DDoS Defence for Service Availability in Cloud Computing

Opeyemi Ayokunle Osanaiye

Supervisors:

A/Prof. Mqhele Dlodlo

A/Prof. Kim-Kwang Raymond Choo

Thesis Presented for the Degree of

DOCTOR OF PHILOSOPHY

In the Department of Electrical Engineering

UNIVERSITY OF CAPE TOWN

July 2016

“The only secure computer is one that is turned off, locked in a safe, and buried 20 feet down in a secret location – and I’m not completely confident of that one, either.”

- Schneier, 1995

Declaration

I declare that this thesis is my own unaided work. Where collaboration with other people has taken place, or material generated by other researchers is included, the parties and/or materials are indicated in the acknowledgement; or they are explicitly stated with references, where appropriate.

This work is being submitted for the degree of Doctor of Philosophy in Electrical Engineering, at the University of Cape Town. It has not been submitted before for any degree or examination in any other university.

Signature of Author …….

Cape Town

July 2016

Dedication

Dedicated to Almighty God, the Author and Finisher of my faith

Acknowledgement

First and foremost, I would like to express my profound gratitude to almighty God, who has kept and provided for me, throughout the duration of my PhD studies.

I would also like to thank my supervisors A/Prof. Mqhele Dlodlo and A/Prof. Kim-Kwang Raymond Choo for their immense contributions, support, guidance and advice. Your input and constructive criticism during the research process, the writing and the submission are highly appreciated; and this is hereby acknowledged.

To my wonderful parents, Prof. and Mrs P.A. Osanaiye, I thank you from the bottom of my heart for the continuous faith you have shown in me, and also for your immense contributions in making me who I am today. Your support, both morally and financially, is highly appreciated. To my siblings and entire family, thank you for all your prayers and words of encouragement all through my PhD journey. Your kind words have kept me going; and I sincerely appreciate this.

I am extremely grateful to the University of Cape Town for the partial funding received towards my international exchange visit, and also to the University of South Australia for hosting me. To all the members of my research groups, COMMED and CRG in the University of Cape Town and the Information Alliance Research Group in the University of South Australia, thank you all for your immense contributions towards the improvement of the quality of my research output.

To all my friends in Cape Town, Adelaide, Lagos, and all over the world, who have contributed towards my PhD in one way or another, I owe you all a great deal. Thank you all for your support; and God bless you!

Table of Contents

List of Table

List of Figures

Abstract

Cloud computing presents a convenient way of accessing services, resources and applications over the Internet by shifting the focus of industries and organizations from the deployment and day-to-day running of their IT facilities, to provide an on-demand, self-service, and pay-as-you-go business model. Despite its increased popularity, ensuring security and availability of data, resources and services remains an ongoing research challenge. Distributed Denial of Service (DDoS) attