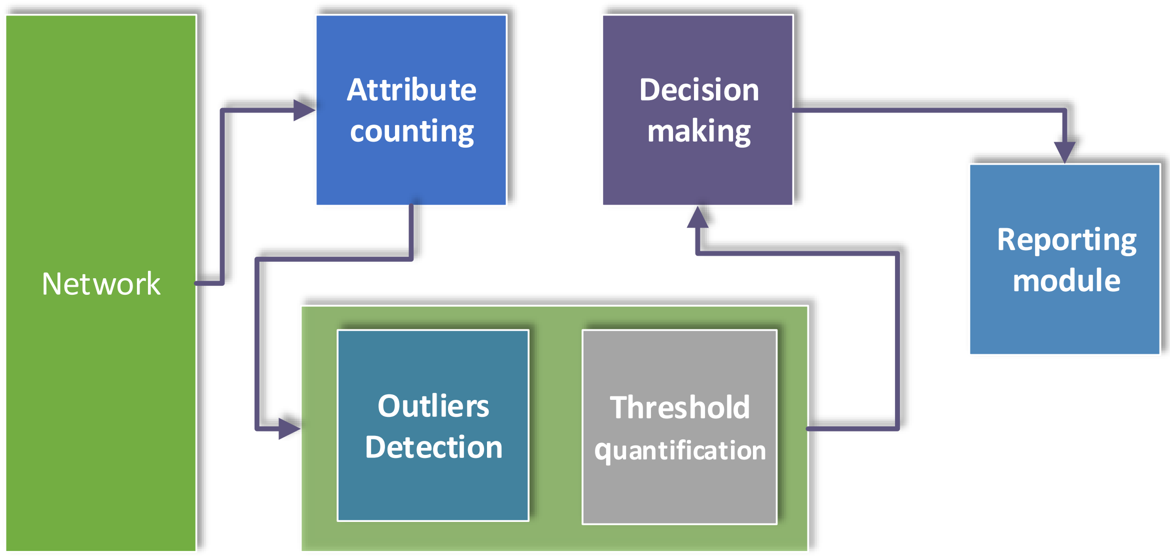


Figure 4.4 Overview of Scan Detection Solution.

The counting module reads the traffic from the network and focuses on counting specific attributes in the network stream that reveals the scanning activities. Then the results of counting are produced to identify anomalies to check if there is any values that spikes among the set values, which are called outliers. Then, the distance between the outliers and the rest of the values is calculated. A scan will be detected if the distance exceeded a threshold. Figure 4.5 shows the algorithm steps as follows:

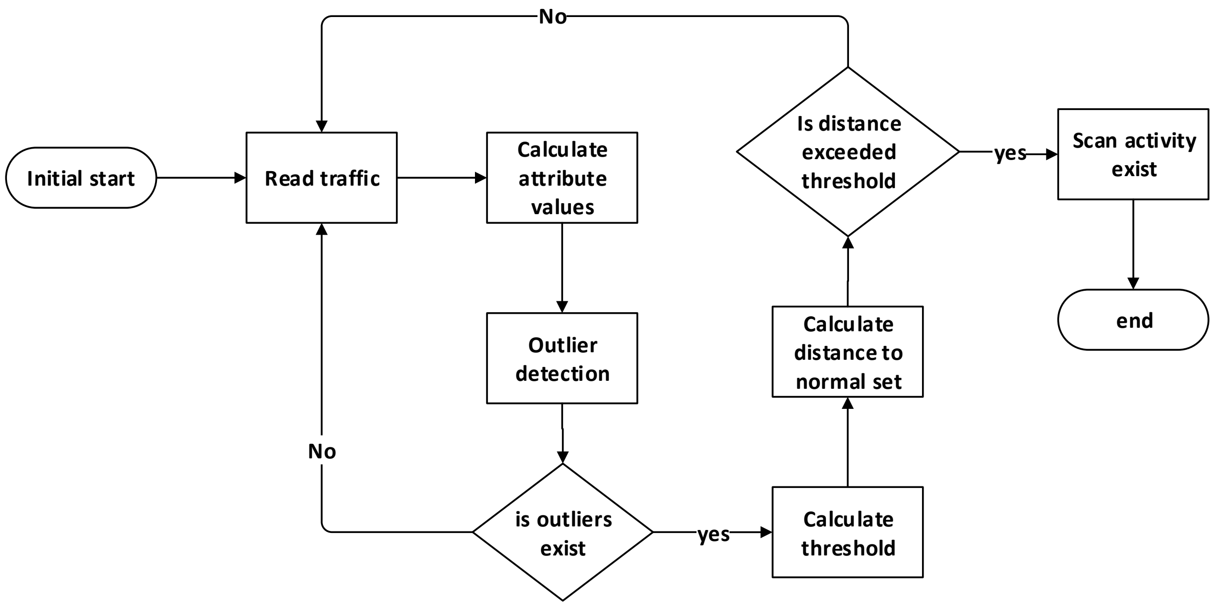


Figure 4.5 the Proposed Method for Scan Detection.

The algorithm works in the fashion of analysing the network traffic based on a single time window of analysis and calculations. This means that the results and detection of the scan is achieved for every given subset of traffic. By working in this style, our solution eliminates the dependence on historical data that might be problematic in some cases, and is adaptive to the network situation. In addition, our solution does not need training data or training phase, as it works directly on the network traffic and produces results simultaneously. On the other hand, the length of the time window may affect the accuracy of our solution; therefore, the length of the time window should be tuned to cover scanning activities periods. In each single window, we do the following:

4.4.2 Traffic Analysis and Scanning Models

Analysing all the data streaming ingress and egress of the targeted network is a highly complicated and expensive process, especially nowadays with the huge amount of data transmitted every day with the outside world. Investigating this data in favour of security might be conducted in reliance on different types of indicators such as data content, network information, domain and URL, etc., that relies on different levels of viability and reliability [227]. Network information that resides in IP packets headers provides valuable statistical information that could be used to gauge the network current situation, and predict the malicious intentions and future actions. Monitoring all variants and abundant information contained in the IP header might distort the monitoring outcomes and confuse the security administration. Therefore, we will observe some attributes in the traffic packets that are related to scanning activities. We are focusing on analysing the access attempts that are similar to the surveillance activity that aims at network and host mapping, which are actually scan attacks.

We consider network scanning as many access attempts from a source s_i to host h_i on the same network *n* as $\langle s_i , n,\ h^* \rangle$. Usually the attacker sends packets of the same type to all hosts in the network and waits for responses to check which host is alive. The host scan is an access attempt from s_i to multiple ports p_i on the same host h_i such as$\langle s_i , h_i, p^* \rangle$. Usually this is the next step after the network scan and then the attacker sends an access attempt to all/many ports on the same host. These two steps comprise the surveillance activity that the attacker takes to gain information about the targeted network and machine.

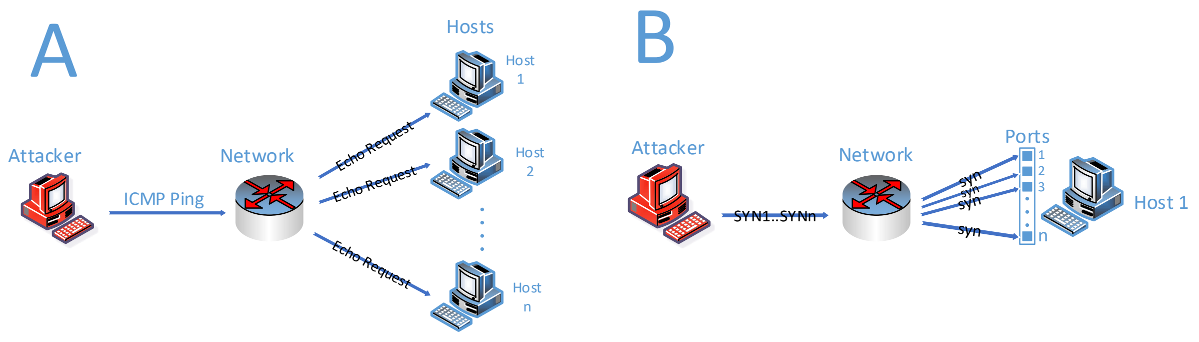


Figure 4.6 A - Network Scan B - Host Scan

Figure 4.6 displays the basic idea of the two types of scan. The source IP, source port, destination IP, and destination port are basic information used to check scan activity, although, we need more information to make a decision of current and future situation. In port scan, the attacker sends many TCP/IP SYN packets to many ports in a single host and waits for a reply (TCP/IP ACK), and any port that responds with (SYN-ACK) is means this port is open. Therefore, the traffic that containing a scan activity will in most cases contain a high volume of SYN and SYN-ACK packets. In network scan (also IP scan), the attacker exploits the Internet Control Message Protocol (ICMP), which is a mechanism for network problem reporting, to check which host is connected to the network and accessible. The attacker sends ICMP echo request to a network or range of IPs, as a response all alive IPs will send back ICMP echo reply. Therefore, traffic that contains a network scanning will have a high volume of ICMP packets.

4.4.3 Feature- based Traffic Counting

Our method is based on observing the network traffic variables that indicate a surveillance activity is undertaken. To achieve this goal we focus on counting the changes in specific traffic variables. These changes will be recognized by a means of calculating the increment of the number of destination ports or IPs that originated from specific sources more than other sources. Our practical investigation showed that normal traffic might contain large numbers of connections that come from the same source to the same destination with multiple ports. In a typical port scanning calculation, this normal activity will be considered a scan attack. Therefore, we adopt a technique of considering only unique ports that have been contacted at the time window on a specific destination. When it comes to online detection of scan attacks nowadays, a huge amount of data is being transmitted across the internet amongst plenty of addresses back and forth. This causes the counting process to imply a considerable amount of memory that is not available to be allocated for the detection function, especially when this function is intended to be in the network nodes or routers.

Unique Values Estimation*:* As a solution for the counting problem, we decided to apply Cardinality estimation methods that can produce near optimum estimation with a trivial amount of memory required. Cardinality Estimation is the process of approximating the number of unique elements in multisets huge data streams [228]. In the case of scan detection, we approximate the number of distinct source IPs that contact unique destination ports or IPs. The more destination ports/IPs are contacted by the same the source counted in the traffic, the more likely this source is a scanner or attacker. In our counting section, we adopt the HyperLogLog algorithm [228], which currently stands as the state-of-the-art method for cardinality estimation, the rest of this section will display this algorithm.

Counting distinct elements in large datasets is an important process in many fields for different purposes such as counting the distinct customers using a service of a company is economically beneficial for the company [229]. In database systems, this concept is used for performance optimization, when these systems perform complex operations such as joins of tables. For example, the complexity of table joining stems from the number of distinct rows in involved tables, that is: more distinct rows means a more complex join operation. Security is another major field that uses the counting process in many aspects such as web activities logging, network anomaly detection, etc. For example, counting the distinct requests that have been made to a server within a time slice is a way of identifying volume attacks such as Denial of service attack (Dos) [230].

The reason behind using counting distinct elements for detecting scanning activities is that the normal number of ports used in normal web and network activities is a limited set of ports; these ports are used continuously hence appears repeatedly in the traffic as shown in figure 4.7. Therefore, the appearance of many new (unique) ports in the traffic form outside the normal ports set is abnormal and needs to be drawn to the attention of the security monitoring team; the following figure displayed this concept.

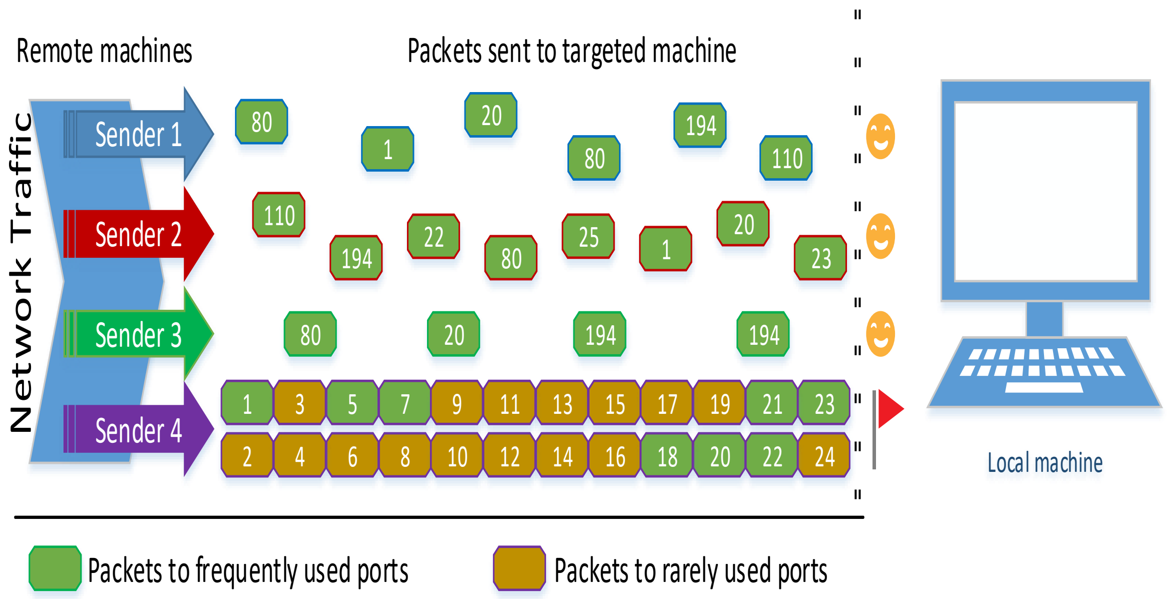


Figure 4.7 Usual and unusual ports.

Despite the aforementioned advantages of cardinality counting, this process becomes infeasible for counting the exact number of the distinct elements due to the increase in the memory required. The insufficiency in memory might occur in the case of a huge number of distinct elements to be counted, or in the case of limited memory available for counting when intended to be placed in network apparatus such as routers. To clarify how "huge" is the number of distinct elements that might cumulatively exceed the available memory, consider the brute force approach of counting. For dataset D that has *d* distinct elements, each new distinct element should be saved in memory directly on arrival (appearance in the traffic), if each element needs $\log_2(element)$ bits to save the element in the memory. Counting for big datasets that *d* contain billions, will soon reach a need of $\log_2(element)$ Gigabytes memory [230].

To overcome the problems of counting; many statisticians proposed different methods and algorithms to estimate rather than counting the exact number of distinct elements, where this estimation is memory friendly and near to optimal with acceptable error ratio such as in [231][232][233] [228][234] [235] and many others. Because of its memory efficiency and very low error ratio we will adopt the algorithm proposed by [228] in our solution for estimating the counts of traffic variables that are used for scan detection.

HyperLogLog (HLL) algorithm is proposed as near-optimal cardinality estimation algorithm, as shown in figure 4.8.

```
The Hyperloglog algorithm
Input: T multiset of items from domain O.
Suppose $t = 2^b$ where $b \in W_{>0}$;
Initialize a set of $t$ registers, $T[1], \ldots, T[t]$
  For a ∈ T do
Set $d := h(a)$;
Set $k := 1 + \langle a_1 a_2 \ldots a_b \rangle$ ; {the binary address of the first b bits of a}
Set $n := a_{b+1}\, a_{b+2} \ldots$ ;
Set $T[k] := \max(T[k], \rho(n))$;
Compute
                    $Q = \left(\sum_{k=1}^{t} 2^{-T[k]}\right)^{-1}$
Return $R := \alpha_t t^2 Q$ with $\alpha_t$ as given by equation (4)
```

Figure 4.8. The Hyperloglog Algorithm.

The accuracy of this algorithm is claimed to reach optimal estimation with the error ratio of 2%, where the error is caused by the randomness used by HLL (as in all efficient cardinality estimators). This algorithm input is a stream of a multiset T of data items. The output is the cardinality estimation of T elements. Choosing an appropriate hash function *h*, every element in the T has been hashed. After hashing and saving the resulting values in memory (in binary), the algorithm checks a specific bit-pattern that arrives in a stochastic averaging fashion. For a string $n \in \{0,1\}^{\infty}$, let $\rho(n)$ indicate the location of the leftmost 1 (the number the initial string of zeros in addition to1). The data stream T is divided into substreams $T_1 \ldots T_t$, according to the beginning bits *b* of the binary-hashed values of substreams elements, that is $t = 2^b$ and all substreams are handled individually.

For any given substream $N \equiv T_k$, we observe:

$$Max(N) := \max_{n \in N} \rho(n), \tag{2}$$

The algorithm scans (and store in registers *T*[k]) the values of T^k of the Max(T_k) for k= 1…t. then the algorithm computes the harmonic mean as following in equation 3 :

$$Q = \left(\sum_{k=1}^{t} 2^{-T[k]}\right)^{-1} \tag{3}$$

Finally, return a normalization of (2) as following:

$$R \coloneqq \frac{\alpha_t t^2}{\sum_{k=1}^{t} 2^{-T^{(k)}}} \tag{4}$$

Where

$$\alpha_t \coloneqq \left(t \int_0^\infty \left(\log_2\left(\frac{2+u}{1+u}\right)\right)^t du\right)^{-1}.$$

The algorithm is based on the following understanding. Assign *t* as the unknown number of distinct values in the multiset stream T. each subset of this stream (e.g. packets to specific destination from the whole traffic) will contain an approximation of *t/m* elements. The maximum memory needed to be allocated to the process of approximation is close to $\log_2(t/m)$. The harmonic mean (here is *tQ*) of the subsets is likely to have a value close to *t/m*. therefore, t^2Q will approximately equal *t*. normalization to the multiplicative bias produced by t^2Q is then applied using the constant α_t.

To simplify and summarise the mechanism of the algorithm; each element in the stream will be encoded into a binary string using a specific hash function to form an ideal multiset, where the elements are equally distributed over the domain. These binary strings are then grouped into approximately equal size buckets (memory registers), the grouping process is accomplished according to an index representing the first *b* bits of the hash value. At this stage, each bucket contains a subset from the stream and has similar or identical elements (hashed values), and all buckets contain approximately equal number of elements. The algorithm now calculates the harmonic mean of all buckets, and returns a normalization of the harmonic mean. The normalization step is achieved to remove the systematic bias produced by multiplicative operations.

Selective Feature Estimation: As the targeted networks are dealing with increasing data volumes, security solutions that analyse network traffic will reach the point of non-feasibility due to many features to be tested that need increasing memory and processing powers. On the other hand, when we are focusing on scan detection statistically, many packet-level traffic features will be irrelevant to the process; that gives us more freedom in choosing the most relevant and effective information to the detection. Therefore, for detecting port scanning we choose to select one aspect that is the number of new session packets (or first in session) sent from a source to a destination over a time window, hence we monitor the sequence field in the TCP packet header. The use of the later technique is inspired by work proposed by [207] with difference in the featured used, as we depend on the sequence of the packet rather than the size of packet in the mentioned work. For IP scanning we select to focus on the type field of the ICPM packet header that is associated with IP scanning. Analysing only the fine-grained information rather than analysing the whole packet header information will significantly decrease the amount of information to be analysed and hence decrease the needed resources as shown in figure 4.9.

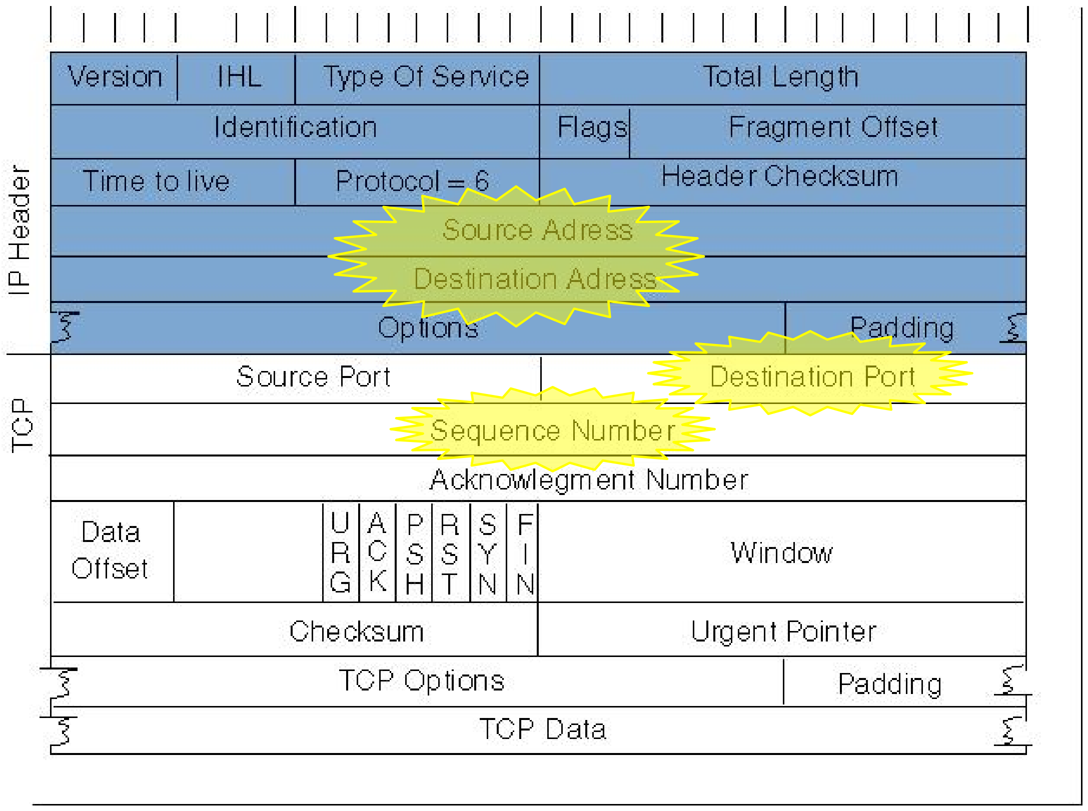


Figure 4.9 Packet-header Selective Feature Monitoring [236]

4.4.4 Anomaly Identification

In this step, the calculations performed in the Feature- based traffic counting will be used to identify the odd elements in the stream. Many methods and models have proposed to tackle the problem of anomaly detection using different parameters and aspects of the network traffic. Our focus is to sense the network for any attack or attempt to attack signs. Surveillance and information gathering is the initial step that precedes an actual attack, IP/Port scanning is the main activity for surveillance. To identify anomalies, we will focus on the scanning activities and the type of changes that this scanning made to network traffic. We based our assumption about the anomalous traffic contents on the basis of IP and Port scanning, that is sending a request to multiple ports on a specific host or multiple hosts on a specific network and in both types there will be an increment in the number of requests to a specific host or network. In the case of port scanning, we will focus on the abnormal requests to a specific host that exceeds the average (normal) traffic to other hosts in the same traffic.

Defining the abnormality in a network is a challenging task as networks nowadays are so versatile and no such traffic can be considered "standard", so it can be the benchmark to check the abnormality of tested traffic. Therefore, many and different methods and algorithms such as statistical [237][238], classification-based [239][240], soft computing [241][242], knowledge based [243][244] and many others are proposed to identify the abnormality in traffic according to the perception of anomaly, and what and when a traffic could be considered as anomalous. In our work, we focus on statistical methods as it computationally feasible, no prior knowledge is needed nor training time; that enables the solution to perform sensing and detection scanning activities online in different spots even those that have moderate computational capabilities such as network routers.

To identify the scanning activities, we will use the outlier detection to isolate the element that receives a spike in requests over the other elements in the domain. For this reason, we use Grubbs Test to identify the IP or network that receives an abnormal number of requests. Grubbs Test [226] is used to detect a single outlier in a univariate data set that follows a near normal distribution. Here we are using the same group to define normal and abnormal numbers of requests as depicted in figure 4.10, as the outlier of this group will be considered the abnormal. Using the same data instantly is hugely beneficial as there is no need for test dataset and training, and this method is adoptive to the situation of the network, rather than depending on previously set settings of what normal and what is abnormal.

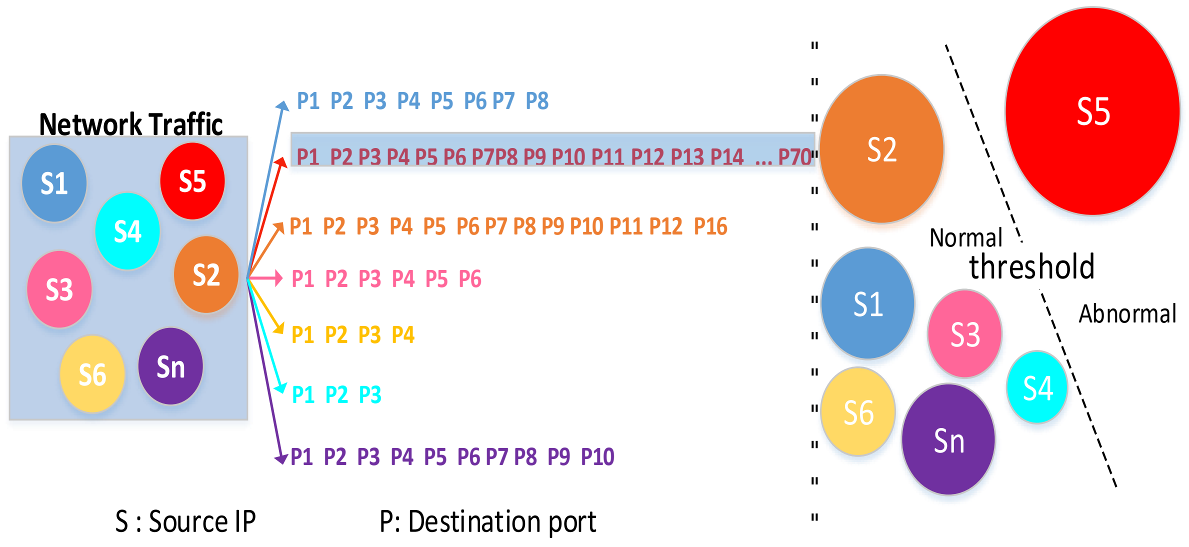


Figure 4.10 Identify the Source IP Connecting to Abnormal Number of Destination Ports.

We adopted a modified implementation of the Grubbs Test that tests single value at a time. In our implementation, multiple outliers can be detected according to the significance level. Grubbs test in essence is a statistical method for detecting a single outlier in a univariate dataset that closely follows Normal Distribution. For a sample of *b* observations, the outlier will be calculated as follows:

$$Gr = \frac{\max|n_i - \bar{n}|}{s} \tag{5}$$

Where n_i is the value under test, $\bar{n}$ is the mean for the whole set, while s is the standard deviation as follows:

$$s = \left\{\frac{\sum(n_i - \bar{n})^2}{b-1}\right\}^{1/2} = \left\{\frac{b\sum n_i^2 - (\sum n_i)^2}{b(b-1)}\right\}^{1/2} \tag{6}$$

Equation 6 is an estimation of standard deviation for the sample set with b-1 degrees of freedom.

Dynamic Threshold Quantifying: the purpose of threshold is to separate between the normal and abnormal for a given point of data. Defining the threshold is a sensitive and important process as it is the base to judge if tested data is normal or anomalous. This means security administrators will make different arrangements based on the judgment of the threshold, these arrangements might be escalated to halt the system operation in severe cases. Therefore, choosing a suitable threshold is a challenging issue to satisfy the trade-off of not affecting the performance of the system by choosing a strict threshold, or jeopardizing system security by choosing a permissive verge. Therefore, many methods in the literature have been proposed according to the view of researchers with regards to system security and system situation. For example, in [245] researchers consider the highest point from the mean of normal data as a threshold. In [246] they depend on the Mahalanobis distance of the training set to define the threshold as well. While in [247] the threshold is set at -3δ to $+3\delta$ (where δ is the standard deviation), and researchers claimed that 99% of the normal sample is lying beneath.

Based on the last claim in [247], they adopted the threshold for scan detection to be greater than 3δ, which appeared to be suitable for our system after many tunings made to choose the most suitable threshold. Although defining a fixed threshold may be a bit risky as many attacks might be successfully accomplished under this threshold, in our system analysing the test data itself makes our threshold semi-dynamic as it changes according to tested data rather than a fixed threshold built according to training data and fixed for all data sets. In addition, this specific type of scan attack is approximately subjected to normal distribution, that is amongst the whole traffic of the network, the anomalous traffic will exceed the 3δ.

Dynamic Decision-making: Our algorithm can be used to operate and make decisions of scanning detection in an online fashion. This could be achieved as a consequence of using a statistical method, which is light in calculations and can be performed to produce results in very short times. The process of making a decision is executed in a periodically repeated process, choosing and calibrating a reasonably short time-window will lead the processes of estimating, scan detection, and decision making to be rapidly consecutive, that this near to concurrent working fashion is giving the online results. In any fashion of our algorithm, the core of the decision-making process is to receive the calculations from the anomaly identification process, measure the anomalous values in regards to the rest of the traffic, and decide upon the threshold calculated from the same examined set. Those sources that sends anomalous numbers of requests (S) are isolated from the rest of the other sources in the traffic (N), and then we calculate the mean μ and standard deviation δ for the normal set. Then we calculate the distance between S and C the central point of N using Euclidian distance. C is a virtual point representing the middle of the normal values, and calculated from μ of number of requests per source and μ time. C was proposed as a reference point to be used to measure the Euclidian distance between C and S. Then the resulting distance is compared to a threshold that can be defined as 3δ of N, and is considered a scan when ($\overline{S\,C}$) is greater than 3δ. The dynamicity of our algorithm stems from the fact that it compares the number of connections from a source to other connections sent from other sources for the same time window, rather than setting a threshold before the work of the algorithm as in other different solutions.

4.5 Knowledge Module

In this part of the system, information is stored from different sources for different purposes. Two databases are contained in this module; a database for registering the scanning activities and a database for vulnerabilities. The first database is the ‘scanning database’ that stores all scanning activities that are launched against the system along with sources of these activities, and the risk associated with those sources. The second database is the ‘vulnerabilities database’ that contains information about the system vulnerabilities from different sources. This information will be used in the prediction process by analysing the current vulnerabilities of the system. The information about vulnerabilities in the system comes from two primary sources; host vulnerability scanners and public vulnerability databases. These data are collected from the sources and prepared for processing by select features in the data that will used to compute the properties related to the vulnerabilities such as severity and exploitability.

In conceptual design of the vulnerability database, the most suitable model for this purpose is an entity-relationship model that could clarify the interaction among vulnerabilities, attacks, and risks. The data is gathered from the public vulnerability database that contains recent information about the discovered vulnerabilities such as National Vulnerability Database (NVD) and Common Vulnerabilities and Exposures (CVE). These databases contain detailed information about the vulnerabilities; we are only interested in the information related to severity and exploitability of the vulnerability that is used to calculate the exposure level. Information gathering will includes data cleaning that unifies the format of data to be further processed.

4.6 Vulnerabilities Analyser

This module is responsible for analysing and evaluating the vulnerabilities is the system, where vulnerabilities scanners search for discovering the security bugs and faults using the information of the vulnerabilities from the knowledge module. This module performs the processes related to vulnerability namely; the severity and exploitability of each vulnerability. The output of this module would be introduced to the prediction module for calculating risk and then making prediction. First, we give a formal definition of the concept's vulnerabilities, severity, exploitability, and Degree of exposure as follows:

Vulnerabilities: are the weaknesses in the system that could be exploited by an adversary to attack the system. The set of the system vulnerabilities can be denoted as $V = \{v_1, v_2, \ldots v_n\}$. $v(h_\alpha) \subseteq V$ is the set of vulnerabilities that could be exploited to launch the threat h_α..

Degree of exposure: this is a mechanism to gauge the amount of the system exposure to intrusions. This mechanism can be defined by the number of the vulnerabilities in the system; however, we adapt two defining factors that control these vulnerabilities that are the severity and the exploitability of the vulnerabilities. The severity factor relates to how critical this weakness is, which means how severe is the impact on the system if this vulnerability is exploited and the type of harm to the organization which could result consequently. The severity of a vulnerability v_i is denoted by $S(v_i)$, and may take any value in the range {1, 2, 3}. The exploitability factor refers to the ease of exploiting the vulnerability, the level of skill needed for the process, and the availability of the exploit code. The exploitability of the vulnerability v_i can be denoted as $X(v_i)$ and the value for this factor is in the range of {0, 1, 2, 3}. Although exploitability as well as severity could take any quantitative value, as proof of concept we choose specific ranges of numeric values. Thus the exposure that a vulnerability v_i could cause is:

$$Ex(v_i) = S(v_i) \times X(v_i) \tag{7}$$

In (7) the calculation is to measure the impact of a single vulnerability on the system. As different vulnerabilities could be used to harm the system, the total exposure will be:

$$Ex(t_\propto) = \sum_{i=1}^{n} S(v_i) \times X(v_i) \tag{8}$$

Where $t_\propto$ is the time of measuring the exposure, n is the total number of vulnerabilities. However, the greatest impact on the system occurs from the vulnerability that leads to maximum exposure. Therefore, the level of exposure can be computed as follows:

$$Ex = \prod_{i=0}^{n} \max\ (S(v_i))\, X(v_i) \tag{9}$$

Equation (9) laid the base to calculate the level of exposure for the system in general. However, to compute the two attributes of Ex precisely namely the severity and exploitability, we exploit the Common Vulnerability Scoring System (CVSS) [33]. In this system different attributes such as impact, temporal, and environmental are considered to score the vulnerabilities. As we focused in our calculation on the most important attributes that we think have more effect on the on the system, especially when the network is used as medium of the attacks, therefore, the severity of a vulnerability is determined by the impact of this vulnerability on the Confidentiality (C), Integrity (I), and Availability (A) of the system. The exploitability is influenced by three factors that are determined by how exposed the system is to the adversaries, these factors are Attack Complexity (AC), Attack Vector (AV), and Exploit Code Maturity (E).

4.6.1 Severity Metrics

Confidentiality Impact is the first metric that we account for in the severity of vulnerability. This metric measures the harm to the confidentiality of the information that is stored in the system, as confidentiality implies restricting the access and disclosure to information to only authorized persons, and preventing unauthorized people from accessing or reading the protected information. The potential values that the confidentiality impact could take are displayed in table 4.1 as follows:

Table 4.1. Confidentiality Impact Values

Value	**Effect**
High	There is a major damage to the confidentiality that results in the information being exposed to the attacker, or highly restricted information is revealed causing a broad harm to the system such as an adversary gaining administrator` password or encryption key.
Low	There is minor damage to the confidentiality that results in some information being exposed to the attacker with no control for the attacker to attain specific information to expose.
None	No countable effect on the confidentiality

Integrity Impact metric measures the effect of the integrity in case a related vulnerability is exploited successfully. This term is concerned with the trustworthiness of the information, so it is a vital metric in our calculations. The value of this metric is increased according to the consequences of a successful exploitation of related vulnerability. The potential values of the Integrity impact are displayed in as follows:

Table 4.2 Integrity Impact Values

Value	**Effect**
High	There is major damage to the Integrity or a whole damage of protection. This results in attacker being capable to make changes to encrypted data in targeted system. Another scenario of damage might occur to the system is that attacker might only able to change specific data, but these changes affect the system severely.
Low	Attacker might be able to perform changes to the data to some extent, but limited minor damage might occur to the system.
None	No countable effect on the Integrity

The last factor that contributes to the severity of vulnerability is the Availability Impact. This metric measures the effect on the availability that could result when related vulnerability is exploited. Availability refers to providing the desired service on time by the functional part whether it is software or hardware. While the two latter metrics were targeting the data itself, this metric targets the functionality of the vulnerable part not what is contained inside it. Based on that the attacks that target the availability are aiming to block or minimize the accessibility to the system by consuming the network bandwidth, processing capacities, disk spaces limits. Table 4.3 clarifies the levels of the availability impacts and their potential values:

Table 4.3 Availability Impact Values

Value	**Effect**
High	Complete blocking of service availability, causing the attacker to get a full access denial for legit customers or resources beneficiaries to the vulnerable part of the system. This condition could last for different periods such as only during the time of attack launching, or for a longer time even after the ending of the attack.
Low	There is a partial interruption in the performance of the resource availability. This be in the form of the resources in the vulnerable part being even partially available any time or fully available for discrete periods.
None	No countable effect on the Availability

4.6.2 Exploitability Metrics

As mentioned earlier, these metrics rate the vulnerability based on the methods required in exploitation, the type of potential attackers in regard to their location to the network, and the facilities available for a successful exploitation. The first metric that contributes to the exploitability of a vulnerability is the Attack Complexity, is the conditions and requirements to accomplish successful attacks and normally these conditions are out of the attacker`s control. This means how easy or difficult it is to launch attack in regard to programming skills, the preparations prior to the attacks such as information gathering and target scanning, the current system configurations, etc. Table 4.4 lists these values regarding the Attack Complexity:

Table 4.4 Attack Complexity Values

Value	**Effect**
High	A successful exploitation of the vulnerability requires a measurable amount of effort to prepare and execute the code and approach of exploitation. For example, target-specific information gathering must be done, special

	preparation to the targeted environment to guarantee a successful attack, gaining a valid logical position in the network, etc.
Low	No special exploitation conditions required, and the attacker might repeat the exploitations to the same vulnerable part with high probability of success.

The second metric contributing to exploitability is the Attack Vector, which is the different paths and means by which exploiting the vulnerability is possible. This factor defines the logical and geographical areas in which the vulnerability is accessible and possibly exploitable in an attack. This metric is based on an intuition that the number of potential attackers from the internet for a specific vulnerability is bigger than the number of potential attackers in smaller networks. The values of this factor are displayed in table 4.5:

Table 4.5 Attack Vector Values

Value	**Description**
Network	Potential attackers could exploit the vulnerability from remote distances at wide range networks, which is called a "remotely exploitable" vulnerability. An obvious example of this type is exploiting the vulnerability CVE-2004-0230 to launch a denial of service attack.
Local	The vulnerability could be exploited with local access using a path that provides read/write/execute capabilities.
Adjacent	In this type, the vulnerability could be exploited from a machine that is connected to the same network stack. An example of this is ARP scanning or flooding that could lead to DoS on the shared segment.

The last metric that contributes to the extent of exploiting a vulnerability is the Exploit Code Maturity, which determines if there is a code written to exploit the vulnerability or not and how complex is it to use that code if it is available. This in turn reflects the probability of attacking using this vulnerability, estimated by considering state-of-art exploiting techniques, exploit codes availability, and active exploitations. The probability of exploiting a specific vulnerability will increase with the public availability of exploiting codes that are easy to use, because there are a wide range of attackers even those who are less skilled. The values for this metric are shown in table 4.6 as follows:

Table 4.6 Exploit Code Maturity Values

Value	**Description**
High	Active and automatic exploitations are presented and available to use, or the vulnerability exploitation could be achieved manually without code. Another situation is when the exploit is active in different conditions or is delivered using worms or viruses.
Low	Exploitation is available in very limited ranges and needs sophisticated skills to work successfully. In addition, when the exploitation is theoretical and no known exploitation is available.
None	No exploitation is available.

CVSS does not consider the last metric as a part of factors that contribute to exploitability; rather they account for it as a temporal metric. On the other hand, we believe that this metric directly affects the ability to exploit a vulnerability, because a vulnerability will not be considering an exposing point unless it can be used in malicious action. The latter in turn cannot be achievable without an exploitation code or facility. Therefore, we consider that a vulnerability with no exploitation has a minor or negligible effect on exposure level for the system.

To compute equation (9) in detail, we substitute the variables with their contributing factors, and clarify the amount of effect each factor (or metric) impacts the related variable. In severity, it is obvious that the more the vulnerability impact on the security aspect the higher severity this vulnerability will take. For example, a bug that impacted the confidentiality and integrity is more severe than one that has a single effect. Therefore, it is suitable to suggest that the severity of vulnerability is the product of the values of severity factors, this can be displayed as follows:

$$S(v_i) = C_{v_i} \times I_{v_i} \times A_{v_i} \qquad (10)$$

Where C is the confidentiality, I is the integrity, and A is availability of the system. These factors should always have positive values as long as the vulnerability has an effect on the security aspect. On the other hand, a factor with null value is not influencing the severity, and hence should not considered in the calculations. The same process can be used with the exploitability metrics; however, different values of attack complexity and attack vector can escalate the value of exploitability only in the presence of usable exploit code. Therefore, we forged the value of exploitability to take the form:

$$X(v_i) = (AC_{v_i} + AV_{v_i}) \times E_{v_i} \tag{11}$$

Where AC is attack complexity, AV is attack vector, and E is exploit code maturity. Although E could take numerical value of null as no exploit code available, however for the sake of prediction, we consider there is probably an exploit code in private sources, which always has a positive value.

4.7 Prediction Module

The process of making predictions about the future state of the system is performed in this module. The prediction of the future state of the system is dependent on the current level of the system security and the network situation. To form a prediction of the probability of the system to be targeted by a malicious activity, we incorporate two of the main concepts of system security that is how insecure the system and is there any intention to target the system.

To measure the insecurity of the system we evaluate the risk level of the system. The literature showed that computer systems risk level comprises three main factors; threats, vulnerabilities, and assets. Threats are those malicious acts that the system is subjected to such as causing harm to the system, stealing information from the system, using the system as a platform to attack targets that are more important, etc. Normally, the occurrence of these threats is uncertain and subject to likelihood. The second contributor in risk level is the vulnerabilities of the system and associated probability of discovery and exposure. The last factor of risk is the assets of the system and the consequences of a successful attack on these assets.

Risk factors are controlled by an external party, that is the attackers, and an internal party, that is the defenders of the system. Because attackers are unknown until attacks are detected, risk factors relating to the attackers can only be quantified by calculating the probability of threat occurrence. On the other hand, the defender can derive information about the system by evaluating the vulnerabilities and their consequences. While the assets of the system represent a main pillar in risk assessment, we discard including the assets in our predictive risk assessment because we focus on predicting the potential network based attacks where the system assets are not yet known to the adversaries, and hence not taken to be such a major contributor to the risks in the system.

4.7.1 Risk Model

The risk model we suggest depends on essential aspects, which will be discussed along with the adapted methods to calculate the risk efficiently. Our model is analysing the risk that the system faces, this analysis is based on factors that will be discussed as follows:

Threat: is a potential source of harm that occurs to the system by targeting the system software, applications, hardware, or functionality. Therefore, any type of attack is considered a threat, the set of threats is denoted by $Th = \{h_1, h_2, \ldots h_n\}$, where Th is the threat produced by a set of known attacks. h_α is known attack that is composed of sequence of events $E(h_\alpha) = \{e_1, e_2, \ldots\}$. The attack is detected when all these events occurred in sequence.

Likelihood: the likelihood of a threat is the probability of occurrence when related attacks are launched, receiving a specific type of traffic in a specific sequence, or when a security alert is raised indicating an attack in preparation. We use the term attack in preparation as an indication for the reconnaissance stage, where the attacker collects information about the target by locating and mapping the victims using scan techniques. As the probability of a threat depends on malicious events to occur and these events preceded by information collection, we associate the probability of a threat with information gathering. The initial probability of a threat Th denoted $P(Th)$ can be calculated proactively depending on factors such as type of defence software used, sensitivity of system data, system assets values, etc. $P(Th)$ increases when any events of $E(h_\alpha)$ of related attacks h_α have been detected. We use the notation $P(Th \mid h_\alpha)$ to express the conditional probability of threat occurrence when an attack has happened. This leads us to infer that the probability of Th is also increased when a security incident e_i has happened; we can denote this probability in the form $P(Th \mid e_i)$. We will use $P(Th \mid h_\alpha)$ to represent the conditional probability between threats and attack/event. The probability of the system to be under a threat will be as follows:

$$P(Th_j) = \sum_{i=1}^{m} P(Th_j) \times P(Th_j|h_i) \qquad (12)$$

Where $P(Th_j)$ is the overall probability of threats, $P(Th_j)$ is the initial probability of the jth threat, $P(Th_j|h_i)$ is the conditional probability of the ith attack leads to jth threat. In equation (12), we calculate the total probability of threat that could occur to the system due to different attacks related to that threat. As we calculate the probability of the system to be under the danger of any threats that could affect the system, this will make the probability of the absolute threat the accumulation of all possible threats probabilities. The probability of the absolute threat then can be expressed as follows:

$$P(Th_{\propto}) = \sum_{j=1}^{n} \sum_{i=1}^{m} P(Th_j) \times P(Th_j|h_i) \qquad (13)$$

Where $j = 1..n$ is the number of different possible threats, $i = 1..m$ is the number of all attacks related to that specific threat.

4.7.2 Risk calculation:

The risk to the system represents the accumulative risks stem from all threats that the system might face. The risk will then be calculated by the product of threat and the exposure level as follows:

$$Risk = Ex \times P(Th_{\propto}) \qquad (14)$$

Where Ex is level of exposure. As we stated earlier that the start of gathering information is intuitively referred to as an interest in the system that will probably be malicious. Therefore, we increase the risk value each time the system is scanned, which will applied to equation (14) and expressed as follows:

$$Risk = \prod_{k=1}^{x} S_k \times (\sum_{j=1}^{n} \sum_{i=1}^{m} P(Th_j) \times P(Th_j|h_i)) \times \prod_{i=0}^{n} \max(S(v_i))\, X(v_i) \qquad (15)$$

Where x is the number of different sources, S_k is the scan activity that targets the system. In equation (15) the risk is calculated numerically that is suitable for further calculation, however, we aimed to classify the risk value to qualitative levels. Therefore, to convert the risk into qualitative indication, we project equation (15) into a Risk Matrix that output the risk as qualitative value. The Risk Matrix will result three levels of risk: *Low, Medium, and High* as shown in the next table.

Table 4.7 Risk Matrix with Absolute Threat

Level of Exposure / Absolute Threat	Low	Moderate	High
Low	Low	Low	Medium
Moderate	Low	Medium	High
High	Medium	High	Critical

To apply equation (13) with Risk Matrix, we consider the output of table 4.8 into second stage Risk matrix in regards to network situation. We retain the same level of risk as long as the network traffic is behaving normally; raise the risk to next level with port scan and vulnerability scan. With this risk matrix, we proposed a new level of risk that is the 'Critical' level, where we predict that an attack is in preparation to be launched against the system. The next Table shows the risk matrix for the effect of the network situation on risk level:

Table 4.8 Risk Matrix with Scanning Activities

Risk / Network Situation	Low	Moderate	High
Normal	Low	Moderate	High
Port scans	Moderate	High	Critical
Vulnerability scan	High	Critical	Critical

4.8 Reporting Module

In this module the output of the prediction module, which is the outcome of the whole system, will be informed to the security administration. The informative notifications that will be sent to will give the security team the time they need to countermeasure the attacks while the attacker still in the preparation stage. This will happen before the dangerous parts of the attack being commenced. The security notifications will contain level of the current of system security, type of network preparation (IP scan, Port scan, vulnerability scan).

4.9 Summary

This chapter presented an overview of the IPFCC framework and a detailed explanation of the framework design in section 4.2. This section provided details and justifications for the framework modules and design choices.

A novel scan-detection technique specifically developed for the IPFCC framework has been presented in this chapter, in order to overcome the limitations of other techniques found in literature. This technique uses statistical and probabilistic methods to detect the anomalies in the network traffic, the type that indicates the presence of attack preparation. It combines the statistical analysis of selected attributes of the traffic, as well as undertaking comparative outlier analysis of tested traffic slices.

This chapter presented an adaptation of risk assessment that includes a customization of the well-known Common Vulnerability Scoring System (CVSS) in order to analyse the vulnerabilities in the system and score it according to the impacting attributes. This method uses vulnerabilities scanners and global vulnerability databases to discover the system security holes.

Finally, this chapter illustrated our method of decision-making to predict intrusions by using risk matrix technique that incorporated two levels to give the final verdict. The first level defined the risk value according to the level of exposure and the absolute threat to the system, and the second level checked the resulting risk to the network situation and produced the prediction of potential intrusions.

Chapter 5

Implementation

In order to test the success of the techniques used to propose the IPFCC framework, it is crucial to produce a working implementation. This validates the claims of the thesis and verifies that IPFCC meets the aims, objectives and requirements that were defined initially. This chapter provides the implementation details of the proposed IPFCC framework in chapter 4, all the included techniques, evaluation approaches and the testbeds. This thesis conducted an exploration study of statistical and probabilistic Ports-scan detection methods, a preview of the rebuilding, the explored methods and the tools and simulations used to achieve re-generation of the other methods. These two parts of implementation will be covered by the two main sections of this chapter.

5.1 Preliminary Analysis and Experimental Testbed

Prior to propose our solution, the extensive exploration of the network based attacks done throughout the literature review and background produced a solid certainty to us that the best way to predict the attacks is to identify their indications. Based on that, we conclude that the IP scan and port scan are the best indications to predict the potential attacks. Therefore, we launched practical investigation to identify the traces of scan activities in network traffic when the traffic contain scanning packets or normal traffic. The investigation includes launching a real scans from machine A that contains a scanner software against machine B that contains packet sniffer software in subnet via universal server, as shown in figure 5.1.

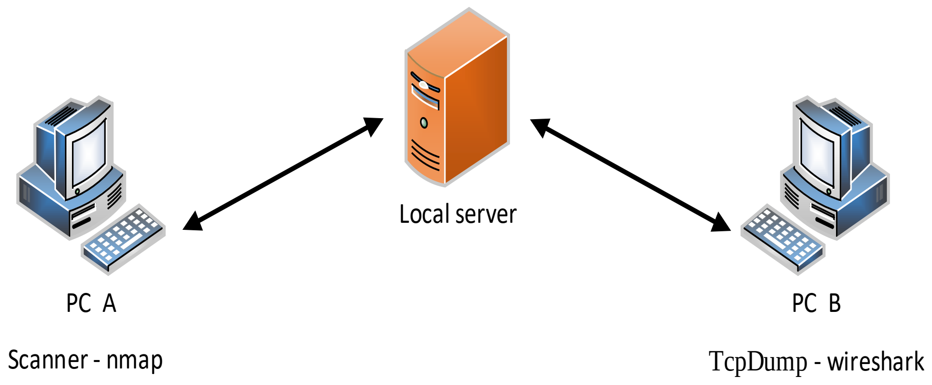
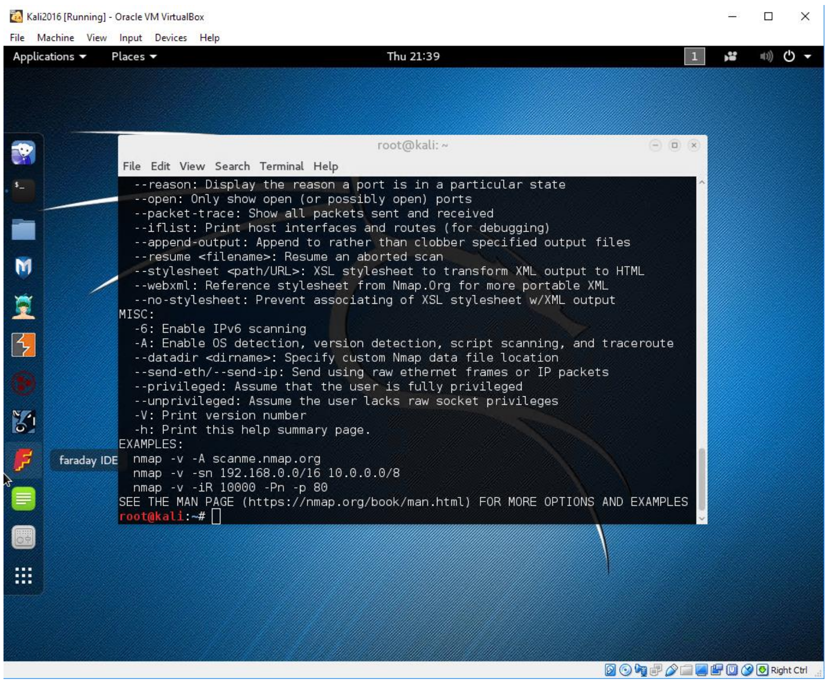


Figure 5.1 Testbed Settup

A-The first machine represents the attacker machine, and running windows 10 64-bit OS, core i7, 3.40 GHz CPU, and 16 GB memory. To avoid any damages and complications, we installed the Oracle VM VirtualBox [248] that is virtual environment allows to create virtual machines with the ability to host fully functional guest operating systems. To launch the scan, we create a virtual machine and install the Kali Linux [249] that used for penetration test, which contains different tools and facilities for testing the network and host such as Nmap [136] as shown in figure 5.2.



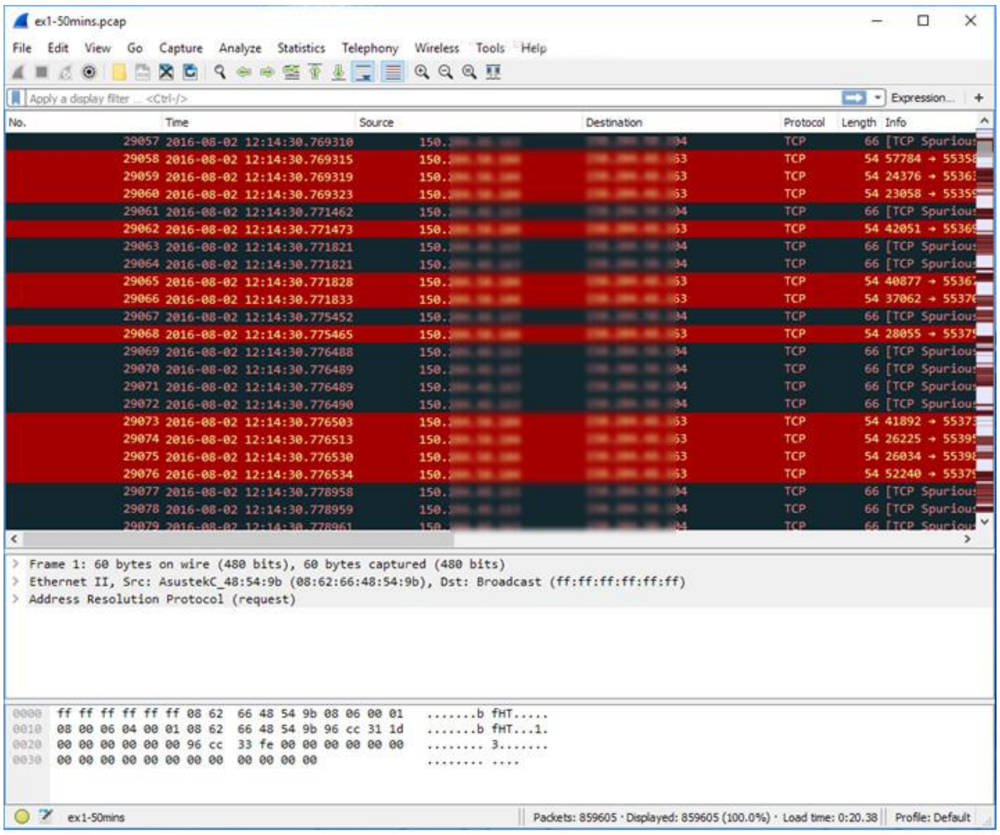
Kali2016 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications
Places
Thu 21:39
root@kali: ~
File Edit View Search Terminal Help
faraday IDE
Right Ctrl

```
  --reason: Display the reason a port is in a particular state
  --open: Only show open (or possibly open) ports
  --packet-trace: Show all packets sent and received
  --iflist: Print host interfaces and routes (for debugging)
  --append-output: Append to rather than clobber specified output files
  --resume <filename>: Resume an aborted scan
  --stylesheet <path/URL>: XSL stylesheet to transform XML output to HTML
  --webxml: Reference stylesheet from Nmap.Org for more portable XML
  --no-stylesheet: Prevent associating of XSL stylesheet w/XML output
MISC:
  -6: Enable IPv6 scanning
  -A: Enable OS detection, version detection, script scanning, and traceroute
  --datadir <dirname>: Specify custom Nmap data file location
  --send-eth/--send-ip: Send using raw ethernet frames or IP packets
  --privileged: Assume that the user is fully privileged
  --unprivileged: Assume the user lacks raw socket privileges
  -V: Print version number
  -h: Print this help summary page.
EXAMPLES:
  nmap -v -A scanme.nmap.org
  nmap -v -sn 192.168.0.0/16 10.0.0.0/8
  nmap -v -iR 10000 -Pn -p 80
SEE THE MAN PAGE (https://nmap.org/book/man.html) FOR MORE OPTIONS AND EXAMPLES
root@kali:~# 
```

Figure 5.2 Kali Linux and Nmap

The Nmap is a security scanner that used to discover hosts on a network and the applications running on each host, hence structure a "map" of the network. The initial aim of using Nmap is to generate the scan packets that are embedded in normal traffic.

B-The second machine is the one that been scanned (the victim), where the traffic will be collected. This machine is a separate physical machine with its own IP address and running services and applications. This machine runs Windows 10 64-bit operating system, Core i7 3.6 GHz CPU, and 8 GB of RAM memory. For the purpose of collecting traffic we used the TcpDump [250], which is a well-known packet analyser that operate under command line and working as background application. For deep and thoroughly analysing the traffic that contain scan packets, we used the Wireshark [251] that is a world standard application for analysing network traffic that offers a graphical user interface and different analytical and statistical options for deep traffic analysis. We launched a port scan against the second machine while using the machine for normal internet access, so the collected traffic contain both normal and abnormal traffic that captured by TcpDump and analysed by Wireshark as shown in figure 5.3.


ex1-50mins.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

```
No.           Time                        Source              Destination       Protocol  Length  Info
       29057 2016-08-02 12:14:30.769310     150.[redacted]      [redacted]94      TCP         66 [TCP Spurious
       29058 2016-08-02 12:14:30.769315     150.[redacted]      [redacted]53      TCP         54 57784 → 55358
       29059 2016-08-02 12:14:30.769319     150.[redacted]      [redacted]53      TCP         54 24376 → 55363
       29060 2016-08-02 12:14:30.769323     150.[redacted]      [redacted]53      TCP         54 23058 → 55359
       29061 2016-08-02 12:14:30.771462     150.[redacted]      [redacted]94      TCP         66 [TCP Spurious
       29062 2016-08-02 12:14:30.771473     150.[redacted]      [redacted]53      TCP         54 42051 → 55369
       29063 2016-08-02 12:14:30.771821     150.[redacted]      [redacted]94      TCP         66 [TCP Spurious
       29064 2016-08-02 12:14:30.771821     150.[redacted]      [redacted]94      TCP         66 [TCP Spurious
       29065 2016-08-02 12:14:30.771828     150.[redacted]      [redacted]53      TCP         54 40877 → 55367
       29066 2016-08-02 12:14:30.771833     150.[redacted]      [redacted]53      TCP         54 37062 → 55376
       29067 2016-08-02 12:14:30.775452     150.[redacted]      [redacted]94      TCP         66 [TCP Spurious
       29068 2016-08-02 12:14:30.775465     150.[redacted]      [redacted]53      TCP         54 28055 → 55379
       29069 2016-08-02 12:14:30.776488     150.[redacted]      [redacted]94      TCP         66 [TCP Spurious
       29070 2016-08-02 12:14:30.776489     150.[redacted]      [redacted]94      TCP         66 [TCP Spurious
       29071 2016-08-02 12:14:30.776489     150.[redacted]      [redacted]94      TCP         66 [TCP Spurious
       29072 2016-08-02 12:14:30.776490     150.[redacted]      [redacted]94      TCP         66 [TCP Spurious
       29073 2016-08-02 12:14:30.776503     150.[redacted]      [redacted]53      TCP         54 41892 → 55373
       29074 2016-08-02 12:14:30.776513     150.[redacted]      [redacted]53      TCP         54 26225 → 55395
       29075 2016-08-02 12:14:30.776530     150.[redacted]      [redacted]53      TCP         54 26034 → 55398
       29076 2016-08-02 12:14:30.776534     150.[redacted]      [redacted]53      TCP         54 52240 → 55379
       29077 2016-08-02 12:14:30.778958     150.[redacted]      [redacted]94      TCP         66 [TCP Spurious
       29078 2016-08-02 12:14:30.778959     150.[redacted]      [redacted]94      TCP         66 [TCP Spurious
       29079 2016-08-02 12:14:30.778961     150.[redacted]      [redacted]94      TCP         66 [TCP Spurious

> Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
> Ethernet II, Src: AsustekC_48:54:9b (08:62:66:48:54:9b), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Address Resolution Protocol (request)

0000  ff ff ff ff ff ff 08 62  66 48 54 9b 08 06 00 01   .......b fHT.....
0010  08 00 06 04 00 01 08 62  66 48 54 9b 96 cc 31 1d   .......b fHT...1.
0020  00 00 00 00 00 00 96 cc  33 fe 00 00 00 00 00 00   ........ 3.......
0030  00 00 00 00 00 00 00 00  00 00 00 00               ........ ....

ex1-50mins        Packets: 859605 · Displayed: 859605 (100.0%) · Load time: 0:20.38    Profile: Default
```

Figure 5.3 Scan Traffic Analysed by Wireshark

C- The experiments were done by recording traffic in the victim machine for 50-minute time window and the scan activity launched at known time from the attacker machine. The initial analysis for the traffic showed that during port scan, every port is scanned by a TCP request packet that its sequence is either 0 or 1, and the for the IP scan is the big number of ICMP echo request packets (normally referred to as type 8) that is send to the network. We identified these types of packets, as these packets are sent as the initial packets of the communication session between two machines. As a session is the communication between source IP and source port to destination IP and destination port, therefore, for each destination port there will an individual communication session with it. The normal communication between two machines is one session that includes establish the link between the machines to use a specific service, and then multiple sending and receiving of packets until completing the service providing. Observe multiple initial packets means establishing multiple sessions from the same source to the same destination, which is either the source requests different service from the destination (that is rarely occurred) or there is an abnormal behaviour.

5.2 IPFCC Framework

Initially, the IPFCC requires having a system snapshot of vulnerabilities that will be used as a basis to operate in real-time. This initial vulnerability snapshot will be used to calculate the risk of the system in the current state, in addition to the vulnerabilities information gathered from public websites. Due to the difficulties of writing and applying a monitoring system for such a huge environment like cloud computing and targeting the network-based attacks, we build our proof of concepts implementation in a simulation environment. All the constituted methods and algorithms have been programmed and incorporated in a framework that worked successfully and consistently.

The implementation and evaluation of the framework is conducted using MATLAB environment. This environment enables developing IPFCC and conduct the evaluation framework performance easily and clearly. In addition, the simulation environment provides us with the opportunity to proof the concepts of the thesis and ability to avoid the complications of licencing or proprietary issues of real clouds centres. Another important reason for depending on the simulation environment is the difficulty of obtaining the security information of organizations, as this type of data is very sensitive and valuable information.

The IPFCC is split into three separate yet integrated processes that are the detection module, knowledge base, and predictor. The detection module is responsible for probing the network for scan activities that include quantifying, thresholding, and decision-making as regards potential scans. The second part of the implemented framework is the knowledge base that was built to contain the information about the vulnerabilities in the system as well as the history of scanned sources. The third part of IPFCC is the vulnerability analyser that calculates the vulnerabilities' attributes in order to evaluate them. The last part of the framework is the prediction model that plays a central role in the framework by processing the information from the other modules and producing a prediction about potential intrusions.

5.2.1 Data Collection and Data Sets

As the framework is implemented in a simulation environment, the data collection depends on a public dataset collected by Lincoln Laboratory of the Massachusetts Institute of Technology (MIT) [252]. This set is called the DARPA dataset, which is a well-known dataset that used to train and evaluate the performance of intrusion detection systems. DARPA intrusion detection data sets have been gathered and distributed by Massachusetts Institute of Technology – Lincoln Laboratory in corporation with Air Force Research Laboratory in the United States of America. Dataset collection carried out in 1998, 1999, and a scenario-specific dataset collected in 2000. All these versions contain labelled attacks along with normal traffic and background traffic. Labelled attacks means all information about the attacks are mentioned such as types of attacks, source machine (attacker), destination machine (victim), timestamp, duration. The dataset has been saved in pcap files that is all packets information is readable by any traffic analysis software [253]. The dataset contains many different attacks including the surveillance activities that include IPSweep scan and PortSweep scan as mentioned in the dataset documents. We focus on the two latter scan types in evaluation of our scanning detection algorithm evaluation. Many researchers have stated the usefulness of this data set and clarify the reasons such as the unavailability of better alternatives, evaluation oriented nature of the data set [254] [255]. To test our solution we used the 1999 DARPA Intrusion Detection Evaluation Dataset that was specifically designed for testing and evaluating the intrusion detection systems, as it contains data collected over three weeks with labelled attacks in a specific week. These attacks are injected in the network traffic along with normal activities traffic.

5.2.2 Scan Detection Module

This module is based entirely on the Selective Dynamic Scan Detection (SASD) algorithm that was proposed in 4.4.1. This module implemented all the methods and algorithms proposed in SASD, which are namely; the counting method, outliers detection method, thresholding algorithm, Euclidian distance. The serialization of the processes is shown as follows:

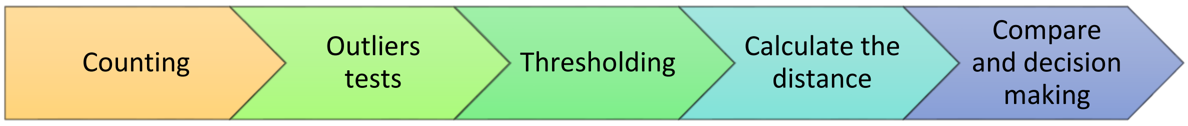
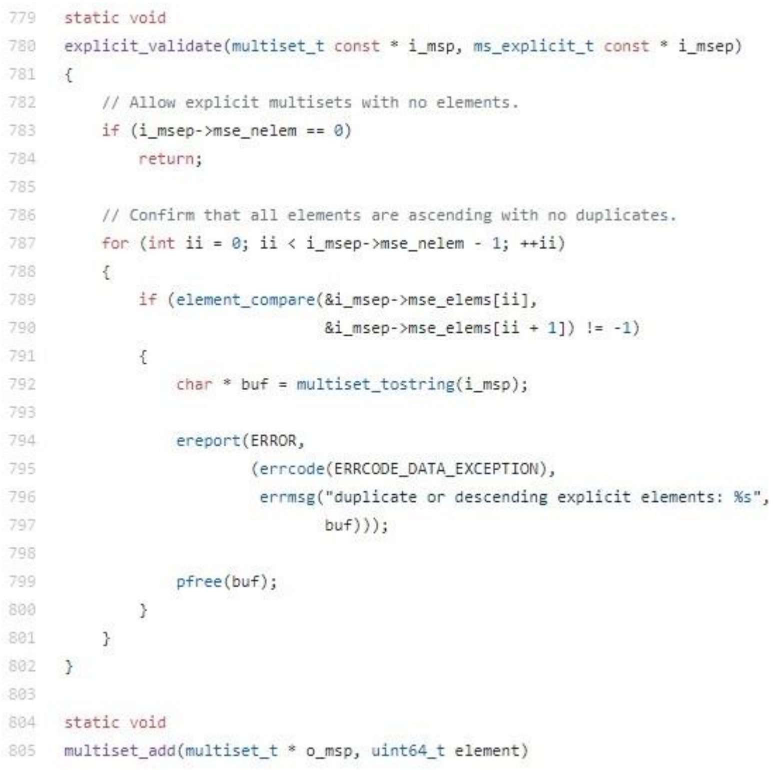


Figure 5.4 The Process Sequence In Detection Module.

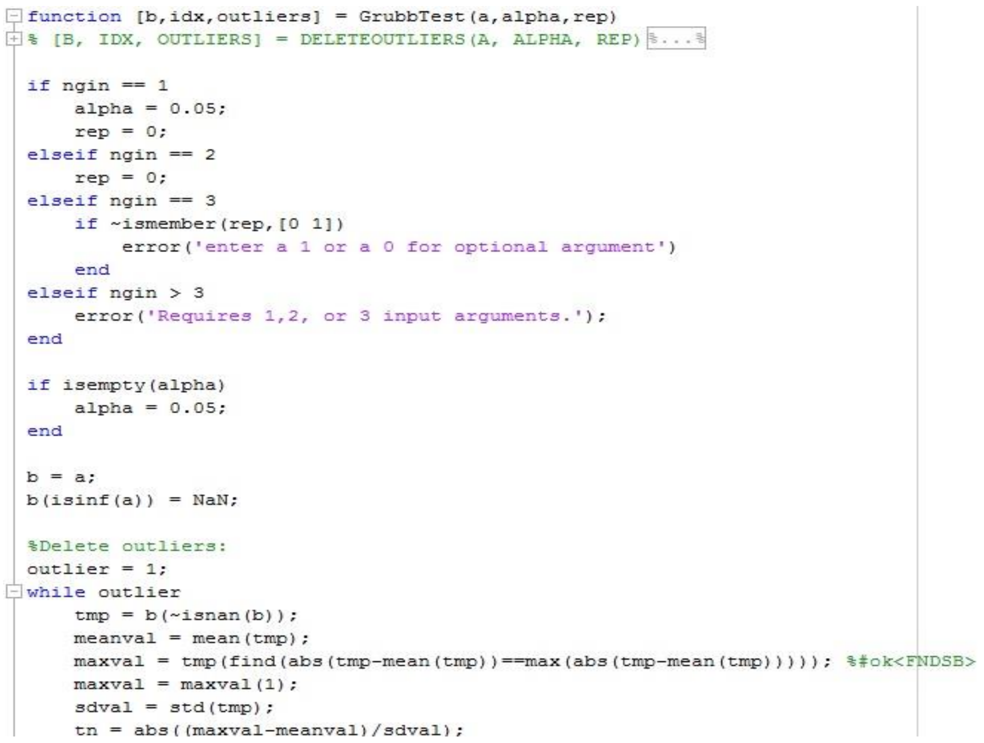
For counting the targeting attributes we used the Hyperloglog algorithm [228] that gives a near optimal estimation for the unique ports numbers or IPs, figure 5.5 shows a segment of Hyperloglog implementation.



```
static void
explicit_validate(multiset_t const * i_msp, ms_explicit_t const * i_msep)
{
    // Allow explicit multisets with no elements.
    if (i_msep->mse_nelem == 0)
        return;

    // Confirm that all elements are ascending with no duplicates.
    for (int ii = 0; ii < i_msep->mse_nelem - 1; ++ii)
    {
        if (element_compare(&i_msep->mse_elems[ii],
                            &i_msep->mse_elems[ii + 1]) != -1)
        {
            char * buf = multiset_tostring(i_msp);

            ereport(ERROR,
                    (errcode(ERRCODE_DATA_EXCEPTION),
                     errmsg("duplicate or descending explicit elements: %s",
                            buf)));

            pfree(buf);
        }
    }
}

static void
multiset_add(multiset_t * o_msp, uint64_t element)
```

Figure 5.5 Code Segment of Hyperloglog Algorithm.

For the outliers test we use a modification of Grubbs` test [226], this algorithm will classify the data into normal and abnormal, the mechanism of the test is depicted in the code excerpt shown in figure 5.6.



```
function [b,idx,outliers] = GrubbTest(a,alpha,rep)
% [B, IDX, OUTLIERS] = DELETEOUTLIERS(A, ALPHA, REP) %...%

if ngin == 1
    alpha = 0.05;
    rep = 0;
elseif ngin == 2
    rep = 0;
elseif ngin == 3
    if ~ismember(rep,[0 1])
        error('enter a 1 or a 0 for optional argument')
    end
elseif ngin > 3
    error('Requires 1,2, or 3 input arguments.');
end

if isempty(alpha)
    alpha = 0.05;
end

b = a;
b(isinf(a)) = NaN;

%Delete outliers:
outlier = 1;
while outlier
    tmp = b(~isnan(b));
    meanval = mean(tmp);
    maxval = tmp(find(abs(tmp-mean(tmp))==max(abs(tmp-mean(tmp))))); %#ok<FNDSB>
    maxval = maxval(1);
    sdval = std(tmp);
    tn = abs((maxval-meanval)/sdval);
```

Figure 5.6 Code Segment of Outliers Test Algorithm.

The separation of data for this test is performed according to a significance level that defines the limit of normal set of data, which is referred to as α. In Grubb test, the significance level used to decide if an observation is an outlier. If the probability of the observation is less α, then this observation is considered an outlier. Normally the recommended value of significance level for Grubb test is 0.05 with a range of possible values 0.001 to 0.2. In our implantation of the outliers, we found best results could be obtained at α = 0.005, as this value gives us best separation between normal network activities and malicious ones. At this point, we will have two separated sets of data; one contains the normal data and the other the abnormal.

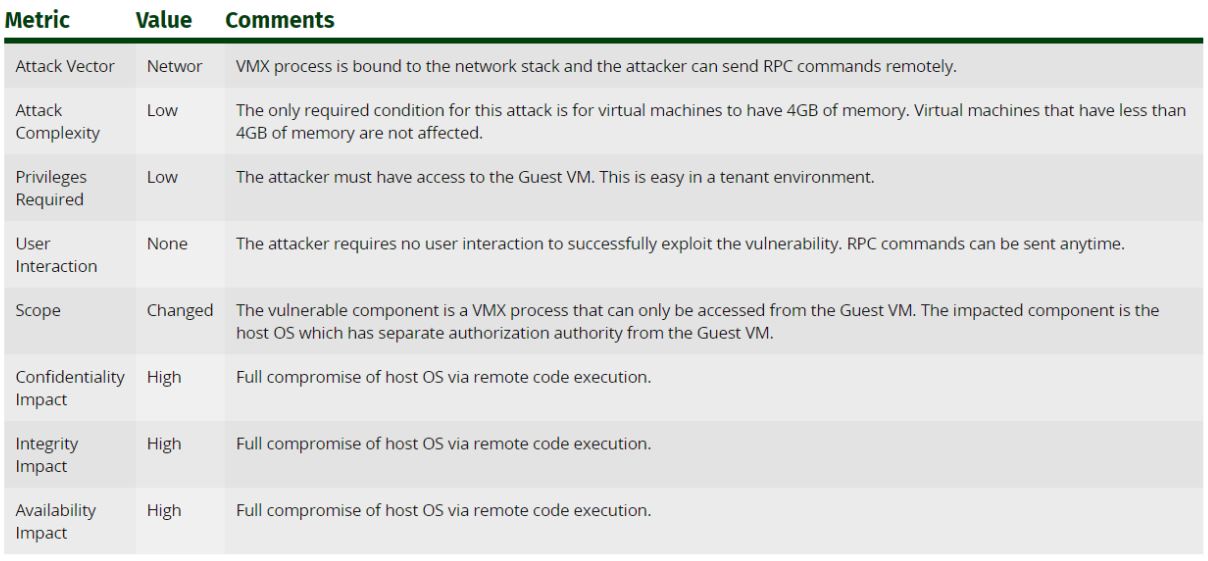
In the thresholding, the classical method investigated in the literature is to test normal data then calculate the standard deviation σ, and threshold then will be a quantity multiply σ such as 3σ. We proposed our threshold mechanism so that instead of depending on normal sample data tested prior to the work, rather we suggest applying the standard deviation on the resulting set after applying the outliers test. In this mechanism, we ensure that our calculations for the thresholds adapt dynamically according to the current data set, which is adjusted to a network situation. This means a different threshold is calculated for every time window.

After calculating the threshold, we check how far the outliers are from the normal data, for that reason, we use the Euclidian distance to measure the distance between the outliers and the centre of the normal data set. If the resulting distance is bigger than 3σ, then this outlier point is a scanner, otherwise it is benign traffic.

5.2.3 Knowledge Base

In this module, we created two databases that represent the sources of information to rely on in the process of decision-making. The first database will contain the benign IPs that contacted or scanned the system such as external IPs used by the security team to investigate the system. The role of this database could be further developed to contain useful information about the traffic such as sources, destinations, volumes, etc. this however, will need Artificial Intelligence algorithms that are beyond the scope of our thesis. The second database will contain information about the vulnerabilities that are found in the system, where this information is collected from specialized public sources. The vulnerability scanning listed the vulnerabilities that existed in the system in the vulnerability database, a detailed information about each vulnerability collected from the public vulnerability databases. For the implementation purpose, we build these databases using MS Access 2013, which is easy to use and suites our system setup.

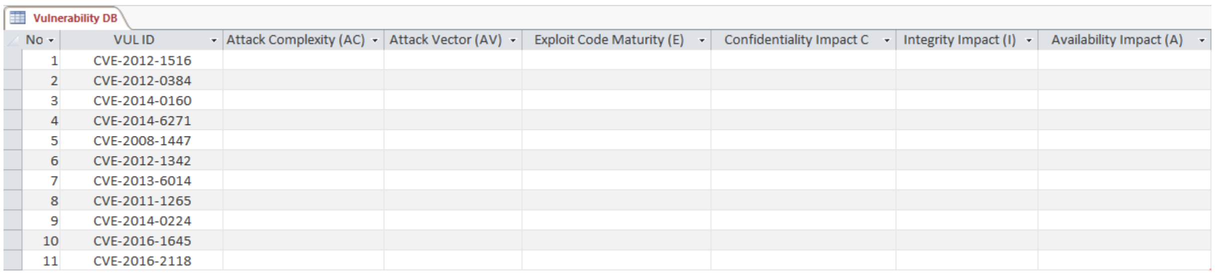
Although, the previous table displayed a group of vulnerabilities that might occur in the system with descriptive information and potential usage (attacks), however, more detailed information is fundamental to analyse the system weaknesses to the level that sufficient to calculate the risk factors and the vulnerabilities metrics. Such information produces in the form shown in the figure 5.7.



Metric	Value	Comments
Attack Vector	Networ	VMX process is bound to the network stack and the attacker can send RPC commands remotely.
Attack Complexity	Low	The only required condition for this attack is for virtual machines to have 4GB of memory. Virtual machines that have less than 4GB of memory are not affected.
Privileges Required	Low	The attacker must have access to the Guest VM. This is easy in a tenant environment.
User Interaction	None	The attacker requires no user interaction to successfully exploit the vulnerability. RPC commands can be sent anytime.
Scope	Changed	The vulnerable component is a VMX process that can only be accessed from the Guest VM. The impacted component is the host OS which has separate authorization authority from the Guest VM.
Confidentiality Impact	High	Full compromise of host OS via remote code execution.
Integrity Impact	High	Full compromise of host OS via remote code execution.
Availability Impact	High	Full compromise of host OS via remote code execution.

Figure 5.7 Detailed Information about CVE-2012-1516 Vulnerabilty [256]

After collecting raw and detailed information about the vulnerabilities in the system, the vulnerabilities entered in the database and are ready to be analysed by the vulnerability analyser. Then, the vulnerabilities names and corresponding metrics values are form the vulnerability database as shown the figure 5.8.

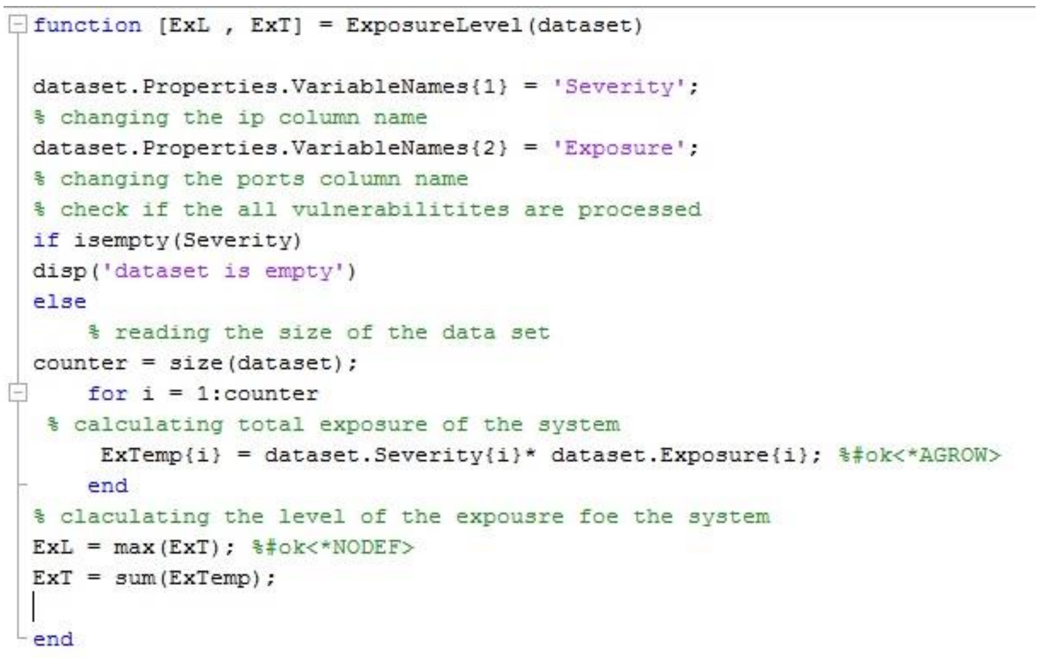

Vulnerability DB

No	VUL ID	Attack Complexity (AC)	Attack Vector (AV)	Exploit Code Maturity (E)	Confidentiality Impact C	Integrity Impact (I)	Availability Impact (A)
1	CVE-2012-1516						
2	CVE-2012-0384						
3	CVE-2014-0160						
4	CVE-2014-6271						
5	CVE-2008-1447						
6	CVE-2012-1342						
7	CVE-2013-6014						
8	CVE-2011-1265						
9	CVE-2014-0224						
10	CVE-2016-1645						
11	CVE-2016-2118						

Figure 5.8 Vulnerability Database

5.2.4 Vulnerabilities Analyser

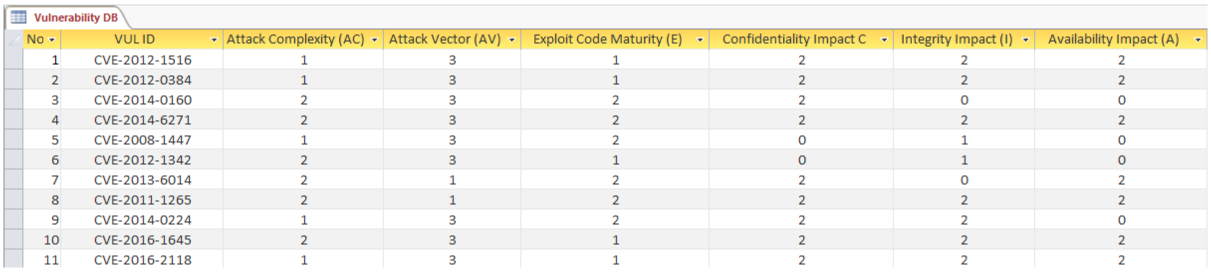
This module is designed to evaluate the vulnerabilities and assign their scores according to the proposed method described in the design chapter. First, this module retrieves the collected raw information from the public vulnerability databases, analyse these information to assign metric values for each identified vulnerability, and calculates the severity and exploitability values. The resulting values after calculation is directed in two directions, the first the vulnerability database where the related information should be stored and the second fed to the prediction module to produces predictions according to the current system and network situation. Figure 5.9 shows a segment of the vulnerability analyser implementation as following:



```
function [ExL , ExT] = ExposureLevel(dataset)

dataset.Properties.VariableNames{1} = 'Severity';
% changing the ip column name
dataset.Properties.VariableNames{2} = 'Exposure';
% changing the ports column name
% check if the all vulnerabilitites are processed
if isempty(Severity)
disp('dataset is empty')
else
    % reading the size of the data set
counter = size(dataset);
    for i = 1:counter
 % calculating total exposure of the system
     ExTemp{i} = dataset.Severity{i}* dataset.Exposure{i}; %#ok<*AGROW>
    end
% claculating the level of the expousre foe the system
ExL = max(ExT); %#ok<*NODEF>
ExT = sum(ExTemp);

end
```

Figure 5.9 The implementation of vulnerabilities analyser.

The calculation the severity and exploitability values produces the exposure level that define the amount each vulnerability affect the system, as mentioned in sections 4.6.1 and 4.6.2 respectively. However, the severity and exploitability metrics has been given a qualitative values in the design phase, but to calculate an exact value of the exposure level, these metrics will assigned to numeric values that reflect the equivalent qualitative values. After the collection process is completed, the process of analysing the vulnerabilities is begins to assign a numeric values for the selected metrics and calculate the exposure level variables. Then the results will fills the database as shown in figure 5.10.



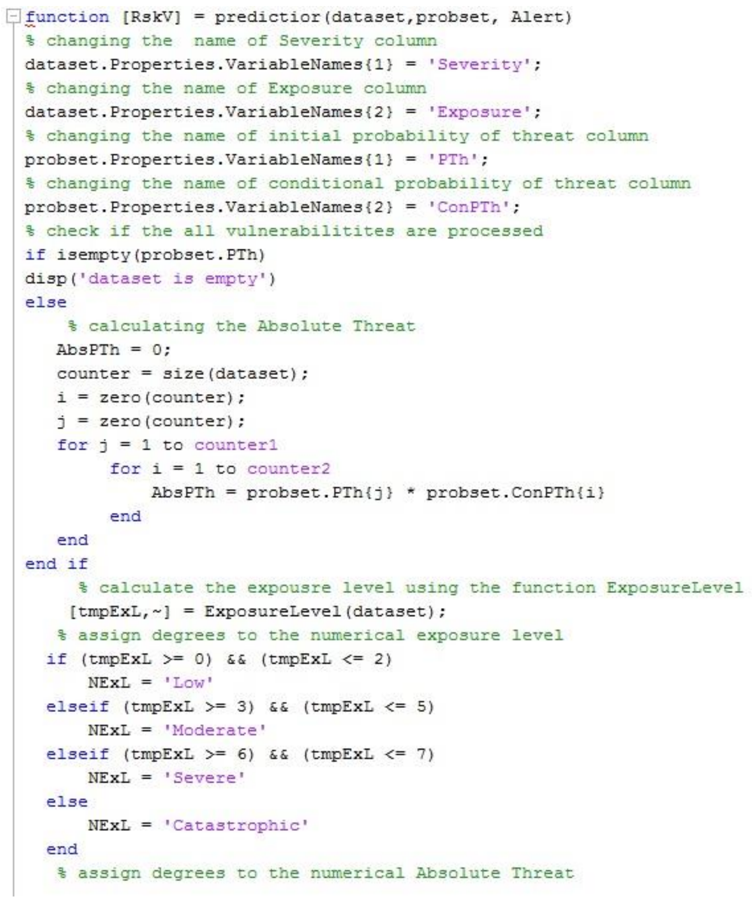
Vulnerability DB

No	VUL ID	Attack Complexity (AC)	Attack Vector (AV)	Exploit Code Maturity (E)	Confidentiality Impact C	Integrity Impact (I)	Availability Impact (A)
1	CVE-2012-1516	1	3	1	2	2	2
2	CVE-2012-0384	1	3	1	2	2	2
3	CVE-2014-0160	2	3	2	2	0	0
4	CVE-2014-6271	2	3	2	2	2	2
5	CVE-2008-1447	1	3	2	0	1	0
6	CVE-2012-1342	2	3	1	0	1	0
7	CVE-2013-6014	2	1	2	2	0	2
8	CVE-2011-1265	2	1	2	2	2	2
9	CVE-2014-0224	1	3	2	2	2	0
10	CVE-2016-1645	2	3	1	2	2	2
11	CVE-2016-2118	1	3	1	2	2	2

Figure 5.10. Analysed Vulnerabilities in the Database.

5.2.5 Prediction Module

The prediction module proposed in chapter 4 is meant to be the central module that is responsible for decision-making about security incidents and raising the security level of the system. As shown in Figure 5.11, this module receives the vulnerability assessment from the vulnerability analyser by calling the module function ‘ExposureLevel’. Then calculating the total threat using the function ‘AbsPTh’ that also gives the threat probability from 0 to 1. Based on the latter two functions, the risk matrix is constructed and the risk level is produced. This module predicts the threat level according to the type of traffic and alters the level of threat accordingly using the final prediction function.



```
function [RskV] = predictior(dataset,probset, Alert)
% changing the  name of Severity column
dataset.Properties.VariableNames{1} = 'Severity';
% changing the name of Exposure column
dataset.Properties.VariableNames{2} = 'Exposure';
% changing the name of initial probability of threat column
probset.Properties.VariableNames{1} = 'PTh';
% changing the name of conditional probability of threat column
probset.Properties.VariableNames{2} = 'ConPTh';
% check if the all vulnerabilitites are processed
if isempty(probset.PTh)
disp('dataset is empty')
else
    % calculating the Absolute Threat
   AbsPTh = 0;
   counter = size(dataset);
   i = zero(counter);
   j = zero(counter);
   for j = 1 to counter1
        for i = 1 to counter2
            AbsPTh = probset.PTh{j} * probset.ConPTh{i}
        end
   end
end if
     % calculate the expousre level using the function ExposureLevel
    [tmpExL,~] = ExposureLevel(dataset);
   % assign degrees to the numerical exposure level
  if (tmpExL >= 0) && (tmpExL <= 2)
      NExL = 'Low'
  elseif (tmpExL >= 3) && (tmpExL <= 5)
      NExL = 'Moderate'
  elseif (tmpExL >= 6) && (tmpExL <= 7)
      NExL = 'Severe'
  else
      NExL = 'Catastrophic'
  end
   % assign degrees to the numerical Absolute Threat
```

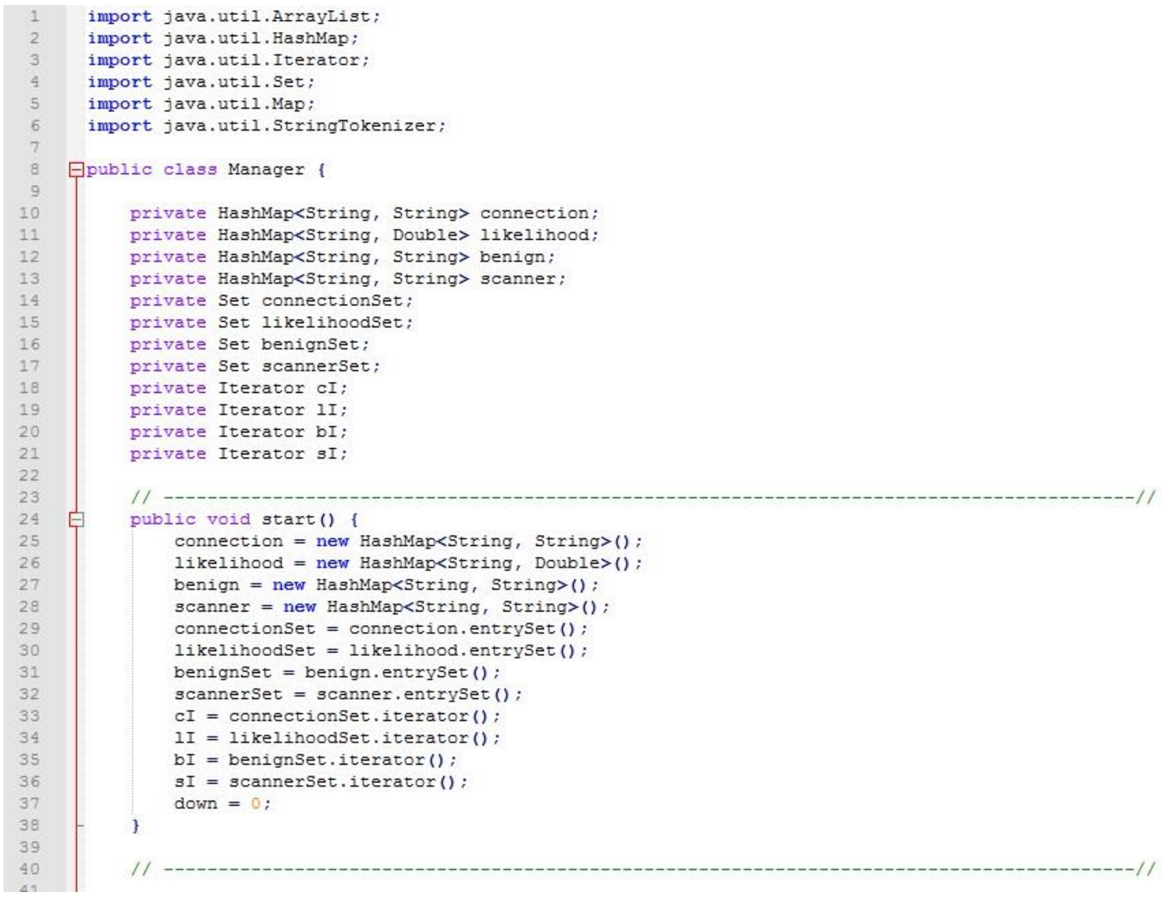
Figure 5.11 The Implementation of Prediction Module.

5.3 Regenerated Methods

In our quest to propose a method for scan detection, we investigated different statistical and probabilistic methods. These methods have been examined and evaluated in order to come up with a novel one that we think will be most appropriate to the cloud-computing environment and network-based attack. In this section, we will display the implementation of the regenerated methods that comprise our explorative study of port scan detection.

5.3.1 Threshold Random Walk (TRW)

As mentioned in chapter 4, we started with this method as it is the gold standard for port scan detection [215] and most solutions for scan detection compared their efficiency to the TRW. We uses the same simulation environment that we used to implement our solution. Due to the limitation in some programming aspects, we incorporated code from Java to implement dynamic codes as shown in figure 5.12.



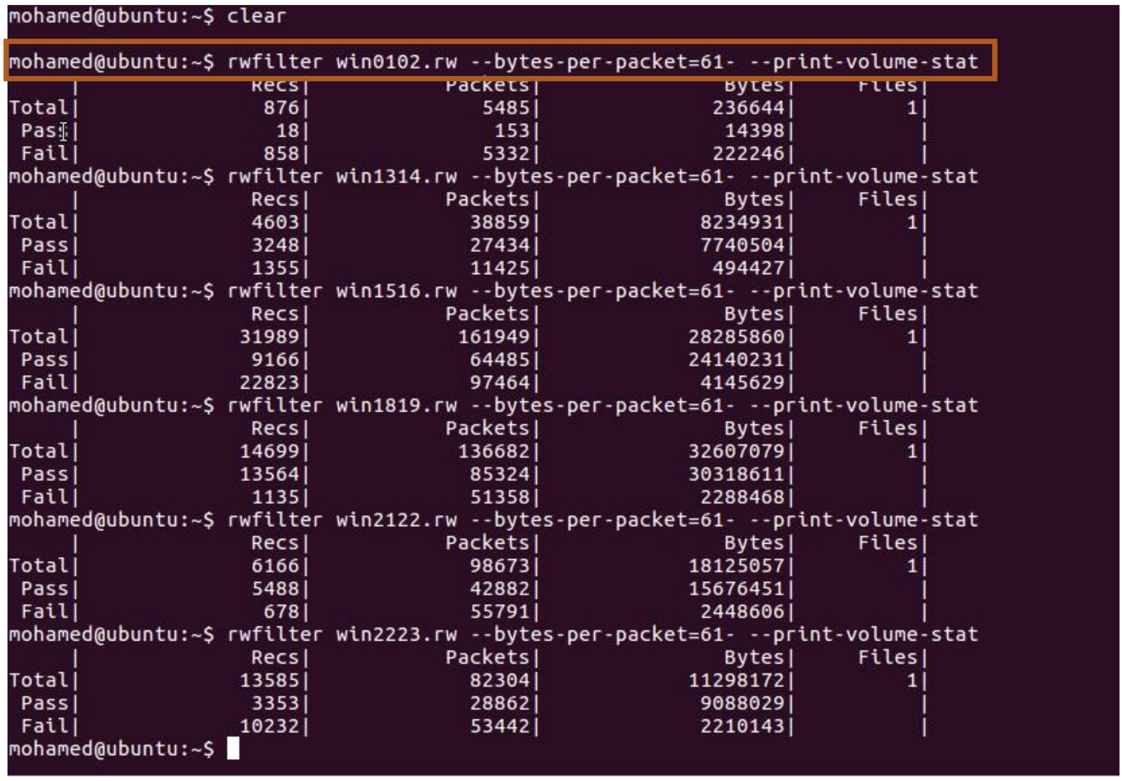
```java
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Set;
import java.util.Map;
import java.util.StringTokenizer;

public class Manager {

    private HashMap<String, String> connection;
    private HashMap<String, Double> likelihood;
    private HashMap<String, String> benign;
    private HashMap<String, String> scanner;
    private Set connectionSet;
    private Set likelihoodSet;
    private Set benignSet;
    private Set scannerSet;
    private Iterator cI;
    private Iterator lI;
    private Iterator bI;
    private Iterator sI;

    // ------------------------------------------------------------------------------------------//
    public void start() {
        connection = new HashMap<String, String>();
        likelihood = new HashMap<String, Double>();
        benign = new HashMap<String, String>();
        scanner = new HashMap<String, String>();
        connectionSet = connection.entrySet();
        likelihoodSet = likelihood.entrySet();
        benignSet = benign.entrySet();
        scannerSet = scanner.entrySet();
        cI = connectionSet.iterator();
        lI = likelihoodSet.iterator();
        bI = benignSet.iterator();
        sI = scannerSet.iterator();
        down = 0;
    }

    // ------------------------------------------------------------------------------------------//
```

Figure 5.12. Java Code Segment from TRW.

We applied the TRW on the same dataset that was divided into time window slices, where we used a time window of about 50 minutes. We used the same set-up of the initial values in the algorithm, where $\theta_0 = 0.8$, $\theta_1 = 0.2$, $\alpha = 0.01$, and $\beta = 0.99$.

5.3.2 Probabilistic Methods

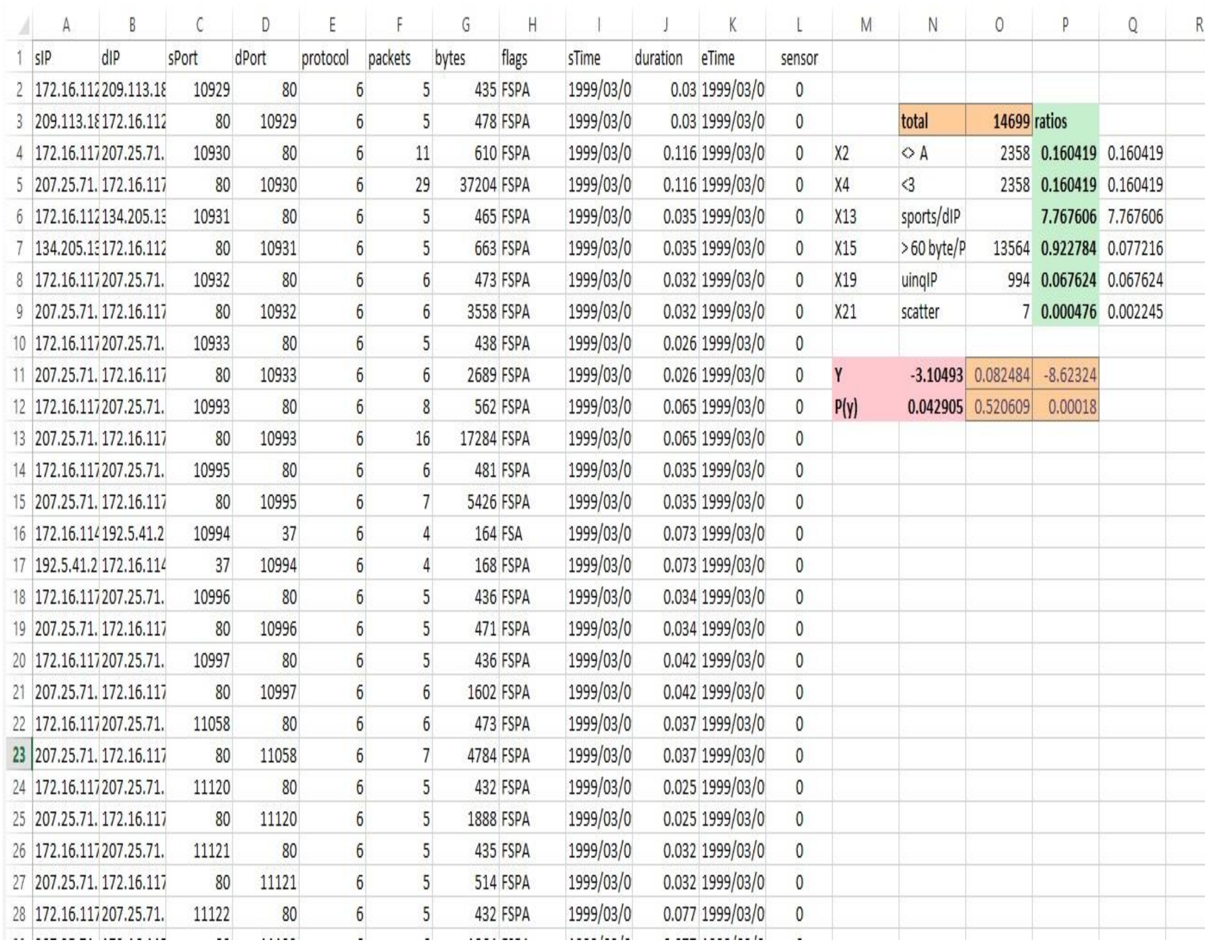
For these methods, we chose to implemented the proposed work of Gates [215] as a representative of probabilistic approaches. This is because their solution is based on a proposed analytical suite that monitors the traffic and analyses different attributes of incoming data. This method was developed according to a specific configuration of traffic that is called a ‘data flow’, where these flows are unidirectional streams. Unlike these flows, our datasets have been collected on configurations of bidirectional data streams; therefore, we modified the dataset from unidirectional to bidirectional using a utility in the suite to convert unidirectional flows. The suite is SiLK [257] that is a collection of traffic analysis tools mainly implemented in C, Perl, or Python.



```
mohamed@ubuntu:~$ clear

mohamed@ubuntu:~$ rwfilter win0102.rw --bytes-per-packet=61- --print-volume-stat
     |              Recs|            Packets|                Bytes|     Files|
Total|               876|               5485|               236644|         1|
 Pass|                18|                153|                14398|          |
 Fail|               858|               5332|               222246|          |
mohamed@ubuntu:~$ rwfilter win1314.rw --bytes-per-packet=61- --print-volume-stat
     |              Recs|            Packets|                Bytes|     Files|
Total|              4603|              38859|              8234931|         1|
 Pass|              3248|              27434|              7740504|          |
 Fail|              1355|              11425|               494427|          |
mohamed@ubuntu:~$ rwfilter win1516.rw --bytes-per-packet=61- --print-volume-stat
     |              Recs|            Packets|                Bytes|     Files|
Total|             31989|             161949|             28285860|         1|
 Pass|              9166|              64485|             24140231|          |
 Fail|             22823|              97464|              4145629|          |
mohamed@ubuntu:~$ rwfilter win1819.rw --bytes-per-packet=61- --print-volume-stat
     |              Recs|            Packets|                Bytes|     Files|
Total|             14699|             136682|             32607079|         1|
 Pass|             13564|              85324|             30318611|          |
 Fail|              1135|              51358|              2288468|          |
mohamed@ubuntu:~$ rwfilter win2122.rw --bytes-per-packet=61- --print-volume-stat
     |              Recs|            Packets|                Bytes|     Files|
Total|              6166|              98673|             18125057|         1|
 Pass|              5488|              42882|             15676451|          |
 Fail|               678|              55791|              2448606|          |
mohamed@ubuntu:~$ rwfilter win2223.rw --bytes-per-packet=61- --print-volume-stat
     |              Recs|            Packets|                Bytes|     Files|
Total|             13585|              82304|             11298172|         1|
 Pass|              3353|              28862|              9088029|          |
 Fail|             10232|              53442|              2210143|          |
mohamed@ubuntu:~$
```

Figure 5.13 Analysing Experiment Dataset Using SiLK

As SiLK is a command line suite, we installed it on Linux Ubuntu OS and analysed our dataset using a single direct instruction at a time as shown in Figure 5.13. After completing analysing all the requested attributes, we exported the results to MS Excel to perform the calculations of the probability of port scan. The probability of scan according to the researchers was impacted mostly by six variables with weights for each variable as shown in figure 5.14.



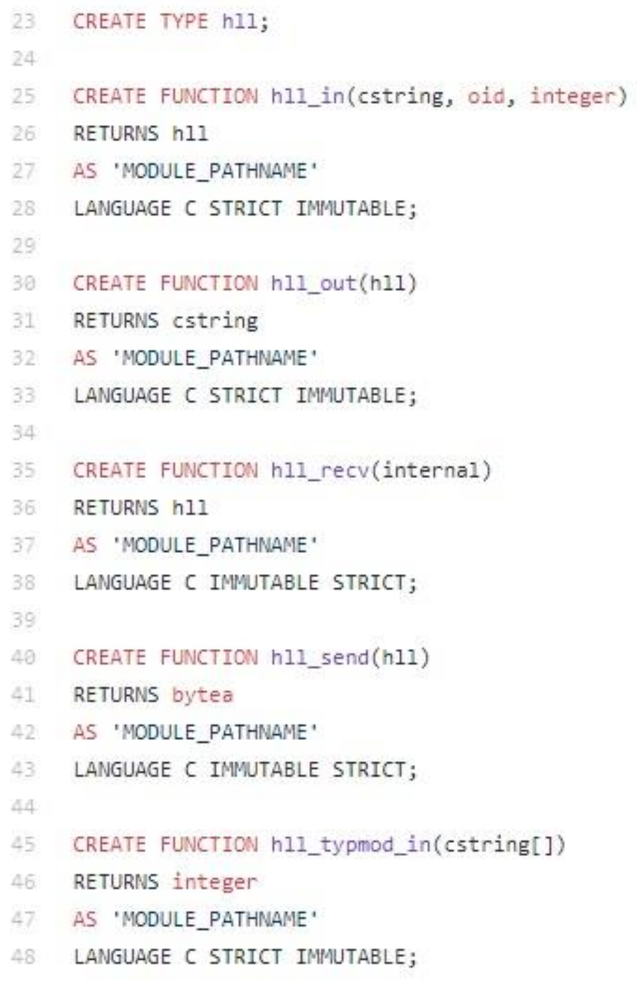
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	sIP	dIP	sPort	dPort	protocol	packets	bytes	flags	sTime	duration	eTime	sensor						
2	172.16.112	209.113.18	10929	80	6	5	435	FSPA	1999/03/0	0.03	1999/03/0	0						
3	209.113.18	172.16.112	80	10929	6	5	478	FSPA	1999/03/0	0.03	1999/03/0	0		total	14699	ratios		
4	172.16.117	207.25.71.	10930	80	6	11	610	FSPA	1999/03/0	0.116	1999/03/0	0	X2	<> A	2358	**0.160419**	0.160419	
5	207.25.71.	172.16.117	80	10930	6	29	37204	FSPA	1999/03/0	0.116	1999/03/0	0	X4	<3	2358	**0.160419**	0.160419	
6	172.16.112	134.205.13	10931	80	6	5	465	FSPA	1999/03/0	0.035	1999/03/0	0	X13	sports/dIP		**7.767606**	7.767606	
7	134.205.13	172.16.112	80	10931	6	5	663	FSPA	1999/03/0	0.035	1999/03/0	0	X15	> 60 byte/P	13564	**0.922784**	0.077216	
8	172.16.117	207.25.71.	10932	80	6	6	473	FSPA	1999/03/0	0.032	1999/03/0	0	X19	uinqIP	994	**0.067624**	0.067624	
9	207.25.71.	172.16.117	80	10932	6	6	3558	FSPA	1999/03/0	0.032	1999/03/0	0	X21	scatter	7	**0.000476**	0.002245	
10	172.16.117	207.25.71.	10933	80	6	5	438	FSPA	1999/03/0	0.026	1999/03/0	0						
11	207.25.71.	172.16.117	80	10933	6	6	2689	FSPA	1999/03/0	0.026	1999/03/0	0	**Y**	**-3.10493**	0.082484	-8.62324		
12	172.16.117	207.25.71.	10993	80	6	8	562	FSPA	1999/03/0	0.065	1999/03/0	0	**P(y)**	**0.042905**	0.520609	0.00018		
13	207.25.71.	172.16.117	80	10993	6	16	17284	FSPA	1999/03/0	0.065	1999/03/0	0						
14	172.16.117	207.25.71.	10995	80	6	6	481	FSPA	1999/03/0	0.035	1999/03/0	0						
15	207.25.71.	172.16.117	80	10995	6	7	5426	FSPA	1999/03/0	0.035	1999/03/0	0						
16	172.16.114	192.5.41.2	10994	37	6	4	164	FSA	1999/03/0	0.073	1999/03/0	0						
17	192.5.41.2	172.16.114	37	10994	6	4	168	FSPA	1999/03/0	0.073	1999/03/0	0						
18	172.16.117	207.25.71.	10996	80	6	5	436	FSPA	1999/03/0	0.034	1999/03/0	0						
19	207.25.71.	172.16.117	80	10996	6	5	471	FSPA	1999/03/0	0.034	1999/03/0	0						
20	172.16.117	207.25.71.	10997	80	6	5	436	FSPA	1999/03/0	0.042	1999/03/0	0						
21	207.25.71.	172.16.117	80	10997	6	6	1602	FSPA	1999/03/0	0.042	1999/03/0	0						
22	172.16.117	207.25.71.	11058	80	6	6	473	FSPA	1999/03/0	0.037	1999/03/0	0						
23	207.25.71.	172.16.117	80	11058	6	7	4784	FSPA	1999/03/0	0.037	1999/03/0	0						
24	172.16.117	207.25.71.	11120	80	6	5	432	FSPA	1999/03/0	0.025	1999/03/0	0						
25	207.25.71.	172.16.117	80	11120	6	5	1888	FSPA	1999/03/0	0.025	1999/03/0	0						
26	172.16.117	207.25.71.	11121	80	6	5	435	FSPA	1999/03/0	0.032	1999/03/0	0						
27	207.25.71.	172.16.117	80	11121	6	5	514	FSPA	1999/03/0	0.032	1999/03/0	0						
28	172.16.117	207.25.71.	11122	80	6	5	432	FSPA	1999/03/0	0.077	1999/03/0	0						

Figure 5.14 Excel Calculations of Probability of Port Scan

In the previous figure, we see the results of the network sniffer that collects information of the network traffic in both direction; send and receive. This shown in figure 5.14 from column A to column L. then in column M represents the title of the method variables and column O contains the values of these variables. The main collective variable Y and its value highlighted in red, and the P(Y) represents probability of the port scan that calculated after applying Logistic Regression of the value of Y, and highlighted in red as well.

5.3.3 Statistical Methods

The intended meaning of the statistical detection is an approach that depends entirely on statistical methods for measuring and judging malicious activities. For this purpose, we chose to rebuild the work of Chabchoub et al [258] as a sample of these methods that are lightweight computationally and efficient in performance, which suits cloud computing and big data environments. The first component of their solution exploits the HyperLogLog algorithm for estimating the number of unique ports (as counting mechanism) in the IP traffic, a code extracted from the algorithm implementation shown in figure 5.15.



```
CREATE TYPE hll;

CREATE FUNCTION hll_in(cstring, oid, integer)
RETURNS hll
AS 'MODULE_PATHNAME'
LANGUAGE C STRICT IMMUTABLE;

CREATE FUNCTION hll_out(hll)
RETURNS cstring
AS 'MODULE_PATHNAME'
LANGUAGE C STRICT IMMUTABLE;

CREATE FUNCTION hll_recv(internal)
RETURNS hll
AS 'MODULE_PATHNAME'
LANGUAGE C IMMUTABLE STRICT;

CREATE FUNCTION hll_send(hll)
RETURNS bytea
AS 'MODULE_PATHNAME'
LANGUAGE C IMMUTABLE STRICT;

CREATE FUNCTION hll_typmod_in(cstring[])
RETURNS integer
AS 'MODULE_PATHNAME'
LANGUAGE C STRICT IMMUTABLE;
```

Figure 5.15 Code of HyperLogLog Implementation

Then the solution detects the intrusive contacts using the Exponentially Weighted Moving Average (EWMA), with a time window of 5 minutes. A source is considered malicious if the EWMA of the incoming number of unique ports exceeds the predefined threshold.

5.4 Summary

The chapter starts with the research initial testbed and traffic analysis that enable us to examine the traffic and observe the changes and effects that poses in the traffic before and during the attack. We focus on the changes in the traffic that happens before the real attacks, which called the reconnaissance stage. Therefore, our analysis results in verdict to monitor scan activities as attack indicator.

This chapter then demonstrate the implementation of the framework that proposed in Chapter 4 and all included modules. Starting with the data collection, the dataset used, and the port scan and IP scans included. Then a demonstration of the scan module and the included algorithms that used to sense the scan activities. Then, the Knowledge base implementation and the collected data, in addition to databases used in this module. The analyser module implementation is also presented and the way of analysing the system vulnerabilities. Finally, this chapter provide the prediction module implementation, which is the central module that sums all the work achieved by the framework and presented it to security administration.

The last part of this chapter provides an overview of the implementation of the comparable methods that regenerated to be used to measure our solution performance. These methods included the hypothesis testing, probabilistic, and statistical approaches.

Chapter 6

Evaluation

Our framework works through two independent parts, namely; the vulnerability part and network part. The vulnerabilities part is standardized, which means calculating the impact on the system security will not differ drastically from one approach to another, as the severity and importance of vulnerability will be evaluated similarly in different solutions. Therefore, a great part of this research focus is toward the impact of sensing network situation on the system evaluation process. This chapter provides evaluation of the proposed system.

6.1 Dynamic selective scan detection (SASD) evaluation

In this part, we present details of the evaluation of the scan detection method that proposed in our solution. We test our method against the selected dataset and observe its ability to detect the two types of surveillance as follows:

6.1.2. A Port Scanning Detection:

In this type of scanning, the attacker sends connection requests to many ports in the same machine (IP destination) this means a high increment of connection requesting packets during a short time window. As port scanning mainly depends on TCP/IP protocol, where every packet should have a sequence number from 0 to 4294967296, in this protocol connection requesting packet is labelled with 0 or 1. Thus, in the port scan detection we count the increment in the packets with sequence 0 or sequence 1. Three port scanning has been launched on two different destinations at different times as clarified in table 6.1.

Table 6.1 Labelled Port Scans Attacks.

No	**Date**	**Start_Time**	**Destination**
1	03/09/1999	08:44:17	marx.eyrie.af.mil
2	03/11/1999	10:50:11	marx.eyrie.af.mil
3	03/12/1999	17:13:10	pascal.eyrie.af.mil

All these attacks have been successfully identified by our algorithm and recognized as attacks. The algorithm first counts the connection requesting packets and highlights the source IPs that have sent high numbers of requests by detecting outliers as shown in figures 6.1-3. The reliance on connection requesting packets has made the scan related packets more obvious in the traffic statistics that enable the decision of scanning to be much easier.

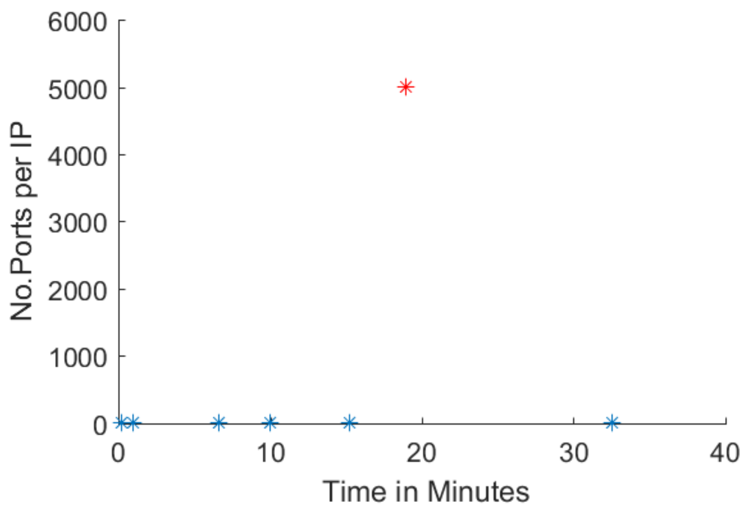


Figure 6.1 Port Scan Attack No.1 Identification.

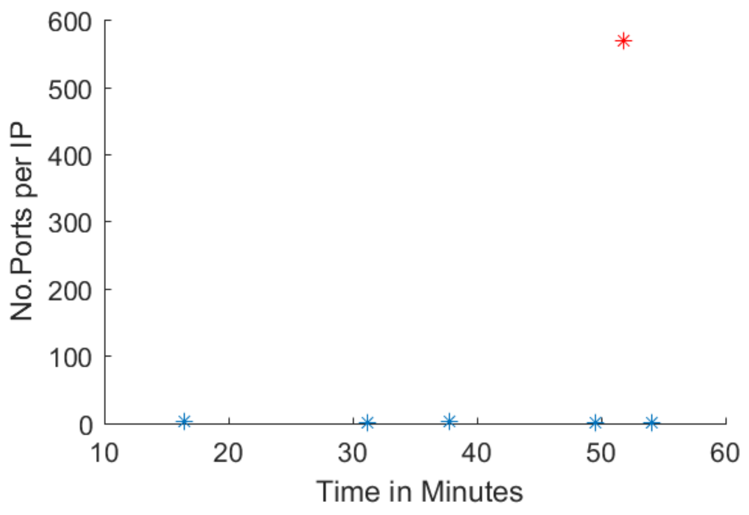


Figure 6.2 Port Scan Attack 2 Identification.

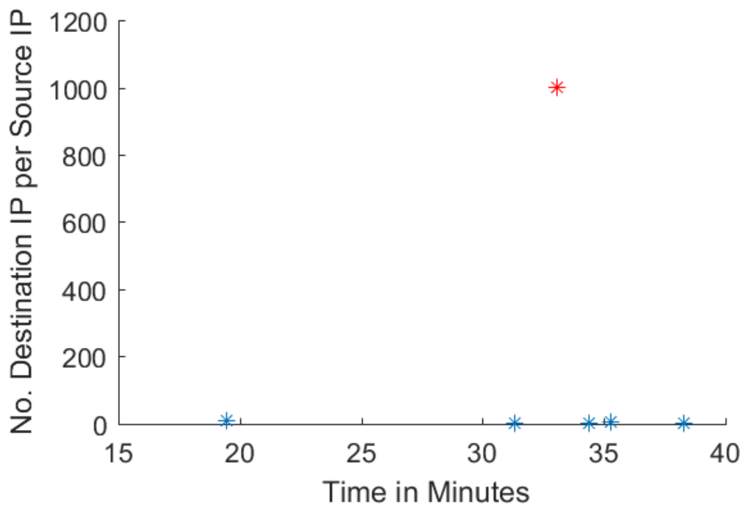


Figure 6.3 Port Scan Attack 3 Identification.

The previous figures show the distinction between the number of connections requested by a potential attacker and the connections requested by normal sources. Despite the previous three figure has shown clearly distinguishing results of specific traffic packets examination (marked in red colour), however, this is alone is not sufficient to make a decision of intrusion as no detection threshold involved. Therefore, the algorithm continues to check how far this abnormal measurement is from the accepted limits represented by the detection threshold.

6.1.2.B IP Sweeping Detection

In this type of scanning as mentioned earlier, the attacker sends connection requests to many machines (IPs) in the same the network or domain. IP scanning is a surveillance technique that scans a network or a domain and returns the live machines (IPs). This process performed by sending an ICMP ECHO request to all machines in the network, and the live machine will respond by sending ICMP ECHO reply. Therefore, to detect this type of scan we focus on counting ICMP ECHO requests during a specific time window. DARPA dataset contains three IP scanning that has been launched at different times as clarified in table 6.2.

Table 6.2 Labelled IP Scanning Activities.

ID	Date	Start Time	Target	Score	Name
14	03/09/1999	13:05:10	172.016.112. 001-114.254	1	ipsweep
20	03/10/1999	20:17:10	172.016.112. 001-114.254	1	ipsweep
30	03/11/1999	16:36:10	172.016.112. 001-254	1	ipsweep

The same technique used to identify port scans previously, has been used to detect the IP sweeps attacks. Our algorithm has successfully identified the three IP sweeps in the dataset.

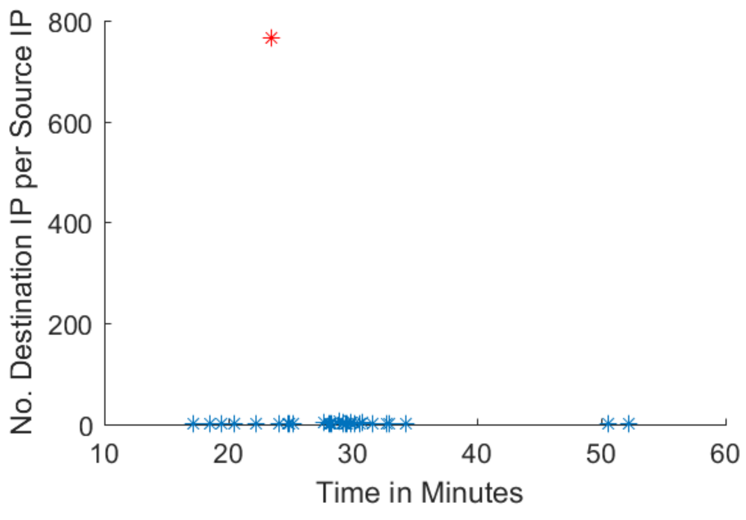


Figure 6.4 IP Sweep Attack 1.

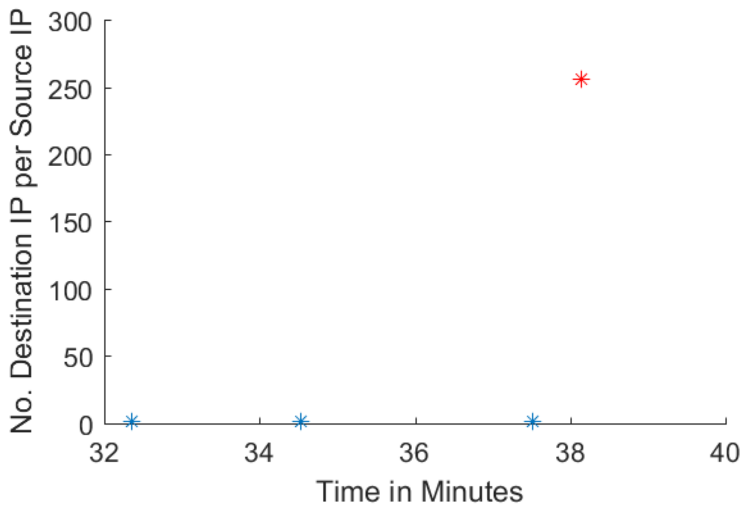


Figure 6.5 IP Sweep Attack 2.

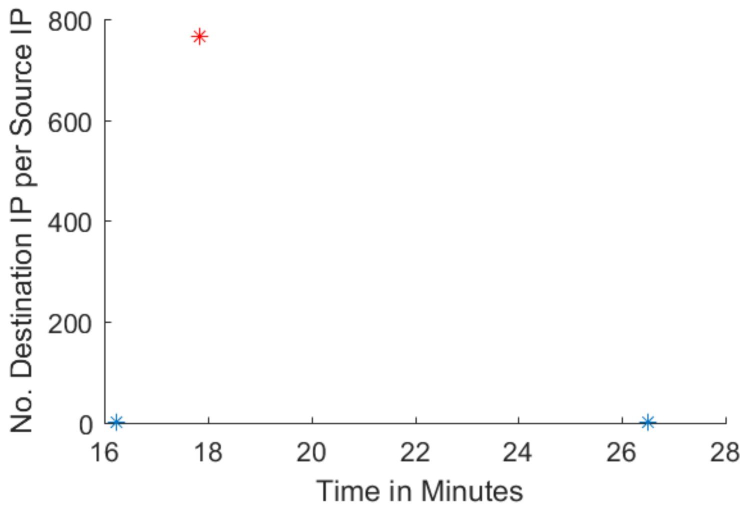


Figure 6.6 IP Sweep Attack 3.

In figures 6.48-50, the number of ICMP ECHO requests sent from the same source IP is represented by a star in the graph in accordance with the average time it arrived. The graphs also show visually the number of echo requests that the normal IP sent and the abnormal number of requests.

6.2 Comparisons to Other Methods

Although SASD has been able to identify all IP sweep and port sweep attacks that are labelled in the dataset, however, in order to evaluate the detection capability of our algorithm we tried to rebuild some other solutions that depend on statistical and probabilistic methods. SASD is related to statistical methods as it uses statistical methods in cardinality estimation, outlier detection and computing the threshold. On the other hand, SASD is related to probabilistic methods such as the Hyperloglog algorithm that use randomness and probability in the hashing process. Therefore, we will compare SASD to probabilistic and statistical methods. We apply the rebuilt solutions on the same dataset to observe the results of detection of the same IP/Port sweep attacks and compare with the results of our solution. In this section, we will demonstrate the rebuilt solutions and the results as follows:

6.2.1 Threshold Random Walk (TRW)

We choose to regenerate this method as it represents the gold standard for port-scan detection methods [215]. The researchers proposed online detection algorithm [21] to fulfil the requirements of port scan detection namely the promptness and accuracy in detecting malicious scanners. The method is developed based on the theory of sequential hypothesis testing by applying this theory to the access patterns to the local hosts that modelled as a random walk (RW). The by analysing the traces between the source (scanner) and the destination (host), the methods classify the source into malicious and non-malicious depending on the percentage of the failed connection to the local hosts. The failed connections as defined by the researchers are the connection that made to inactive hosts or requests of invalid services on active hosts; as such connections are more likely to be requested by malicious parties. Hypothesis testing method is binary hypothesis testing that considered only two hypotheses, H_0 that the source is benign and H_1 where the sender is malicious. This process is performed for each individual observation by assigning either H_0 or H_1 for any given connection. The connection is represented by an independent random variable R_i, and the conditional relationship between the random variable and the hypothesis $R_i|H_i$ $i= 1,2,\ldots$ is modelled according to Bernoulli distribution of R_i as following:

$$\Pr[R_i = 0 | H_0] = \theta_0, \quad \Pr[R_i = 1 | H_0] = 1 - \theta_0$$

$$\Pr[R_i = 0 | H_1] = \theta_1, \quad \Pr[R_i = 1 | H_1] = 1 - \theta_1 \tag{16}$$

A source is considered 'benign' if fulfil the following condition:

$$\theta_0 > \theta_1$$

As there are only two hypotheses, there are four results produces by the hypotheses testing. When H_1 hypotheses is true, then the '*detection'* is occurred and the H_0 represents a *'false negative'*. Vice versa, when H_0 is true this result in H_0 represent a '*nominal'* and H_1 represents the '*false positive*'. The detect criteria is specified using the detection probability P_D and the false positive probability P_F against user-selected sensitivity values α and β as following:

$$P_F \leq \alpha \text{ and } P_D \geq \beta \tag{17}$$

As a result, the TRW is an algorithm that depends on observing the variance in frequencies of connections made by benign and frequencies made by malicious hosts. Based on that, the algorithm detect malicious host with very few attempts to connection that make this algorithm is the faster than the predecessor methods.

6.2.2 Probabilistic Methods

In these methods, a statistical analysis is done to the traffic and an observation on specific attributes of traffic and then a probability is assigned. We tried to rebuild the work [215], where the researchers suggested a scan detection approach for very large networks. This solution was proposed under certain constraints and requirements such as the collection of the traffic will be on flow level only, at multiple collection points, at multiple geographical domains, and on base of single direction traffic. The researchers stated that the current scan detection approaches are not working under those conditions. Therefore, they adopted a probabilistic approach to adapt with the specified constraints. Based on that, they first defined the event as a set of flows originated from single IP address to single destination that this set lies between inactivity periods (at least 32 flows preceded and succeeded by 5 minutes of inactivity). Then each event is analysed to determine if contains a scan or not. The scan analysis consists of computing attributes values of unidirectional data flow (the flow contains multiple packets travelling towards a single destination such as Cisco Net Flow setup) and then calculates the probability of the result using Bayesian logistic regression. First, the solution computes a global variable that consists of a summation of the most important variables that are most influential in scan detection as follows:

$$\hat{z} = -2.83835 + 3.30902\, v_1 - 0.15705 v_2 - 0.00232 v_3 - 1.04741 v_4 + 3.16302 v_5 - 3.26027 v_6 \tag{18}$$

Where v_1 the ratio of flows that ACK flag set to 0 to all flows, v_2 the ratio of flows that has two packets or less to all flows, v_3 the average of source ports for each IP destination. v_4 is the ratio of flows that contain packets size 60 bytes or greater to all flows, v_5 is the number of unique destinations to total number of flows. v_6 the ratio of flows that contain backscatter combination to all flows. Although the researcher listed 21 variables that they believed indicate the scan activity, they stated that the last six variables have the most influence on the scan detection. Then the probability is computed by applying the logistic regression on the global variable $\hat{z}$ is as follows:

$$\hat{P}\,(X) = \frac{e^{\hat{z}}}{1+e^{\hat{z}}} \tag{19}$$

Where X is a traffic slice that contains a scan. The result from equation 7 is a probability of 0 to 1 that the tested traffic contains a scan. $\hat{P}(X)$ is then compared to a threshold equal to 0.5, if the value of $\hat{P}(X)$ is greater than the threshold then the traffic contains a scan otherwise the traffic is normal. The researchers have stated that they had a classification accuracy of 98.5% during the training phase, and 99.3% during the test phase.

6.2.3 Statistical Methods

In these methods, a statistical analysis is performed on the traffic and measurements are applied to check the change or deviation in traffic attributes values. We tried to rebuild the work in [258], where the researchers proposed an algorithm to detect port scan in IP traffic using Hyperloglog algorithm and Exponential Weighted Moving Average (EWMA). As mentioned earlier, Hyperloglog is used to estimate online the number of distinct ports in the IP traffic. The researchers adopted a sliding window version of Hyperloglog, and they claimed that this technique does not affect the accuracy of the original algorithm. The counting process is achieved using a window W equal to 60 s, and the sliding leap is every 30 s. While, the EWMA is used to sense the change point when it exceeds an Upper Control Limit (UCL) as threshold, and defined as:

$$\text{UCL} = \text{EWMA}(0) + m * \sqrt{\frac{\lambda}{(2-\lambda)d_0^2}} \tag{20}$$

Where EWMA(0) is the average of the whole dataset, m is an factor whose value is set to 3 or selected from specific calculation in [259], λ is multiplicative factor ($0 < \lambda \leq 1$) and set during implementation at 0.3 , *d* is the standard deviation of the whole dataset. The results approve the performance of Hyperloglog and EWMA in regard to accuracy, memory exploitation, and time response.

6.2.4 Results of Scanning Detection

We run three separate experiments, one for each solution on the same data slice that contains the surveillance or scanning activity. As mentioned earlier, DARPA was used as the dataset and it contains six scanning activities denoted as surveillance. As the whole dataset is collected over three weeks, we only apply the experiments on the data subset that contains what is denoted as surveillance in the DARPA documentation. The probabilistic solution was able to identify only one of the scan attacks, while the statistical solution was able to identify five out of six attacks listed. Finally, our solution was able to identify all the scans successfully. table 6.3 displays the results as following:

Table 6.3. Results of Scan Detection of All Experiments.

Solution	**Port Scan1**	**Port Scan 2**	**Port Scan 3**	**IP Scan 1**	**IP Scan 2**	**IP Scan 3**
TRW	*	*	*	-	-	-
Probabilistic	-	-	*	-	-	-
Statistical	*	*	-	*	*	*
Our solution	*	*	*	*	*	*

- not detected * detected

6.2.5 Detection Performance

The essential characteristics of IPFCC that requires to be supported by robust evidence are the detection capabilities and false alarm rate. In this section, we will evaluate the proposed system ability to detect scanning activities by focusing on the detection accuracy, detection Speed, High Performance, Real-Time and Scalable.

IPFCC is designed to detect the anomalies in the network traffic, focusing on the anomalies caused by the scanning activities. In the conducted experiments, IPFCC`s capability to detect different scanning activities is examined. Varying the type of scan activity and the time of launching the activity enables us to measure the detection, and false positive and false negative rates. Details of the experiments and the results are presented in table 6.4.

Table 6.4 IPFCC Detection Performance.

Attack	**Threshold**	**Malicious value**	**Detection rate**	**False negative rate**	**False positive rate**
IPsweep1	1.7500	764.1863	100 %	0	0
IPsweep2	1.0000	255.4227	100 %	0	0
IPsweep3	1.0000	764.9263	100 %	0	0
Portsweep1	2.1667	5000.8	100 %	0	0
Portsweep2	3.0000	998.3329	100 %	0	0
Portsweep3	1.4000	568.0433	100 %	0	0

IPsweep is a scan activity that is aimed at scouting the whole network and identifying the online hosts. Portsweep is the port scanning that is aimed at identifying open ports on a specific host. Note that these two scans activities names are taken from the original dataset. The results from the experiments verified that IPFCC is able to detect scan activities at a high rate on the test dataset, while preserving the false alarms at very low rates. This fulfils the accuracy criterion in our evaluation. This performance is achieved due to the selective detection algorithm used and dynamic thresholding used to make a decision of intrusion. Choosing a selective detection algorithm means that we focus only on analysing specific data that is directly related to intrusion preparations. This in turn means a huge decrease of the amount of data that the system needs to analyse in order to detect attacks. This reduces the computational and memory resources required to operate the system. In the selective detection, rather than checking all packets information in a session, we focus only on the first packet of that session. This will lead to discarding many packets in the traffic and focusing only initial packets. This concept is based on practice followed by the attackers that they only send a single packet to check the port`s state, which is normally the initial packet of the session. Therefore, we monitor this type of packet by checking only the packets that are labelled with sequence number of zero or one. Using fine-grained information of TCP/IP protocol, we managed to reduce the amount of data needed for experimental computations as shown in figure 6.7.

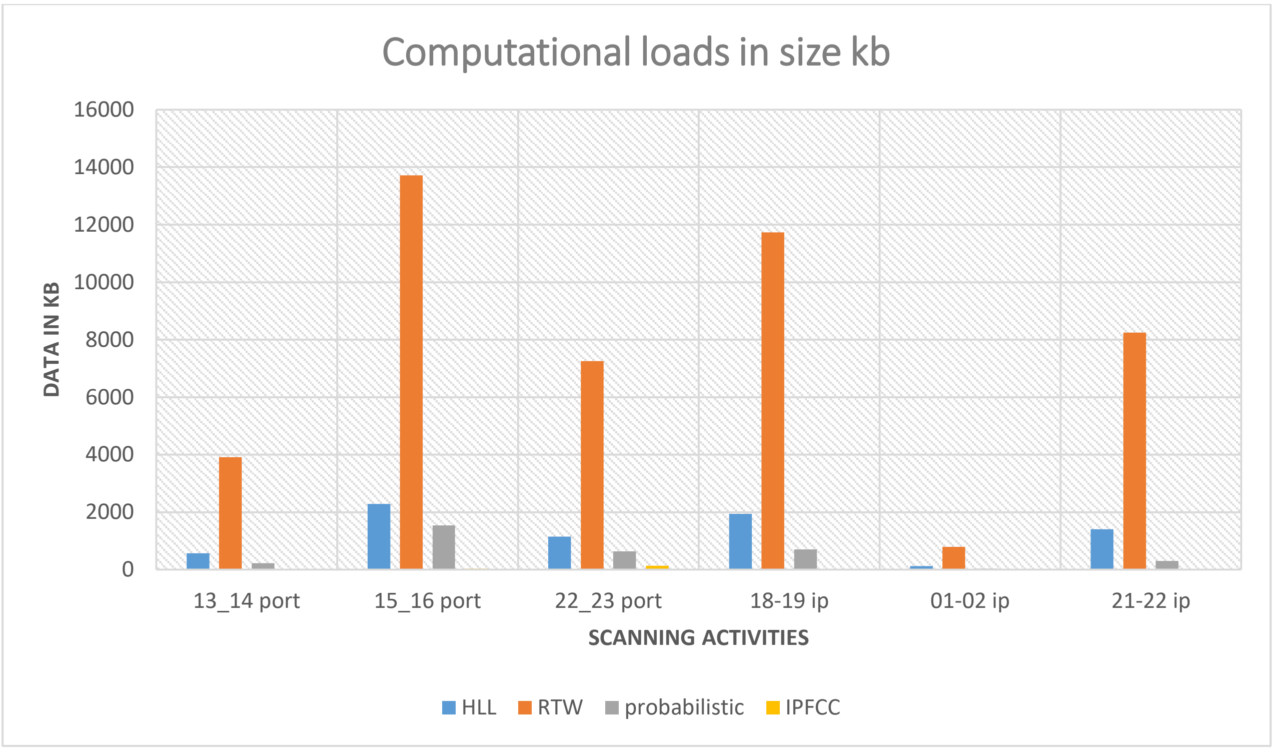


Figure 6.7 Computational Loads in Size Kb.

The selectivity attribute has empowered IPFCC to achieve high performance and outperform existing methods of the tested datasets. Table 6.5 shows the results of the performance experiments.

Table 6.5 Tested/Proposed Methods Detection Performance

Method	**Network situation**	**Detection rate**	**False negative rate**	**False positive rate**
TRW	Port scanning	94 %	6%	2 %
Probabilistic	Port scanning	16 %	84%	1 %

HLL	Port scanning	83 %	13%	20 %
IPFCC	Port & IP scanning	100%	0 %	50%

As shown in the previous table, IPFCC scores the highest detection of the attacks on the tested data, which contains labelled attacks. IPFCC was successful in detecting all the labelled attacks and identified the attacker with no false alarms. On the other hand, IPFCC acted perversely when testing normal data in that half of the samples from the dataset were found to contain scan activities, where they should have been clear. In other words, IPFCC scored a high rate of false positives, where normal instances were (wrongly) considered malicious. Two potential factors could have contributed to this situation; the datasets contained unlabelled benign scan activities, and the threshold might need more calibration to remove these false alarms.

The second criterion of the detection performance of IPFCC is the detection time that is attributed to the use of HLL for the main part of the detection process, which is counting the number of unique connections from remote sources. As mentioned in chapter five, HLL is an estimation algorithm that is well known for using a trivial amount of memory to operate, which makes this algorithm an optimal solution to handle online a huge amount of data at high-speed networks. Because the detection depends on analysing network traffic, the detection time is determined by the length of the time window and time of detection execution. Figure 6.8 display the counters creation and execution for time windows of equal sizes.

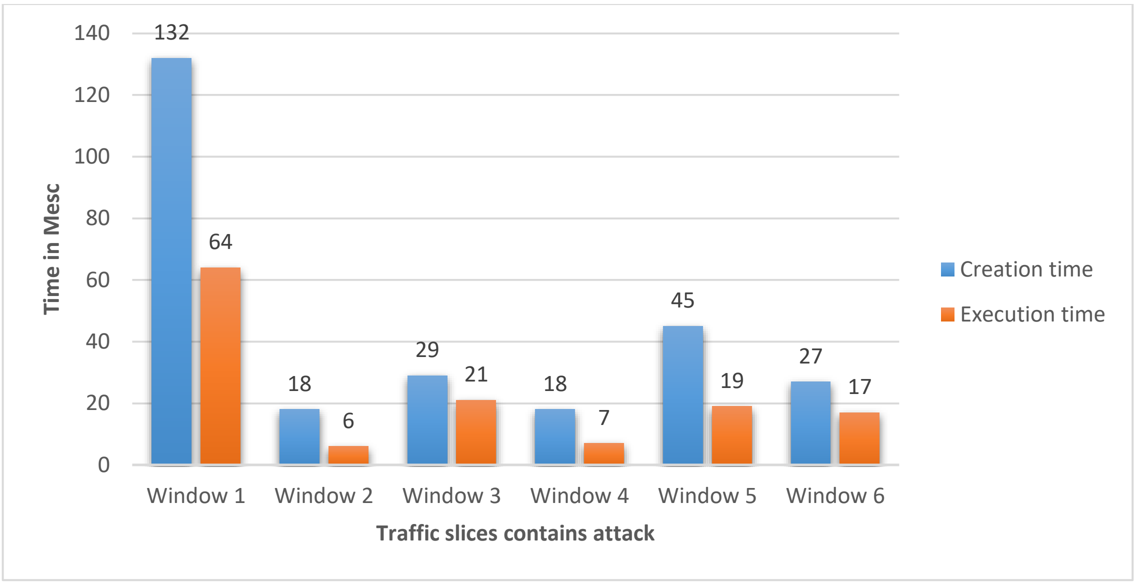


Figure 6.8 Creation and Execution Time for Detection Counters.

Considering the relational stability of detection performance and the type of analysis results shown in figure 6.8 execution time, the High Performance requirement can met. When it comes to the real-time requirement of IPFCC this can be met due to the small amount of data that is presented for analysis, which results from the selective attribute of the detection algorithm. Figure 6.53 showed the amount of each time window that was presented for analysis as follows:

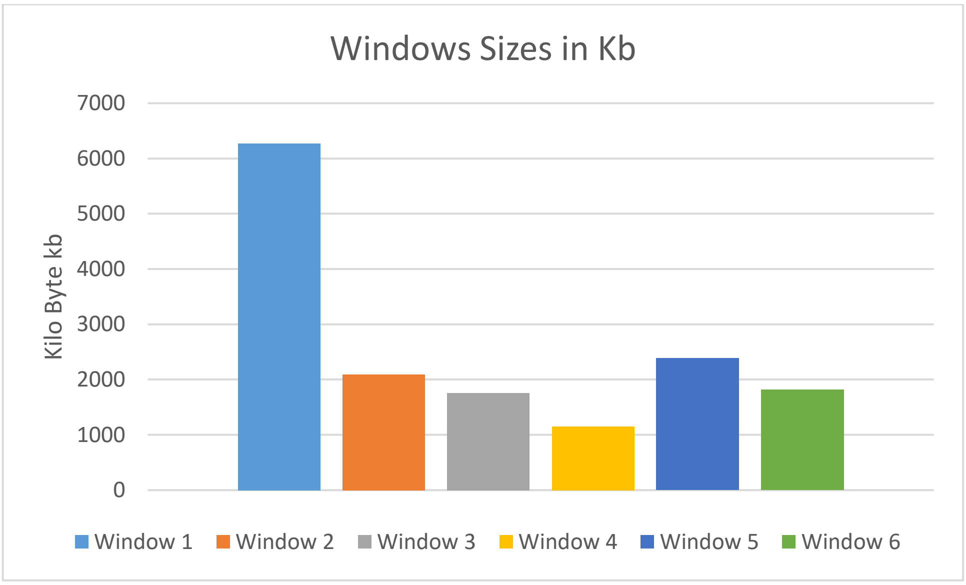


Figure 6.2 Traffic Slices Sizes in Kb

By combining the two previous figures, it shows the scalability of the proposed system; that means the performance has been preserved when analysing different data volumes. It is important to notice that there is a trade-off between the detection time and the window size. The time window should not be too short that the outliers would miss the malicious behaviour, nor too long that would increase the detection time and therefore the response to that malicious activity. Depending on the natural traffic loads, the security administration can calibrate and balance the time window sizes to optimize the detection performance.

6.3 Intrusion Prediction Scenarios

As mentioned in Chapter 4, we will use some terms and specifications from Common Vulnerability Scoring System (CVSS) [33] to calculate the two pillars of the exposure, namely; the Severity and the exploitability, which we incorporate to estimate the Risk. As mentioned in chapter four, for the exploitability we use three factors that we thought most affected the exposure level, which are Attack Complexity (AC), Attack Vector (AV), and Exploit Code Maturity (E). Likewise, we used the following factors as having the most influence on the Severity; Confidentiality Impact (C), Integrity Impact (I), and Availability Impact (A).

In Chapter 4, we stated that these factors take three qualitative values for each variable, and for the sake of the implementation, we will replace these qualitative values with quantitative values as high takes numeric value of 2, low takes the numeric value of 1, and none assigned to 0.

Table 6.6 Severity Attributes Numerical Values

Value / Factor	High	Low	None
Confidentiality Impact (C)	2	1	0
Integrity Impact (I)	2	1	0
Availability Impact (A)	2	1	0

Likewise, we set numerical values for the exposure level factors, as follows:

Table 6.7 Exposure Level Attributes Numeric Values

Value / Factor	High	Low	None
Attack Complexity (AC)	2	1	0
Attack Vector (AV)	3	2	1
Exploit Code Maturity (E)	2	1	1

The method of public databases is to assign values and weights for different vulnerability attributes by analysis conducted by experts, and normally manually evaluated and rated. Based on that, it is safe to some extent to assign assumed values in our implementation to each of the factors that control the selected attributes of the vulnerabilities. We will use the synthesized vulnerabilities listed in table 6.8 in different security scenarios and conduct the prediction as follows:

Table 6.8 System Vulnerabilities and Their Controlling Factors

No	**Vul Id**	**Severity attributes**			**Exploitability attributes**		
		C	**I**	**A**	**AC**	**AV**	**E**
1	Vul-1	1	0	0	1	2	2
2	Vul-2	0	1	0	1	1	1
3	Vul-3	0	0	1	2	3	1
4	Vul-4	2	0	0	1	2	2
5	Vul-5	0	2	0	1	1	1
6	Vul-6	2	0	2	2	3	1
7	Vul-7	1	0	1	1	2	1
8	Vul-8	1	1	1	1	1	1

9	Vul-9	1	2	2	2	2	2

Using the equations from chapter Four, we calculate the exposure level by calculating the severity and exploitability each vulnerability could cause. As mentioned in the design chapter we will only use the positive values in calculating the severity. For the sake of implementation, we choose to put equation (3) in the form:

$$Ex = 0.25 \times \text{Severity} \times \text{exploitability} \tag{21}$$

Where 0.25 is a procedural regulator that keeps the value of exposure degree at the selected range of values. Then the results analysis of the collected vulnerabilities will be as follows:

Table 6.9 Analysis of the Collected Vulnerabilities.

No	**Vulnerability ID**	**Severity**	**Exploitability**	**Exposure**
1	Vul-1	1	6	1.5
2	Vul-2	1	2	0.5
3	Vul-3	1	5	1.25
4	Vul-4	2	6	3
5	Vul-5	2	2	1
6	Vul-6	4	5	5
7	Vul-7	1	3	0.75
8	Vul-8	1	2	0.5
9	Vul-9	4	8	8

The last analysis in table 6.18 showed a variety of exposure levels for the collected vulnerabilities. A computer system initially might have any combination of these potential vulnerabilities; therefore, we suggest different scenarios where the system contains different types of weaknesses as follows:

6.3.1 Low Exposure Scenario

In this situation, the system contain the vulnerabilities; Vul-1, Vul-5, Vul-3, Vul-8, Vul-2, Vul-7. The calculated exposure levels of these weaknesses all lie under the low exposure area. On the other hand, the current absolute threat value might take one of the three probable values, and hence in conjunction with the exposure level we get the following probable risk matrix:

Table 6.10 Matrix of Absolute Threat at Low Exposure Level

Level of Exposure / Absolute Threat	Low
Low	Low
Moderate	Low
High	Low

As all possible Risk values refer to the ‘Low’ value, then we apply the network situation monitoring result to the current risk value as following:

Table 6.11 Matrix of Network Situation Affect at Low System Risk

Risk / Network Situation	Low
Normal	Low
Port scans	Moderate
Vulnerability scan	High

6.3.2 Medium Exposure Scenario

In this situation, we assume the system contains the vulnerabilities; Vul-1, Vul-5, Vul-3, Vul-8, Vul-2, Vul-6. The calculated exposure levels of these weaknesses lie under the low exposure area except for Vul-6 that lies under the ‘Medium’ risk level. In this case, Vul-6 will lead the level of exposure that faces the system, which means in this case the exposure level for the whole system will be 'Medium'. The risk level will be affected accordingly and considering the absolute threat to be as in the table 6.12.

Table 6.12 Matrix of Absolute Threat at Moderate Exposure Level

Level of Exposure / Absolute Threat	Moderate
Low	Low
Moderate	Moderate
High	Moderate

Then the final risk level will be determined according to the network situation bearing in mind that despite the absolute threat taking three potential values, the risk will only take two values with greater possibility than the value 'Moderate'. Therefore, the greater possible value of the risk level will only be considered as a precaution as follows:

Table 6.13 Matrix of Network Situation Affect at Moderate System Risk

Risk / Network Situation	Moderate
Normal	Moderate
Port scans	High
Vulnerability scan	Critical

6.3.3 High Exposure Scenario

In a system that contain the vulnerabilities; Vul-1, Vul-5, Vul-6, Vul-8, Vul-3, Vul-9, according to the analysis these weaknesses lie on different levels of exposure. The hardest vulnerability is Vul-9 that lies under high exposure area, therefore, it raises the level of exposure of the whole system to high exposure and the pairing with absolute threat will produce the following risk matrix:

Table 6.14 Matrix of Absolute Threat at High Exposure Level.

Level of Exposure / Absolute Threat	High
Low	Medium
Moderate	High
High	Critical

The previous risk matrix showed the base risk level that could be done offline, which means when the system is isolated from any outer interaction. At this step, our framework will update the risk level according to the network situation, as follows:

Table 6.15 Matrix of Network Situation Affect at High System Risk.

Risk / Network Situation	High
Normal	High
Port scans	Critical
Vulnerability scan	Critical

6.4 Thresholding Evaluation

Monitoring the network to predict attacks is a contentious task because these attacks occur at an unknown time. This involves analyzing vast volumes of data to detect different clues of intrusions that become more complicated and challenging to manage and perform. This section will discuss the thresholding process with particular focus on the following requirements: *Dynamic, Adaptable, No Prior Knowledge, and Autonomous.*

Our scan detection method depends primarily on monitoring the network traffic for anomaly traces in the traffic. The efficiency of such a methodology is determined by the threshold and the method of defining the threshold, for the targeted environments, where the normal loads of the network are volatile and vary non-linearly during the work time. Therefore a dynamic thresholding method is more suitable for such environments. Our thresholding method is designed to calculate the threshold for each traffic slice online and regardless of the previous traffic thresholds. This threshold will only be applied to that traffic slice and does not affect the successor threshold. This allows the detection method to compare the data sources against each other rather than against a predefined fixed threshold. Table 6.19 clearly displayed the threshold for each traffic slice and obviously appeared varied according to the size of the tested data, which provided the evidence for the dynamic requirement. All the regenerated methods and other reviewed methods in the literature depend on fixed thresholds that are defined previously whether by learning data or calculated manually by an expert statistician.

Another key aspect of the detection method and hence the IPFCC is the ability to make decisions automatically without involvement of human intervention. This actually achieves one of the main goals of designing the IPFCC. Quantifying the threshold online directly from the investigated data and then using that obtained threshold at the same time for decision making (as explained in section 4.4.3.2) fulfils the *Autonomous* criterion. The only human intervention is represented by designing the template of the threshold during the development stage and calibrating the threshold rule to get best results, but the numerical value of the threshold is calculated each time a time window lapses as shown in table 6.19. This leads us also to claim that the *Adaptable* criterion has been met, because the same table showed the performance does not change when different work conditions change. The *No Prior Knowledge* criterion is obviously met because the thresholding mechanism does not need any information acquired from previously detected incidents.

It is important to re-emphasize that all investigated methods define the threshold values initially before the detection process is launched, while our method calculates the threshold during the runtime.

6.5 A Case Study of IPFCC

The previous sections evaluate our solution modules solely, in this section, we present the holistic walk-through the modules functions and the dataflow using a case when supposed vulnerability exist. Before starting a system evaluation using real data, it is important to refer to that the system risk and security level is vary from time to time according to the vulnerabilities existed in the system. This means that the system risk may increase after discovering new vulnerabilities or decrease after patching the discovered vulnerabilities.

Therefore we will conduct our evaluation on a base of assumed vulnerabilities are exist and scans are occur. Another important point to be considered is that although the vulnerabilities are been patched after they been discovered and publicly been announced, and therefore, not represent a threat to the system. However, the average time to close a vulnerability is 62 days [260], which is a comfortable period for a malicious parties to develop an exploit code and use it against vulnerable system for malicious purposes.

Another important point to be considered is that the risk and the conditional probability of each individual attack or security incident is normally quantified manually be experts who assess and produce these values.

6.5.1 Database State

In order to apply our framework, we evaluate real-world vulnerabilities that discovered and announced publically that been collected in section 5.2.3. As in the table 6.16.

Table 6.16 Real Evaluated Vulnerabilities.

Vul ID	AC	AV	E	C	I	A
CVE-2012-1516	1	3	1	2	2	2
CVE-2012-0384	1	3	1	2	2	2
CVE-2014-0160	2	3	2	2	0	0
CVE-2014-6271	2	3	2	2	2	2
CVE-2008-1447	1	3	2	0	1	0
CVE-2012-1342	2	3	1	0	1	0
CVE-2013-6014	2	1	2	2	0	2
CVE-2011-1265	2	1	2	2	2	2
CVE-2014-0224	1	3	2	2	2	0
CVE-2016-1645	2	3	1	2	2	2
CVE-2016-2118	1	3	1	2	2	2
CVE-2003-0352	3	3	2	1	1	1

We suppose that the database inventory is contain only the evaluated vulnerabilities, and based on that we will nominate vulnerability to evaluate its effect on the system security. We select to undertake the assessment on the vulnerability ‘CVE-2003-0352’ with a detailed information about the vulnerability mentioned in the table 6.16.

6.5.2 Vulnerability Description

The vulnerability under the evaluation is the vulnerability named ‘CVE-2003-0352’. This vulnerability affect the Remote Procedure Call (RPC) in some version of Microsoft windows that uses specific version of Distributed Component Object Model (DCOM) interface [261]. This vulnerability allow a remote attacker to execute a malicious code via crafted packets.

6.5.3 Potential Attacks and Attacks Attack Vector

This vulnerability enables a buffer overflow attack; the first attack discovered is a worm known as ‘Blaster’ (known also as ‘Lovsan’) and other variations that follows such as dcomrpc.c and DComExpl_UnixWin32.zip [261]. The attack vector for this vulnerability is the ‘network’ and DCOM-RPC is listening to port 135, therefore, the attacker should target the port 135.

6.5.4 Threats, Risks, and Probabilities

As defined in section 4.6.1, the threat is a type of harm that could occur to the system by different attacks that targeting or using system vulnerabilities. Different threats might occur to the computer system such as destruction of information, corruption of information, Theft and loss of information, illegal usage, data disclosure, denial of service, privilege escalation, etc [262]. Each threat occur to the system according to the type of the service provided or the importance of data contained. Therefore, a system providing multiple services is liable to encounter multiple threats, each threat is resulted by multiple attacks, and each attack might achieved using different vulnerabilities and techniques. These tangled and divergent relationships among the threats and vulnerabilities results in a complicated calculation for such system. For concept proofing, we suppose that the target system is a personal computer PC that contain encrypted credentials and personal information. An attacker to such type of systems might desire to harm the victim or steal their credentials or sensitive data, which exposed the system to threats of destruction of information, and Theft and loss of information. All related details are demonstrated as following:

A. **The target system**: a personal computer.
B. **Potential threats**: destruction of information, and Theft and loss of information
C. **The Absolute Risk :** the initial risk might face the target system is quite low we assign it the value of 0.1

6.5.5 Monitoring-Driven Predictions

Having all of those base information as the de facto of the system is the initial state that our solution start with to predict the system security situation. As summary of the de facto facts, information collected and then calculated using the equations from section 5.6.1, and presented as following:

Existed vulnerabilities: CVE-2003-0352

Affected services: DCOM-RPC

Affected ports: 135

Potential threat: buffer overflow

Potential attacks: dcomrpc.c, DComExpl_UnixWin32.zip, Dcomrpc.c.

Vulnerable services and protocols: TCP, X server virtual frame buffer service, I2P HTTP/S proxy, krb524 Linux Service, NV-Video.

System vulnerabilities evaluation:

Vul ID	AC	AV	E	C	I	A
CVE-2003-0352	3	3	2	1	1	1

Vulnerability Severity = 1

Vulnerability Exploitability = 12

Probability of the threat = 0.3

The supposed probability of the threat caused by a single attack is 0.1, and due to there are three potential attacks, therefore, according to equation (12) the total probability of threat is 0.3.

Risk:

The current risk for such system is calculated using equation (13) as following:

Risk = exposure level × probability of absolute risk

$= (1\times12) \times (0.3)$

$= 12 \times 0.3$

$= 3.6$

This is the risk that the system faces in normal cases, which lies in 'Low' level of risk. In this case, table 6.12 is applied as following:

Risk / Network Situation	Low
Normal	Low
Port scans	Moderate
Vulnerability scan	High

On Port Scan:

Starting with 'Low' risk level then a port scan detected, this scan could be initiated by security team or by hostile party, therefore, IPFCC raises risk level to 'Moderate' as there is a chance that this scanning is malicious and security team should be notified of the current system risk level.

On Vulnerability Scan:

Staring with risk level at 'moderate', if a vulnerability scan is detected then the IPFCC raises the risk level to 'High'. As at this risk level, a vulnerability scanning is a malicious act against the system hence IPFCC predict a potential attack is in preparation depending scanning activities, therefore, notifies security team to be alarmed of system situation and prepare suitable countermeasures.

7.6 Summary

This chapter has evaluated the IPFCC system and its fundamental modules against the abstract-level design requirements. These requirements defined the essential aspects that IPFCC should offer to be suitable for intrusion prediction for cloud computing and network based environments. A summary of the supporting evidence that been used to evaluate our design is displayed in Table 6.17.

Table 6.17 Summary of Design Requirement Evidences.

Design Requirement	**Evidence of Fulfilment**
Accuracy	Experimental analysis in section 6.1
Detection Speed	Experimental analysis in section 6.1
High performance	Experimental analysis in section 6.1
Real-time	Experimental analysis in section 6.1
Scalable	Experimental analysis in section 6.1
Dynamic	Experimental analysis in section 6.3
Adaptable	Experimental analysis in section 6.3
No Prior Knowledge	Experimental analysis in section 6.3
Autonomous	Experimental analysis in section 6.3

The intrusion prediction capabilities of the IPFCC were evaluated against several performance aspects via conducting multiple experiments upon well-known datasets in section 6.1. The results of these experiments showed the high detection accuracy and very low false alarm levels that IPFCC has achieved. In section 6.2 the intrusion prediction was discussed via different attack scenarios. Finally in section 6.3, a discussion of the thresholding method performance was presented and showed the effectiveness of our novel mechanism.

Chapter 7

Conclusions and Future Work

This chapter presents an overview of the contributions and findings of the research conducted in this PhD work, and then sheds a light on the potential improvements and extensions for the proposed work into future research.

7.1 Research Contributions and Findings

The ever growing Internet with divergent uses of people and industry, accompanied by the fact that computer networks were not designed with security considerations in mind results in continuously discovering resident security vulnerabilities that are used to launch attacks. This reflects on all systems that use the network as a medium of communication such as cloud computing. The need for predicting potential intrusions to the system poses one of the trendy research areas nowadays. This thesis has presented the IPFCC intrusion prediction system along with novel constituent techniques and a study of port scan detection methods.

The field of intrusion prediction systems that is proposed to be network-based has not witnessed a sufficient amount of research work, and for cloud computing only very few models have been proposed [34] [263]. This field attracts research attention because of the rising number of attacks that indicate the current intrusion detection systems are insufficient to secure information systems. In some cases when the attackers are highly skilled and the launched attacks are sophisticated, the detection is not that feasible because it has only taken place after the actual harm has occurred. Therefore, the importance of intrusion prediction systems stems from the efforts to provide information about potential attacks, so security administration will be equipped with suitable countermeasures.

One drawback that the current solutions have, is dealing with the attacks as a single step and trying to detect that single step using the signature of that attack. This however, can be evaded by the attackers using different techniques, which means achieving the same attack using different variables or tactics. This results in the detection of attacks occurring during or after the duration of launching the attacks. In the attempt to stop the attacks, we focused on sensing the preparations that precede the launching of an attack. The techniques developed for IPFCC allow the prediction of the potential attacks by sensing the preparation for attacking. This is represented in IPFCC by the ability to detect the scan activities that are launched against the targeted system, as demonstrated in section 6.1.

Quantifying the threshold for detection is another area in which the present approaches are struggling. The existing solutions for scan detection often define the threshold offline either by using machine-learning methods, or manually by an expert statistician. This method might not be capable of accommodating the need and nature of the modern networks, with the normal usage not following a steady pattern. To overcome this, the thresholding technique devised for IPFCC uses an online dynamic thresholding that is calculated during the runtime for the tested data as demonstrated in section 6.3.

The above research points were obtained from the analytical study of different scan detection methods that were proposed to work in similar conditions. This investigation led to the formation of the method that is the most suitable for current networks and what addition might improve the work of these detection methods. The study includes a variation of statistical, probabilistic, and hypothesis examination methods.

A number of contributions and findings have been presented by this thesis to the field of intrusion prediction of network based and cloud computing systems:

1. A statistical intrusion prediction framework that overcomes the challenges of monitoring security in network based environments. The framework offers prediction of the potential intrusions before they happen, whilst ensuring not to disrupt the protected system performance and trivial system resources consumption. The IPFCC is considered novel, as the literature survey has not recognised any prediction solution for cloud computing that uses statistical methods and vulnerabilities information. In addition, our framework does not depend on historical data or previous work outcomes to calibrate the setting of the framework.
2. Analytical study of the existing statistical and probabilistic scan detection methods that represent the background for comprehending the scan detection methods. This field has been chosen from a literature survey of prediction systems in computer and non-computer fields. The study concluded that suitable methods for prediction in cloud computing are the statistical methods, which represented the base of IPFCC.

3. A new statistical algorithm for scan detection that selectively analyses network traffic for scan activities detection. The algorithm uses various techniques to analyse specific traces of scanning on which the anomaly occurred at high rates. This algorithm has been proposed to be more compatible with the targeted environment, which was devised to handle huge amounts of data at high speed rates. In order to overcome the inefficiency of other methods of processing high volumes of data to detect intrusions, only specific attributes of the traffic were monitored and processed for the detection that reflected in considerable reduction of the amount of processed data.
4. A dynamic statistical technique for thresholding that was calculated adaptively for each given traffic window. This technique proposed to overcome the tradition of setting an estimated threshold based on historical data, which decreased the detection performance of the whole system. The technique offers in-depth dissimilarity among the monitored attribute values for the different variables (sources) in the same set. The technique is new for the anomaly detection, as the literature survey has revealed that no existing methods offer a comparable level of dissimilarity checking to calculate the threshold.
5. A novel use of the publicly available information about the vulnerabilities of the system in the form of a risk assessment that increased according to the situation of the network. This mechanism was adapted to provide a platform neutral method that quantifies and ranks the weaknesses, and calculates the risk level of the system. This method is considered novel as no solution that appeared in the literature survey has incorporated vulnerability information with the traffic information to predict the potential attacks.

7.2 Limitations

There are some limitations that are associated with the proposed system; this section discussing them as follows:

1. **Specialized Dataset for cloud computing**: unfortunately, a there is a lack of experimental dataset that is specialized for testing cloud security solutions. Although we used a well-known dataset for intrusion detection that uses a cloud setup parameters, however, we believe more precise results would be produced with data collected from real life cloud traffic.
2. **Sensitivity Issue**: the port scan detection algorithm (which is the core of our prediction system) has suffered from high ratio of false positive alarms, in spite of the high

detection ratio. This results of considerable amount of alarms that might confuses the security team. More refinement is needed to decrease the false positive ratio, which in turn improve our prediction system sensitivity.

3. **Dependency on external information sources**: vulnerability scanners has important role in the system security by reporting the current vulnerabilities. These scanners collect information about the vulnerabilities from public databases that are obviously outside the system. This limited the system evaluation process and bind it to external sources that discover, define and rank new vulnerabilities.
4. **Location Centric**: our proposed system is supposed to monitor the incoming traffic to the cloud, which means that our system located on the cloud network interface. This means limited our system effectivity on the attacks that targeted the cloud from the outside, and threats that are launched from the inside like insiders attacks are challenging to the proposed solution.

7.3 Future Work

The work presented in this thesis can be applied to different systems in different domains. Therefore, modification and extensions on different levels could be used to address different systems security needs as follows:

1. IPFCC could be enhanced by conducting fine-grained analysis to predict specific attacks before they happen. All the risk analysis has formed a general risk assessment for the system as whole, while, it is possible to conduct this risk assessment according to individual attacks alone. This will make the prediction more precise on the potential attacks that might occur in the future.
2. All of the experiments that were conducted in this thesis were designed to use only very little information from the packets header information, though this has been done intentionally, however, different header information might be exploited to predict more information such as suspected geographical locations. Although our focus was on the malicious scanning activities, different information could be extracted to support the prediction decision or other purpose.
3. The risk assessment presented in this thesis depends on analyzing the vulnerabilities and calculating associated attributes, however, different aspects could be incorporated to improve the prediction such as behaviour of the users, access control and associated

security events. This in return will add more capabilities to the prediction system to produce more specified predictions.

4. The proposed scan detection method for this thesis can easily be extended to detect volumetric attacks such as denial of service (DOS) attack and similar attacks. This is because the mechanism of the scan detection method is a general method that depends on discriminating between malicious and benign according to numbers in each set, and with a simple modification to sense the attack packets, it is weaponized to detect different attacks.
5. The current scan detection does not store threshold values as they are disposable. However, storing these values and refining them using different statistical methods can be used to calibrate and tune detection and outliers variables to get improved results.
6. Develop knowledge module from simple databases for vulnerabilities information to knowledge center that contains information extracted from both the traffic and the system and processed into beneficial security-wise information.
7. Professional programming development is needed to convert this framework into a universal plug-in tool that could be attached to different intrusion detection systems, and work efficiently to predict intrusions.
8. The concept of anomaly detection that is embedded in the scan detection method, which calculates outliers of the set and sets a threshold then measures the Euclidian distance to the threshold, can be extended and exploited in different domains where the changing point identification is a problematic issue.
9. Allocate research efforts to transform this framework from centralized to decentralized system using small units of the system at each virtual machine, then a confederate unit to be responsible for orchestrating the prediction process, and calibrating variables' values.

References

[1] P. Mell and T. Grance, “The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology,” *National Institute of Standards and Technology Special Publication 800-145*, 2011.

[2] A. Lin and N. Chen, “Cloud computing as an innovation : Percepetion , attitude , and adoption,” *Int. J. Inf. Manage.*, vol. 32, no. 2012, pp. 533–540, 2013.

[3] LIVEWIRE Training Solutions, “Secure a sparkling career in cloud,” *Web Page*, 2018. [Online]. Available: https://www.livewireindia.com/cloud_computing_training.php.

[4] B. Grobauer, T. Walloschek, and E. Stöcker, “Understanding cloud computing vulnerabilities,” *IEEE Secur. Priv.*, vol. 9, no. 2, pp. 50–57, 2011.

[5] R. Bhadauria and S. Sanyal, “Survey on Security Issues in Cloud Computing and Associated Mitigation Techniques,” *Int. J. Comput. Appl.*, vol. 47, no. 18, pp. 47–66, 2012.

[6] G. Galante and L. C. E. De Bona, “A survey on cloud computing elasticity,” in *Proceedings - 2012 IEEE/ACM 5th International Conference on Utility and Cloud Computing, UCC 2012*, 2012, pp. 263–270.

[7] R. Kui, W. Cong, and W. Qian, “Security Challenges for the Public Cloud,” *Internet Comput. IEEE*, vol. 16, no. 1, pp. 69–73, 2012.

[8] K. Owens, “Securing Virtual Computer Infrastructure in the Cloud,” *SAVVIS*, 2009. .

[9] J. Chen, Y. Wang, and X. Wang, “On-demand security architecture for cloud computing,” *Computer (Long. Beach. Calif).*, vol. 45, no. 7, pp. 73–78, 2012.

[10] R. Choubey, R. Dubey, and J. Bhattacharjee, “A survey on cloud computing security, challenges and threats,” *Int. J. Comput. ...*, vol. 3, no. 3, pp. 1227–1231, 2011.

[11] S. Subashini and V. Kavitha, “A survey on security issues in service delivery models of cloud computing,” *Journal of Network and Computer Applications*, vol. 34, no. 1. pp. 1–11, 2011.

[12] S. Roschke, F. Cheng, and C. Meinel, “Intrusion Detection in the Cloud,” in *Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing, 2009. DASC ’09.*, 2009, pp. 729–734.

[13] M. N. Ismail, A. Aborujilah, S. Musa, and Aa. Shahzad, “Detecting flooding based DoS attack in cloud computing environment using covariance matrix approach,” in *ICUIMC ’13 Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication*, 2013, pp. 1–6.

[14] A. Chadd, “The cost of DDoS attacks and guarding against them,” *COMPARE THE CLOUD LTD*, 2018. [Online]. Available: https://www.comparethecloud.net/articles/2017-cyberattacks/.

[15] W. Ashford, “Ransomware to hit cloud computing in 2018, predicts MIT,” *TechTarget*, 2018. [Online]. Available: https://www.computerweekly.com/news/450432488/Ransomware-to-hit-cloud-computing-in-2018-predicts-MIT.

[16] A. D. Rayome, “Cloud security market to reach $12B by 2024, driven by rise of cyber attacks,” *TechRepublic*, 2017. [Online]. Available: https://www.techrepublic.com/article/cloud-security-market-to-reach-12b-by-2024-driven-by-rise-of-cyber-attacks/.

[17] The Snort Team, “Snort,” *Cisco*, 2018. [Online]. Available: https://www.snort.org/.

[18] Security Onion Solutions, “SECURITY ONION,” 2018. [Online]. Available: https://securityonion.net/.

[19] L. 2014 myNetWatchman, “MyNetWatchman,” 2018. [Online]. Available: http://www.mynetwatchman.com/#home.

[20] Internet Storm Center, “Dshield,” 2018. [Online]. Available: https://dshield.org/.

[21] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan, “Fast portscan detection using sequential hypothesis testing,” in *Proceedings of IEEE Symposium on Security and Privacy, 2004.*, 2004, pp. 211–225.

[22] S. E. Schechter, J. Jung, and A. W. Berger, “Fast Detection of Scanning Worm

Infections," in *Recent Advances in Intrusion Detection: 7th International Symposium, RAID 2004, Sophia Antipolis, France, September 15 - 17, 2004. Proceedings*, E. Jonsson, A. Valdes, and M. Almgren, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 59–81.

[23] C. Leckie and R. Kotagiri, "A probabilistic approach to detecting network scans," *IEEE/IFIP Netw. Oper. Manag. Symp. (NOMS).*, pp. 359–372, 2002.

[24] S. Staniford, J. a. Hoagland, and J. M. McAlerney, "Practical automated detection of stealthy portscans," *J. Comput. Secur.*, vol. 10, no. (1,2), pp. 105–136, 2002.

[25] D. Whyte, E. Kranakis, and P. Van Oorschot, "DNS-based detection of scanning worms in an enterprise network," *Netw. Distrib. Syst. Symp.*, no. 1, pp. 1–17, 2005.

[26] M. M. Williamson, "Throttling viruses: Restricting propagation to defeat malicious mobile code," in *Proceedings - Annual Computer Security Applications Conference, ACSAC*, 2002, vol. 2002–Janua, pp. 61–68.

[27] K. Borders, L. Falk, and A. Prakash, "OpenFire: Opening Networks to Reduce Network Attacks on Legitimate Services," 2006.

[28] S. Gupta, P. Kumar, and A. Abraham, "A profile based network intrusion detection and prevention system for securing cloud environment," *Int. J. Distrib. Sens. Networks*, vol. 2013, pp. 1–12, 2013.

[29] W. Yassin, N. I. Udzir, Z. Muda, A. Abdullah, and M. T. Abdullah, "A Cloud-based Intrusion Detection Service framework," in *Proceedings 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic, CyberSec 2012*, 2012, pp. 213–218.

[30] P. S. Antón, R. H. Anderson, R. Mesic, and M. Scheiern, *Finding and Fixing Vulnerabilities in Information Systems: The Vulnerability Assessment and Mitigation Methodology*. 2003.

[31] N. R. Mead and T. Stehney, "Security quality requirements engineering (SQUARE) methodology," *ACM SIGSOFT Softw. Eng. Notes*, vol. 30, no. 4, pp. 1–7, 2005.

[32] The MITRE Corporation, "Cyber Mission Assurance Engineering: A Risk-Based,

Threat-Informed Approach to Address Advanced Adversaries," 2013.

[33] FIRST.Org, "Common Vulnerability Scoring System v3.0: Specification Document," 2015.

[34] H. A. Kholidy, A. M. Yousof, and H. A. Ali, "Online Risk Assessment and Prediction Models For Autonomic Cloud Intrusion Prevention Systems," in *11th International Conference on Computer Systems and Applications (AICCSA), 2014 IEEE/ACS*, 2014, pp. 715–722.

[35] A. A. Hisham A. Kholidy, Abdelkarim Erradi, Sherif Abdelwahed, "A Finite State Hidden Markov Model for Predicting Multistage Attacks in Cloud Systems," in *12th International Conference on Dependable, Autonomic and Secure Computing (DASC), 2014 IEEE*, 2014, no. 1, pp. 14–19.

[36] A. Roy, santonu sarkar, R. Ganesan, and G. Goel, "Secure the Cloud: From the Perspective of a Service-Oriented Organization," *ACM Comput. Surv.*, vol. 47, no. 3, 2015.

[37] P. Nespoli, D. Papamartzivanos, F. G. Mármol, and G. Kambourakis, "Optimal countermeasures selection against cyber attacks: A comprehensive survey on reaction frameworks," *IEEE Commun. Surv. Tutorials*, vol. PP, no. 99, pp. 1–37, 2017.

[38] C. Cloud Security Alliance, "Cloud adoption practices & priorities survey report," 2015.

[39] T. Abels, P. Dhawan, and B. Chandrasekaran, "An overview of xen virtualization," *Dell Power Solut.*, vol. 8, no. August, pp. 109–111, 2005.

[40] VMware, "Virtualization Overview," 2006.

[41] F. Lombardi and R. Di Pietro, "Secure virtualization for cloud computing," *J. Netw. Comput. Appl.*, vol. 34, no. 4, pp. 1113–1122, 2011.

[42] L. MacVittie, "Architectural Multi-tenancy," *DevCentral*, 2010. .

[43] T. Ristenpart and E. Tromer, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *CCS '09 Proceedings of the 16th ACM*

conference on Computer and communications security, 2009, pp. 199–212.

[44] Common Vulnerabilities and Exposures (CVM), “CVE-2014-9718,” *NATIONAL VULNERABILITY DATABASE (NVD),* 2015. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-9718.

[45] G. Zhu, Y. Yin, R. Cai, and K. Li, “Detecting Virtualization Specific Vulnerabilities in Cloud Computing Environment,” in *IEEE International Conference on Cloud Computing, CLOUD,* 2017, vol. 2017–June, pp. 743–748.

[46] C. Modi, D. Patel, B. Borisaniya, A. Patel, and M. Rajarajan, “A survey on security issues and solutions at different layers of Cloud computing,” *J. Supercomput.,* vol. 63, no. 2, pp. 561–592, 2013.

[47] H. Y. Tsai, M. Siebenhaar, A. Miede, Y. Huang, and R. Steinmetz, “Threat as a service?: Virtualization’s impact on cloud security,” *IT Prof.,* vol. 14, no. 1, pp. 32–37, 2012.

[48] T. Garfinkel and M. Rosenblum, “When Virtual is Harder Than Real: Security Challenges in Virtual Machine Based Computing Environments,” *Proc. 10th Conf. Hot Top. Oper. Syst. - Vol. 10,* pp. 20–25, 2005.

[49] J. Oberheide, E. Cooke, and F. Jahanian, “Empirical exploitation of live virtual machine migration,” in *Proc. of BlackHat DC convention,* 2008, pp. 1–6.

[50] M. Zalewski, “Browser Security Handbook,” *Google Security Blog,* 2008. [Online]. Available: https://code.google.com/archive/p/browsersec/wikis/Main.wiki.

[51] Verizon Business, “2008 Data Breach Investigations Report,” *Trends,* pp. 1–29, 2008.

[52] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, “A view of Cloud Computing,” *Commun. acm,* vol. 53, no. 4, pp. 50–58, 2010.

[53] D. Wichers and J. Williams, “OWASP Top 10 2013,” 2010.

[54] A. K. Sood, “Salesforce Accounts Susceptible to Hijacking using XSS Flaw,” *Symantec Official Blog,* 2015. [Online]. Available:

https://www.symantec.com/connect/blogs/salesforce-accounts-susceptible-hijacking-using-xss-flaw.

[55] A. Shrivastava, S. Choudhary, and A. Kumar, “XSS vulnerability assessment and prevention in web application,” *Proc. 2016 2nd Int. Conf. Next Gener. Comput. Technol. NGCT 2016*, no. October, pp. 850–853, 2017.

[56] F. MAHAT, “Blind Cross-Site Scripting (XSS) Attack, Vulnerability, Alert and Solution,” *Enterprise Solution Professional on Information and Network*, 2016. [Online]. Available: https://www.e-spincorp.com/documentation/blind-cross-site-scripting-xss-attack-vulnerability-alert-and-solution/.

[57] M. Price, “The paradox of security in virtual environments,” *Computer (Long. Beach. Calif).*, vol. 41, no. 11, pp. 22–28, 2008.

[58] A. Matrosov, M. Gorobets, O. Bazhaniuk, A. Furtak, and Y. Bulygin, “Attacking Hypervisors via Firmware and Hardware,” 2015.

[59] N. Falliere and E. Chien, “Zeus: King of the Bots,” 2009.

[60] Cloud Security Alliance, “Top Threats to Cloud Computing,” 2010.

[61] L. Constantin, “Moonpig jeopardizes data of millions of customers through insecure API,” *IDG News Service*, 2015. [Online]. Available: https://www.computerworld.com/article/2865794/moonpig-jeopardizes-data-of-millions-of-customers-through-insecure-api.html.

[62] D. Shackleford, “Cloud API security risks: How to assess cloud service provider APIs,” *Voodoo Security*, 2013. [Online]. Available: http://searchcloudsecurity.techtarget.com/tip/Cloud-API-security-risks-How-to-assess-cloud-service-provider-APIs.

[63] B. Albelooshi, K. Salah, T. Martin, and E. Damiani, “Experimental Proof: Data Remanence in Cloud VMs,” in *Proceedings - 2015 IEEE 8th International Conference on Cloud Computing, CLOUD 2015*, 2015, pp. 1017–1020.

[64] W. E. Burr, D. F. Dodson, and W. Timothy Polk, “Electronic Authentication Guideline,” *Natl. Inst. Stand. Technol.*, vol. 800–63, no. 1.0.2, p. 64, 2006.

[65] S. M. Dejamfar and S. Najafzadeh, “Authentication Techniques in Cloud Computing: A Review,” *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 7, no. 1, pp. 95–99, 2017.

[66] Z. Shen, L. Li, F. Yan, and X. Wu, “Cloud Computing System Based on Trusted Computing Platform,” in *2010 International Conference on Intelligent Computation Technology and Automation*, 2010, pp. 942–945.

[67] A. G. Revar and M. D. Bhavsar, “Securing user authentication using single sign-on in Cloud Computing,” in *2011 Nirma University International Conference on Engineering*, 2011, pp. 1–4.

[68] G. L. Masala, P. Ruiu, and E. Grosso, “Biometric Authentication and Data Security in Cloud Computing,” in *Computer and Network Security Essentials*, K. Daimi, Ed. Cham: Springer International Publishing, 2018, pp. 337–353.

[69] M. Küppers, “Anleitung: Multi-Faktor-Authentifizierung in Azure Active Directory aktivieren,” *Windows Pro*, 2016. .

[70] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. Ullah Khan, “The rise of ‘big data’ on cloud computing: Review and open research issues,” *Inf. Syst.*, vol. 47, pp. 98–115, 2015.

[71] B. D. Jayant Research Scholar and A. S. Sulabha, “Analysis of DAC MAC RBAC Access Control based Models for Security,” *Int. J. Comput. Appl.*, vol. 104, no. 5, pp. 975–8887, 2014.

[72] L. Zhou, V. Varadharajan, and M. Hitchens, “Achieving Secure Role-Based Access Control on Encrypted Data in Cloud Storage,” *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 12, pp. 1947–1960, 2013.

[73] Amazon Web Services (AWS), “Amazon Simple Storage Service: Developer Guide,” 2006.

[74] W. Wang, J. Han, M. Song, and W. Xiaohui, “The design of a trust and role based access control model in cloud computing,” in *6th International Conference on Pervasive Computing and Applications (ICPCA), 2011*, 2011, pp. 330–334.

[75] Z. Tianyi, L. Weidong, and S. Jiaxing, “An Efficient Role Based Access Control

System for Cloud Computing," in *2011 IEEE 11th International Conference on Computer and Information Technology*, 2011, pp. 97–102.

[76] L. Sun, H. Wang, J. Yong, and G. Wu, "Semantic access control for cloud computing based on e-Healthcare," in *Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2012*, 2012, pp. 512–518.

[77] H. A. J. Narayanan and M. H. Gunes, "Ensuring access control in cloud provisioned healthcare systems," in *2011 IEEE Consumer Communications and Networking Conference, CCNC'2011*, 2011, pp. 247–251.

[78] Y. A. Younis, K. Kifayat, and M. Merabti, "An access control model for cloud computing," *J. Inf. Secur. Appl.*, vol. 19, no. 1, pp. 45–60, 2014.

[79] R. Silhavy, Roman Senkerik, and Zuzana K. Oplatkova, *Cybernetics and Mathematics Applications in Intelligent Systems*, vol. 2. 2017.

[80] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, Randy H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A Berkeley view of cloud computing," 2009.

[81] B. Butler, "Amazon's S3 cloud storage service isn't working," *Network World from IDG*, 2017. [Online]. Available: https://www.networkworld.com/article/3175686/cloud-computing/amazon-s-s3-cloud-storage-service-isn-t-working.html.

[82] Z. Tan, A. Jamdagni, X. He, P. Nanda, and R. P. Liu, "A system for denial-of-service attack detection based on multivariate correlation analysis," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 447–456, 2014.

[83] CSA, "Security Guidance Critical Areas of Focus for Critical Areas of Focus in Cloud Computing V3," *Cloud Secur. Alliance*, vol. 1, pp. 1–177, 2011.

[84] S. Moss, "Major DDoS attack on Dyn disrupts AWS, Twitter, Spotify and more," *Data Centre Dynamics Ltd*, 2016. [Online]. Available: http://www.datacenterdynamics.com/content-tracks/security-risk/major-ddos-attack-

on-dyn-disrupts-aws-twitter-spotify-and-more/97176.fullarticle.

[85] G. Carl, G. Kesidis, R. R. Brooks, and S. Rai, “Denial-of-service attack-detection techniques,” *IEEE Internet Comput.*, vol. 10, no. 1, pp. 82–89, 2006.

[86] Cisco, “A Cisco Guide to Defending Against Distributed Denial of Service Attacks,” *white paper*, 2017. [Online]. Available: A Cisco Guide to Defending Against Distributed Denial of Service Attacks%0A.

[87] H. Liu, “A New Form of DOS Attack in a Cloud and Its Avoidance Mechanism,” in *CCSW ’10 Proceedings of the 2010 ACM workshop on Cloud computing security workshop*, 2010, pp. 65–75.

[88] M. Jensen, N. Gruschka, and R. Herkenhöner, “A survey of attacks on web services,” *Comput. Sci. - Res. Dev.*, vol. 24, no. 4, p. 185, 2009.

[89] Q. Yan, F. R. Yu, S. Member, Q. Gong, and J. Li, “Software-Defined Networking (SDN) and Distributed Denial of Service (DDoS) Attacks in Cloud Computing Environments : A Survey , Some Research Issues , and Challenges,” *IEEE Commun. Surv. Tutorials*, vol. 18, no. c, pp. 2–23, 2015.

[90] G. Somani, M. S. Gaur, D. Sanghi, and M. Conti, “DDoS attacks in Cloud Computing: Collateral Damage to Non-targets,” *Comput. Networks*, vol. 109, no. March 2015, pp. 157–171, 2015.

[91] T. M, N. S, and S. R, “Denial of Service (DoS) Attacks Over Cloud Environment: A Literature Survey,” in *Advancing Cloud Database Systems and Capacity Planning With Dynamic Applications*, IGI Global, 2017, pp. 289–319.

[92] 1ed, “Malware Risks and Mitigation Report,” *BITS -A DIVISION OF THE FINANCIAL SERVICES ROUNDTABLE*, 2011. [Online]. Available: http://fsroundtable.org/wp-content/uploads/2015/05/BITSMalwareReportJun2011.pdf.

[93] FireEye, “FIREEYE ADVANCED THREAT REPORT: 2H 2012,” 2014.

[94] Websense, “2013 THREAT Report,” *Websense Triton*, 2013. [Online]. Available: http://www.websense.com/assets/reports/websense-2013-threat-report.pdf. [Accessed: 27-Feb-2017].

[95] M. Jensen, J. Schwenk, N. Gruschka, and L. Lo Iacono, “On Technical Security Issues in Cloud Computing,” in *2009 IEEE International Conference on Cloud Computing*, 2009, pp. 109–116.

[96] C. Tankard, “Advanced Persistent threats and how to monitor and deter them,” *Netw. Secur.*, vol. 2011, no. 8, pp. 16–19, 2011.

[97] E. M. Hutchins, M. J. Cloppert, and R. M. Amin, “Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains,” in *6th International Conference on Information Warfare and Security*, 2011, pp. 113–125.

[98] T. Sager, “Killing Advanced Threats in Their Tracks: An Intelligent Approach to Attack Prevention,” 2014.

[99] Mandiant, “APT1 Exposing One of China’s Cyber Espionage Units,” *Report*, pp. 1–76, 2013.

[100] M. Balduzzi, J. Zaddach, D. Balzarotti, and S. Loureiro, “A Security Analysis of Amazon ’ s Elastic Compute Cloud Service – Long Version,” in *SAC ’12 Proceedings of the 27th Annual ACM Symposium on Applied Computing*, 2012, pp. 1427–1434.

[101] E. G. Amoroso, “From the enterprise perimeter to a mobility-enabled secure cloud,” *IEEE Secur. Priv.*, vol. 11, no. 1, pp. 23–31, 2013.

[102] AlertLogic, “Tageted Attacks and Opportunistic Hacks, State of Cloud Security Report,” 2013.

[103] C. Beek, Y. Bulygin, D. Frosst, P. Greve, J. Jarvis, E. Peterson, M. Rosenquist, F. Ruiz, C. Schmugar, R. Simon, B. Snell, D. Sommer, B. Sun, and A. Wosotowsky, “McAfee Labs: 2017 Threats Predictions,” 2016.

[104] “Brute force attack,” *Open Web Application Security Project*. [Online]. Available: https://www.owasp.org/index.php/Brute_force_attack. [Accessed: 03-Mar-2017].

[105] IBM, “Brute force attacks,” *IBM Knowledge Center*. [Online]. Available: https://www.ibm.com/support/knowledgecenter/SSB2MG_4.6.0/com.ibm.ips.doc/concepts/wap_brute_force.htm. [Accessed: 03-Mar-2017].

[106] P. Wood, B. Nahorney, K. Chandrasekar, S. Wallace, and K. Haley, "Symantec Internet Security Threat Report," 2016.

[107] D. Cappelli, A. P. Moore, R. F. Trzeciak, and T. J. Shimeall, "Common Sense Guide to Prevention and Detection of Insider Threats 3rd Edition – Version 3.1," 2009.

[108] D. M. Cappelli, A. P. Moore, and R. F. Trzeciak, *The CERT Guide to Insider Threats: How to Prevent, Detect, and Respond to Information Technology Crimes (Theft, Sabotage, Fraud)*. 2012.

[109] W. R. Claycomb and A. Nicoll, "Insider Threats to Cloud Computing: Directions for New Research Challenges," in *IEEE 36th International Conference on Computer Software and Applications*, 2012, pp. 387–394.

[110] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, "A survey of intrusion detection techniques in Cloud," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 42–57, Jan. 2013.

[111] A. Patel, M. Taghavi, K. Bakhtiyari, and J. Celestino Júnior, "An intrusion detection and prevention system in cloud computing: A systematic review," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 25–41, Jan. 2013.

[112] E. Simmon, K. Kim, R. Lee, and F. De Vaulx, "A Vision of Cyber-Physical Cloud Computing for Smart Networked Systems," 2013.

[113] K. Vieira, A. Schulter, C. B. Westphall, and C. M. Westphall, "Intrusion Detection for Grid and Cloud Computing," *It Prof.*, vol. 12, no. 4, pp. 38–43, 2010.

[114] J.-H. Lee, M.-W. Park, J.-H. Eom, and T.-M. Chung, "Multi-level Intrusion Detection System and log management in Cloud Computing," in *13th International Conference on Advanced Communication Technology (ICACT)*, 2011, pp. 552–555.

[115] A. Bakshi and B. Yogesh, "Securing cloud from DDOS attacks using intrusion detection system in virtual machine," in *2nd International Conference on Communication Software and Networks, ICCSN 2010*, 2010, pp. 260–264.

[116] H. Hamad and M. Hoby, "Managing Intrusion Detection as a Service in Cloud Networks," *Int. J. Comput. Appl.*, vol. 41, no. 1, pp. 35–40, 2012.

[117] C. C. Lo, C. C. Huang, and J. Ku, “A cooperative intrusion detection system framework for cloud computing networks,” in *Proceedings of the International Conference on Parallel Processing Workshops*, 2010, pp. 280–284.

[118] A. V. Dastjerdi, K. A. Bakar, and S. G. H. Tabatabaei, “Distributed Intrusion Detection in Clouds Using Mobile Agents,” in *2009 Third International Conference on Advanced Engineering Computing and Applications in Sciences*, 2009, pp. 175–180.

[119] J. Nikolai and Y. Wang, “Hypervisor-based cloud intrusion detection system,” in *2014 International Conference on Computing, Networking and Communications, ICNC 2014*, 2014, pp. 989–993.

[120] T. Garfinkel and M. Rosenblum, “A Virtual Machine Introspection Based Architecture for Intrusion Detection,” in *Network and Distributed System Security Symposium*, 2003, vol. 1, pp. 253–285.

[121] N. Hoque, M. H. Bhuyan, R. C. Baishya, D. K. Bhattacharyya, and J. K. Kalita, “Network attacks: Taxonomy, tools and systems,” *Journal of Network and Computer Applications*, vol. 40, no. 1. pp. 307–324, 2014.

[122] E. M. Hutchins, M. J. Cloppert, and R. M. Amin, “Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains,” in *6th Annual International Conference on Information Warfare and Security*, 2011, pp. 113–125.

[123] Z. Liu, C. Wang, and S. Chen, “Correlating multi-step attack and constructing attack scenarios based on attack pattern modeling,” in *Proceedings of the 2nd International Conference on Information Security and Assurance, ISA 2008*, 2008, pp. 214–219.

[124] S. Cheung, U. Lindqvist, and M. W. Fong, “Modeling multistep cyber attacks for scenario recognition,” in *Proceedings - DARPA Information Survivability Conference and Exposition, DISCEX 2003*, 2003, vol. 1, pp. 284–292.

[125] Dell Software, “Anatomy of a cyber-attack,” 2014.

[126] B. Harris, E. Konikoff, and P. Petersen, “Breaking the DDoS Attack Chain,” 2013.

[127] M. de Vivo, E. Carrasco, G. Isern, and G. O. de Vivo, “A review of port scanning

techniques," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 2, p. 41, 1999.

[128] C. Bailey Lee, C. Roedel, and E. Silenok, "Detection and Characterization of Port Scan Attacks," 2003.

[129] M. Roesch, "Snort: Lightweight Intrusion Detection for Networks.," in *LISA '99: 13th Systems Administration Conference*, 1999, pp. 229–238.

[130] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Surveying port scans and their detection methodologies," *Comput. J.*, vol. 54, no. 10, pp. 1565–1581, 2011.

[131] W. Kröger and E. Zio, *Vulnerable Systems*. Springer-Verlag London Limited, 2011.

[132] National Institute of Standards and Technology (NIST), "National Vulnerability Database (NVD): Summary," *NIST: National Vulnerability Database (NVD)*, 2016. [Online]. Available: https://www.nist.gov/programs-projects/national-vulnerability-database-nvd.

[133] J. M. Myerson, "Identifying enterprise network vulnerabilities," *Int. J. Netw. Manag.*, vol. 12, no. 3, pp. 135–144, 2002.

[134] J. Ruohonen, S. Hyrynsalmi, and V. Leppänen, "The sigmoidal growth of operating system security vulnerabilities: An empirical revisit," *Comput. Secur.*, vol. 55, pp. 1–20, 2015.

[135] R. Antrobus, S. Frey, B. Green, and A. Rashid, "SimaticScan: Towards A Specialised Vulnerability Scanner for Industrial Control Systems," in *4th International Symposium for ICS & SCADA Cyber Security Research 2016*, 2016, no. Infracritical 2014, pp. 11–18.

[136] NMAP, "Nmap: the Network Mapper - Free Security Scanner," *nmap.org*, 2018. [Online]. Available: https://nmap.org/.

[137] R. Lukanta, Y. Asnar, and A. I. Kistijantoro, "A vulnerability scanning tool for session management vulnerabilities," in *Proceedings of 2014 International Conference on Data and Software Engineering, ICODSE 2014*, 2014, pp. 1–6.

[138] Y. Makino and V. Klyuev, "Evaluation of web vulnerability scanners," in *Proceedings*

of the 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2015, 2015, vol. 1, pp. 399–402.

[139] Diamond, “Microsoft Baseline Security Analyzer (MBSA) 2.3 Reviewed,” *DELL*, 2013. [Online]. Available: https://www.dell.com/community/Virus-Spyware/Microsoft-Baseline-Security-Analyzer-MBSA-2-3-Reviewed/td-p/4213449.

[140] A. Ozment, “Improving Vulnerability Discovery Models,” in *QoP ’07 Proceedings of the 2007 ACM workshop on Quality of protection*, 2007, pp. 6–11.

[141] R. Anderson, “Security in Open versus Closed Systems–The Dance of Boltzmann, Coase and Moore,” *Open Source Softw. Econ. Law Policy*, pp. 127–142, 2002.

[142] O. H. Alhazmi and Y. K. Malaiya, “Quantitative vulnerability assessment of systems software,” in *Proceedings of Annual Reliability and Maintainability Symposium, 2005.*, 2005, pp. 615–620.

[143] O. Alhazmi, S. Woo, and Y. Malaiya, “Security vulnerability categories in major software systems.,” in *Proceedings of the Third IASTED International Conference on Communication, Network, and Information Security, CNIS 2006*, 2006, pp. 138–143.

[144] N. I. Daud, K. A. A. Bakar, and M. S. M. Hasan, “A case study on web application vulnerability scanning tools,” in *Proceedings of 2014 Science and Information Conference, SAI 2014*, 2014, pp. 595–600.

[145] K. Houghton, “Vulnerabilities &; Vulnerability Scanning,” 2003.

[146] I. Tenable, “Nessus Professional | Tenable^{TM},” *Tenable Nessus Webiste*, 2017. [Online]. Available: http://www.tenable.com/products/nessus-vulnerability-scanner/nessus-professional.

[147] SAINT Corporation, “SAINT,” 2017. [Online]. Available: http://www.saintcorporation.com/.

[148] Core Security SDI Corporation, “Core Impact,” 2018. [Online]. Available: https://www.coresecurity.com/core-impact.

[149] H. Holm, T. Sommestad, J. Almroth, and M. Persson, “A quantitative evaluation of vulnerability scanning,” *Inf. Manag. Comput. Secur.*, vol. 19, no. 4, pp. 231–247, 2011.

[150] Organization for Internet Safety, “Guidelines for Security Vulnerability Reporting and Response,” 2004.

[151] Cyber Security & Information Systems Information Analysis Center, “Vulnerability Assessment,” 2011.

[152] R. Bace and P. Mell, “NIST Special Publication on Intrusion Detection Systems NIST Special Publication on Intrusion Detection Systems,” *Natl. Inst. Stand. Technol.*, pp. 1–51, 2011.

[153] K. Scarfone and P. Mell, “Guide to Intrusion Detection and Prevention Systems (IDPS) Recommendations of the National Institute of Standards and Technology,” 2007.

[154] G. K. Jayasinghe, J. Shane Culpepper, and P. Bertok, “Efficient and effective realtime prediction of drive-by download attacks,” *J. Netw. Comput. Appl.*, vol. 38, pp. 135–149, Feb. 2014.

[155] J. Wu, L. Yin, and Y. Guo, “Cyber Attacks Prediction Model Based on Bayesian Network,” *2012 IEEE 18th Int. Conf. Parallel Distrib. Syst.*, pp. 730–731, Dec. 2012.

[156] X. Qin and W. Lee, “Attack plan recognition and prediction using causal networks,” *Proc. - Annu. Comput. Secur. Appl. Conf. ACSAC*, pp. 370–379, 2004.

[157] L. Cheng-Bin, “A New Intrusion Prediction Method Based on Feature Extraction,” *2009 Second Int. Work. Comput. Sci. Eng.*, pp. 7–10, 2009.

[158] Z. Zhengdao, P. Zhumiao, and Z. Zhiping, “The Study of Intrusion Prediction Based on HsMM,” *2008 IEEE Asia-Pacific Serv. Comput. Conf.*, pp. 1358–1363, Dec. 2008.

[159] H. Farhadi, M. Amirhaeri, and M. Khansari, “Alert Correlation and Prediction Using Data Mining and HMM,” *ISC Int. J. Inf. Secur.*, vol. 3, no. 2, pp. 77–101, 2011.

[160] Z. Li, J. Lei, L. Wang, and D. Li, “A Data Mining Approach to Generating Network Attack Graph for Intrusion Prediction,” *Fourth Int. Conf. Fuzzy Syst. Knowl. Discov.*

(FSKD 2007), no. Fskd, pp. 307–311, 2007.

[161] O. Onolaja, R. Bahsoon, and G. Theodoropoulos, “Conceptual framework for dynamic trust monitoring and prediction,” *Procedia Comput. Sci.*, vol. 1, no. 1, pp. 1241–1250, May 2012.

[162] J. Holsopple and M. Sudit, “Evaluating threat assessment for multi-stage cyber attacks,” 1999.

[163] P. Kannadiga, M. Zulkernine, A. Haque, and B. Canada, “E-NIPS : An Event-Based Network Intrusion Prediction,” in *10th International Conference, ISC 2007, Valparaíso, Chile, October 9-12, 2007. Proceedings*, 2007, pp. 37–52.

[164] S. J. Yang, A. Stotz, J. Holsopple, M. Sudit, and M. Kuhl, “High level information fusion for tracking and projection of multistage cyber attacks,” *Inf. Fusion*, vol. 10, no. 1, pp. 107–121, Jan. 2009.

[165] D. Yu and D. Frincke, “Improving the quality of alerts and predicting intruder’s next goal with Hidden Colored Petri-Net,” *Comput. Networks*, vol. 51, no. 3, pp. 632–654, Feb. 2007.

[166] J. Arshad, P. Townend, and J. Xu, “A novel intrusion severity analysis approach for Clouds,” *Futur. Gener. Comput. Syst.*, vol. 29, no. 1, pp. 416–428, Jan. 2013.

[167] A. Shameli Sendi, M. Dagenais, M. Jabbarifar, and M. Couture, “Real Time Intrusion Prediction based on Optimized Alerts with Hidden Markov Model,” *J. Networks*, vol. 7, no. 2, pp. 311–321, Feb. 2012.

[168] R. D. Gardner and D. A. Harle, “Methods and systems for alarm correlation,” *Proc. GLOBECOM’96. 1996 IEEE Glob. Telecommun. Conf.*, vol. 1, pp. 136–140, 1996.

[169] R. Sadoddin and A. Ghorbani, “Alert Correlation Survey : Framework and Techniques,” *Proc. 2006 Int. Conf. Privacy, Secur. Trust Bridg. Gap Between PST Technol. Bus. Serv.*, pp. 1–10, 2006.

[170] Y.-H. Kim and W. H. Park, “A study on cyber threat prediction based on intrusion detection event for APT attack detection,” *Multimed. Tools Appl.*, vol. 71, no. 2, pp. 685–698, Oct. 2014.

[171] E. Pontes, A. E. Guelfi, S. T. Kofuji, A. A. A. Silva, and A. E. Guelfi, “Applying Multi-Correlation for Improving Forecasting in Cyber Security,” in *The Sixth International Conference on Digital Information Management (ICDIM)*, 2011, pp. 179–186.

[172] D. S. Fava, S. R. Byers, and S. J. Yang, “Projecting Cyberattacks Through Variable-Length Markov Models,” *IEEE Trans. Inf. FORENSICS Secur.*, vol. 3, no. 3, pp. 359–369, 2008.

[173] L. Feng, X. Guan, S. Guo, Y. Gao, and P. Liu, “Predicting the intrusion intentions by observing system call sequences,” *Comput. Secur.*, vol. 23, no. 3, pp. 241–252, May 2004.

[174] G. Zhang and J. Sun, “A Novel Network Intrusion Attempts Prediction Model Based on Fuzzy Neural Network,” *Lect. Notes Comput. Sci.*, vol. 3991, no. 2002, pp. 419–426, 2006.

[175] M. Bienkowski, M. Feng, and B. Means, “Enhancing teaching and learning through educational data mining and learning analytics: An issue brief,” *Washington, DC SRI Int.*, pp. 1–57, 2012.

[176] P. Ramasubramanian and A. Kannan, “Quickprop neural network short-term forecasting framework for a database intrusion prediction system,” *Artif. Intell. Soft Comput. - Icaisc 2004*, vol. 3070, no. 1, pp. 847–852, 2004.

[177] S. P. Alampalayam and A. Kumar, “Predictive Security Model using Data Mining,” in *Globecom*, 2004, no. 502, pp. 2208–2212.

[178] C. Fachkha, E. Bou-Harb, and M. Debbabi, “Towards a Forecasting Model for Distributed Denial of Service Activities,” *2013 IEEE 12th Int. Symp. Netw. Comput. Appl.*, pp. 110–117, Aug. 2013.

[179] H. Park, S.-O. D. Jung, H. Lee, and H. P. In, “Cyber weather forecasting forecasting unknown internet worms using randomness analysis,” *IFIP Adv. Inf. Commun. Technol.*, vol. 376, pp. 376–387, 2012.

[180] Z. Ghahramani, “An Introduction to Hidden Markov Models and Bayesian Networks,”

Int. J. Pattern Recognit. Artif. Intell., vol. 15, no. 1, pp. 9–42, 2001.

[181] P. Baruah and R. B. Chinnam, “HMMs for diagnostics and prognostics in machining processes,” *Int. J. Prod. Res.*, vol. 43, no. 6, pp. 1275–1293, 2005.

[182] C. Lai-cheng, “A High-efficiency Intrusion Prediction Technology Based on Markov Chain,” in *International Conference on Computational Intelligence and Security Workshops*, 2007, pp. 522–525.

[183] C. Ishida, Y. Arakawa, I. Sasase, and K. Takemori, “Forecast techniques for predicting increase or decrease of attacks using Bayesian inference,” in *Communications, Computers and signal Processing, 2005. PACRIM. 2005 IEEE Pacific Rim Conference on*, 2005, pp. 450–453.

[184] H. Liu and H. Motoda, *Feature Extraction, Construction and Selection: A Data Mining Perspective*. Springer Science & Business Media, 1998.

[185] I. Guyon and A. Elisseeff, “An introduction to feature extraction,” in *Feature Extraction, Foundations and Applications*, Springer-Verlag Berlin Heidelberg, 2006, p. 24.

[186] P. A. Watters, S. McCombie, R. Layton, and J. Pieperzyk, “Characterising and predicting cyber-attacks using the Cyber Attacker Model Profile (CAMP),” *J. Money Laund. Control*, vol. 15, no. 4, pp. 430–441, 2012.

[187] M. C. Waxman, “Cyber-Attacks and the Use of Force: Back to the Future of Article 2(4),” *YALE J. Int. LAW*, vol. 36, pp. 421–458, 2011.

[188] B. N. Garrett, “Taming the Wild Wild Web : Twenty-First Century Prize Law and Privateers as a Solution to Combating Cyber-Attacks,” *Univ. Cincinnati Law Rev.*, vol. 81, no. 2, pp. 684–706, 2013.

[189] P. Wood, B. Nahorney, K. Chandrasekar, S. Wallace, and K. Haley, “INTERNET SECURITY THREAT REPORT,” *Symantec Corp.*, vol. 19, no. April, 2014.

[190] M. Tomaso, “BP Fights Off Up to 50,000 Cyber-Attacks a Day: CEO,” *http://www.cnbc.com/*. [Online]. Available: http://www.cnbc.com/id/100529483#. [Accessed: 19-Nov-2014].

[191] D. Chinn, J. Kaplan, and A. Weinberg, “Risk and responsibility in a hyperconnected world : Implications for enterprises,” *McKinsey Co.*, 2014.

[192] S. X. Wu and W. Banzhaf, “The use of computational intelligence in intrusion detection systems: A review,” *Appl. Soft Comput.*, vol. 10, no. 1, pp. 1–35, Jan. 2010.

[193] H. Wang and H. Zhou, “The Research of Intrusion Detection System in Cloud Computing Environment,” *Adv. Multimedia, Softw. Eng. Comput.*, vol. 1, pp. 45–49, 2012.

[194] A. Ginsburg, L. J. Santos, E. Scoboria, K. Scoboria, and J. Yeoh, “The Notorious Nine: Cloud Computing Top Threats in 2013,” *Cloud Security Alliance*, no. February. pp. 1–14, 2013.

[195] K. Haslum, A. Abraham, and S. Knapskog, “DIPS: A Framework for Distributed Intrusion Prediction and Prevention Using Hidden Markov Models and Online Fuzzy Risk Assessment,” *Third Int. Symp. Inf. Assur. Secur.*, pp. 183–190, Aug. 2007.

[196] K. Tabia and P. Leray, “Bayesian Network-Based Approaches for Severe Attack Prediction and Handling IDSs` Relaibility,” in *13th International Conference, IPMU 2010, Dortmund, Germany, June 28–July 2, 2010. Proceedings, Part II*, 2010, pp. 632–642.

[197] S. S. S. Sindhu, S. Geetha, S. S. Sivanath, and a. Kannan, “A Neuro-genetic ensemble Short Term Forecasting Framework for Anomaly Intrusion Prediction,” *2006 Int. Conf. Adv. Comput. Commun.*, pp. 187–190, Dec. 2006.

[198] “KDD-CUP-99 Task Description.” [Online]. Available: https://kdd.ics.uci.edu/databases/kddcup99/task.html. [Accessed: 27-Apr-2015].

[199] G. Poojitha, K. Kumar, and P. JayaramiReddy, “Intrusion Detection using Artificial Neural Network,” *Second Int. Conf. Comput. Commun. Netw. Technol.*, pp. 1–7, 2010.

[200] C. Tang, Y. Xie, B. Qiang, X. Wang, and R. Zhang, “Security Situation Prediction Based on Dynamic BP Neural with Covariance,” *Adv. Control Eng. Inf. Sci.*, vol. 15, pp. 3313–3317, 2011.

[201] V. Jaiganesh, S. Mangayarkarasi, and P. Sumathi, “Intrusion Detection Systems: A

Survey and Analysis of Classification Techniques," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 2, no. 4, pp. 1629–1635, 2013.

[202] E. Pontes, P. Lsi, and S. Paulo, "IFS - Intrusion Forecasting System Based on Collaborative Architecture," in *Digital Information Management, 2009. ICDIM 2009. Fourth International Conference on*, 2009, pp. 216–221.

[203] L. Grunske and D. Joyce, "Quantitative risk-based security prediction for component-based systems with explicitly modeled attack profiles," *J. Syst. Softw.*, vol. 81, no. 8, pp. 1327–1345, Aug. 2008.

[204] H. Park and H. Lee, "Detecting Unknown Worms Using Randomness Check," *Inf. Networking. Adv. Data Commun. Wirel. Networks*, vol. 3961, pp. 775–784, 2006.

[205] M. Bailey, E. Cooke, F. Jahanian, A. Myrick, S. Sinha, and A. Arbor, "Practical Darknet Measurement," in *2006 IEEE Conference on Information Sciences and Systems*, 2007, pp. 1496–1501.

[206] M. Abdlhamed, K. Kifayat, Q. Shi, and W. Hurst, "A System for Intrusion Prediction in Cloud Computing," in *proceedings of The International Conference on Internet of things and Cloud Computing (ICC 2016), University of Cambridge, Uk*, 2016, pp. 1–9.

[207] G. K. Haan, "Detection of Portscans Using IP Header Data," in *2nd Twente Bachelor Referaat Conference*, 2005.

[208] A. Wald, "Sequential Tests of Statistical Hypotheses," *Ann. Math. Stat.*, vol. 16, no. 2, pp. 117–186, 1945.

[209] Y. Chabchoub, C. Fricker, and P. Robert, "Improving the detection of on-line vertical port scan in IP traffic," in *7th International Conference on Risks and Security of Internet and Systems, CRiSIS 2012*, 2012, pp. 1–6.

[210] X. Zhang, J. Knockel, and J. R. Crandall, "Original SYN: Finding machines hidden behind firewalls," in *Proceedings - IEEE INFOCOM*, 2015, vol. 26, pp. 720–728.

[211] M. Mahoney and P. K. Chan, "PHAD: Packet header anomaly detection for identifying hostile network traffic," 2001.

[212] J. Gadge and A. A. Patil, “Port scan detection,” in *Proceedings of the 2008 16th International Conference on Networks, ICON 2008*, 2008, pp. 1–6.

[213] M. Dabbagh, A. J. Ghandour, K. Fawaz, W. El-Hajj, and H. Hajj, “Slow port scanning detection,” in *Proceedings of the 2011 7th International Conference on Information Assurance and Security, IAS 2011*, 2011, pp. 228–233.

[214] T. Li, S. Chen, W. Luo, and M. Zhang, “Scan detection in high-speed networks based on optimal dynamic bit sharing,” in *Proceedings - IEEE INFOCOM*, 2011, pp. 3200–3208.

[215] C. Gates, J. J. McNutt, J. B. Kadane, and M. I. Kellner, “Scan Detection on Very Large Networks Using Logistic Regression Modeling,” in *11th IEEE Symposium on Computers and Communications (ISCC’06)*, 2006, pp. 402–408.

[216] J. Mai, A. Sridharan, C. N. Chuah, H. Zang, and T. Ye, “Impact of packet sampling on portscan detection,” *IEEE J. Sel. Areas Commun.*, vol. 24, no. 12, pp. 2285–2298, 2006.

[217] A. Sridharan, T. Ye, and S. Bhattacharyya, “Connectionless port scan detection on the backbone,” in *Proceedings of the IEEE International Performance, Computing, and Communications Conference*, 2006, pp. 567–576.

[218] X. Kuai, Z.-L. Zhang, and S. Bhattacharyya, “Profiling Internet Backbone Traffic: Behavior Models and Applications,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 169–180, 2005.

[219] J. Treurniet, “A network activity classification schema and its application to scan detection,” *IEEE/ACM Trans. Netw.*, vol. 19, no. 5, pp. 1396–1404, 2011.

[220] H. Kim, S. Kim, M. A. Kouritzin, and W. Sun, “Detecting network portscans through anomaly detection,” in *Proceedings of SPIE Vol. 5429*, 2004, pp. 254–259.

[221] Committee on National Security Systems, “National Information Assurance (IA) glossary,” 2015.

[222] T. Nowakowski, S. Werbińska-Wojciechowska, and M. Chlebus, “Supply chain vulnerability assessment methods—possibilities and limitations,” in *Safety and*

Reliability of Complex Engineered Systems., 2015, pp. 1667–1678.

[223] R. Ross, M. McEvilley, and J. C. Oren, “Systems Security Engineering Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems,” 2016.

[224] L. Lowis and R. Accorsi, “Vulnerability analysis in SOA-based business processes,” *IEEE Trans. Serv. Comput.*, vol. 4, no. 3, pp. 230–242, 2011.

[225] G. Cormode, M. Garofalakis, P. J. Haas, and C. Jermaine, “Synopses for Massive Data: Samples, Histograms, Wavelets, Sketches,” *Found. Trends Databases*, vol. 4, no. 1–3, pp. 1–294, 2011.

[226] F. E. Grubbs, “Procedures for Detecting Outlying Observations in Samples,” *Technometrics*, vol. 11, no. 1, pp. 1–21, 1969.

[227] E. Eddy and C. Walker, “Intrusion detection through traffic analysis from the endpoint using Splunk Stream,” 2017.

[228] P. Flajolet, É. Fusy, and O. Gandouet, “HyperLogLog : the analysis of a near-optimal cardinality estimation algorithm,” in *2007 Conference on Analysis of Algorithms, AofA 07*, 2007, pp. 127–146.

[229] A. Metwally, D. Agrawal, and A. El Abbadi, “Why Go Logarithmic if We Can Go Linear? Towards Effective Distinct Counting of Search Traffic,” *Edbt’08*, pp. 618–629, 2008.

[230] P. Flajolet, O. Gandouet, C. Morales, and P. Welke, “Understanding the HyperLogLog : a Near-Optimal Cardinality Estimation Algorithm,” in *The 3rd Computer Science Conference for University of Bonn Students (CSCUBS)*, 2016, pp. 72–85.

[231] K.-Y. Whang, B. T. Vander-Zanden, and H. M. Taylor, “A linear-time probabilistic counting algorithm for database applications,” *ACM Trans. Database Syst.*, vol. 15, no. 2, pp. 208–229, 1990.

[232] C. Estan, G. Varghese, and M. Fisk, “Bitmap algorithms for counting active flows on high-speed links,” *IEEE/ACM Trans. Netw.*, vol. 14, no. 5, pp. 925–937, 2006.

[233] A. Chen, J. Cao, L. Shepp, and T. Nguyen, "Distinct counting with a self-learning bitmap," *J. Am. Stat. Assoc.*, vol. 106, no. 495, pp. 879–890, 2011.

[234] G. Cormode, M. Datar, P. Indyk, and S. Muthukrishnan, "Comparing data streams using Hamming norms (how to zero in)," *IEEE Trans. Knowl. Data Eng.*, vol. 15, no. 3, pp. 529–540, 2003.

[235] F. Giroire, "Order statistics and estimating cardinalities of massive data sets," *Discret. Appl. Math.*, vol. 157, no. 2, pp. 406–427, 2009.

[236] G. Huston, "Anatomy: A Look Inside Network Address Translators," *Internet Protoc. J.*, vol. 7, no. 3, pp. 2–32, 2004.

[237] E. Eskin, "Anomaly detection over noisy data using learned probability distributions," in *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000, pp. 255–262.

[238] C. Manikopoulos and S. Papavassiliou, "Network intrusion and fault detection: A statistical anomaly approach," *IEEE Commun. Mag.*, vol. 40, no. 10, pp. 76–82, 2002.

[239] T. Abbes, A. Bouhoula, and M. Rusinowitch, "Efficient decision tree for protocol analysis in intrusion detection," *Int. J. Secur. Networks*, vol. 5, no. 4, pp. 220–235, 2010.

[240] C. Wagner, J. François, R. State, and T. Engel, "Machine learning approach for IP-flow record anomaly detection," in *NETWORKING'11 Proceedings of the 10th international IFIP TC 6 conference on Networking*, 2011, vol. PART 1, pp. 28–39.

[241] B. Balajinath and S. V. Raghavan, "Intrusion detection through learning behavior model," *Comput. Commun.*, vol. 24, no. 12, pp. 1202–1212, 2001.

[242] M. Amini, R. Jalili, and H. R. Shahriari, "RT-UNNID: A practical solution to real-time network-based intrusion detection using unsupervised neural networks," *Comput. Secur.*, vol. 25, no. 6, pp. 459–468, 2006.

[243] K. Ilgun, R. A. Kemmerer, and P. A. Porras, "State Transition Analysis: A Rule-Based Intrusion Detection Approach," *IEEE Trans. Softw. Eng.*, vol. 21, no. 3, pp. 181–199, 1995.

[244] P. Naldurg, K. Sen, and P. Thati, “A Temporal Logic Based Framework for Intrusion Detection,” *Form. Tech. Networked Distrib. Syst. – FORTE 2004*, vol. 3235, pp. 359–376, 2004.

[245] D. Bayarjargal and G. Cho, “Detecting an Anomalous Traffic Attack Area based on Entropy Distribution and Mahalanobis Distance,” *Int. J. Secur. its Appl.*, vol. 8, no. 2, pp. 87–94, 2014.

[246] J. Santiago-Paz and P. Velarde-Alvarado, D. Torres-Román, “Detecting anomalies in network traffic using Entropy and Mahalanobis distance,” in *CONIELECOMP 2012, 22nd International Conference on Electrical Communications and Computers*, 2012, pp. 86–91.

[247] A. Jamdagni, Z. Tan, P. Nanda, X. He, and R. P. Liu, “Mahalanobis Distance Map approach for Anomaly Detection of web-based attacks,” in *8th Australian Information Security Management Conference, AISM*, 2010, pp. 8–17.

[248] Oracle, “4.1 Oracle VM Server for x86 Supported Guest Operating Systems,” *Oracle*, 2015. [Online]. Available: http://docs.oracle.com/cd/E50245_01/E50246/html/vmrns-guest-os-x86.html.

[249] J. Broad and A. Bindner, *Hacking with Kali*. 2014.

[250] The Tcpdump Team, “TcpDump,” *Tcpdump and libpcap*, 2017. [Online]. Available: https://www.tcpdump.org/index.html#.

[251] G. Combs. and Others, “Wireshark,” *Wireshark Developer’s Guide*, 2018. [Online]. Available: https://www.wireshark.org/.

[252] “1999 DARPA Intrusion Detection Evaluation Data Set,” *Lincoln Laboratory of the Massachusetts Institute of Technology*, 1999. [Online]. Available: https://www.ll.mit.edu/ideval/data/1999data.html.

[253] “CYBER SYSTEMS AND TECHNOLOGY - DARPA Intrusion Detection Data Sets.” [Online]. Available: https://www.ll.mit.edu/ideval/data/. [Accessed: 08-Jun-2017].

[254] M. V Mahoney and P. K. Chan, “An Analysis of the 1999 DARPA/Lincoln Laboratory

Evaluation Data for Network Anomaly Detection," *Proc. Sixth Int. Symp. Recent Adv. Intrusion Detect.*, vol. 2820, no. Ll, pp. 220–237, 2003.

[255] C. Thomas, V. Sharma, and N. Balakrishnan, "Usefulness of DARPA dataset for Intrusion Detection System Evaluation," in *SPIE 6973, Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security,* 2008, pp. 69730–69738.

[256] FIRST.org, "Common Vulnerability Scoring System v3.0: Examples," *FIRST Public\tions,* 2017. [Online]. Available: https://www.first.org/cvss/examples.

[257] "CERT NetSA Security SuiteMonitoring for Large-Scale Networks," *Carnegie Mellon University Software Engineering Institute*. [Online]. Available: https://tools.netsa.cert.org/silk/index.html. [Accessed: 03-Dec-2017].

[258] Y. Chabchoub, R. Chiky, and B. Dogan, "How can sliding HyperLogLog and EWMA detect port scan attacks in IP traffic ?," *EURASIP J. Inf. Secur.*, vol. 5, pp. 1–11, 2014.

[259] J. M. Lucas and M. S. Saccucci, "Exponentially Weighted Moving Average Control Schemes: Properties and Enhancements," *Technometrics,* vol. 32, no. 1, pp. 1–12, 1990.

[260] EDGESCAN^{TM}, "2018 VULNERABILITY STATISTICS REPORT," 2018.

[261] B. K. Porter, "RPC-DCOM Vulnerability & Exploit," 2003.

[262] M. Jouini, L. B. A. Rabai, and A. Ben Aissa, "Classification of security threats in information systems," *Procedia Comput. Sci.,* vol. 32, pp. 489–496, 2014.

[263] W. Xing-Zhu, "Network intrusion prediction model based on RBF features classification," *Int. J. Secur. Its Appl.,* vol. 10, no. 4, pp. 241–248, 2016.

Appendix A

This appendix to show the main MatLab function of our algorithm. The function called ‘`calmahdist3`’ and it takes ‘`set`’ as input and calculates the outliers, then detect the scanners. The main function starts with output of the counting process ‘`set`’ and call the ‘`deleteoutliers`’ that separates the outlier observations and returns normal observations. Then the function calculates the threshold and decide if the observation is scanners if exceeds the threshold. We used to publicly available codes for the Hyperloglog that we used for counting and Grubb test for outlier detection, as it is not the author work we will include only the main function MatLab code as following:

```
function [outl,nset, info] = calmahdist3( set )
set.Properties.VariableNames{1} = 'IP'; % changing the ip column name
set.Properties.VariableNames{2} = 'ports'; % changing the ports column name
set.Properties.VariableNames{3} = 'time';  % changing the time column name
% orgset = set; % save the original set
% copy the ports sketches into variable to calculate the outliers
y = set.ports;
[~, ind, ~] = deleteoutliers(y, 0.5); % calculating the outliers
if isempty(ind)           % if index has no value then there is no outlier
disp('No outliers ... try with another significant factor')
else
x = set((ind), :);       % saving the outliers with all info in outlier
table
set((ind), :)= [] ;      % deleting the outliers from the original table
nset = set ;             % storing the input set after removing outliers
tmpout = [x.ports, x.time];          % removing the IP column from the
outliers to prepare to compute Euclidean Distance
tmptab = [set.ports, set.time];      % removing IP from original table

setstd = std2(tmptab);    % calculating the standard deviation of input set
thre = 3 * setstd;                   % calculate the threshold
setpomean = mean(set.ports);
settimean = mean(set.time);
setcen = [setpomean, settimean]
% checking the distance of every outlier to threshold
  jc = numel(ind);             % store the number of outliers in identifier
    euc = 0;
  fprintf('summary: %d points considered Outliers, based on Euclidean
Distance and Threshold the judge for each point is:\n', jc)
for i = 1:jc
    tmpout1 = tmpout(i,:);
    tmeuc = [tmpout1; setcen];
  euc = pdist(tmeuc,'euclidean')      % calculate the Euclidean distance
between oultiers and normal set
     ms = x.IP{i};
     if euc > thre
      fprintf('the IP: %s is scan or malicious IP distanced %s from normal
threshold %s  \n', ms, euc, thre)
    elseif euc == thre
      fprintf('%s malicious IP as it reached the threshold  \n', ms)
     else
       fprintf('the IP: %s is normal IP  \n', ms)
     end
    end
outl =x;
end
end
```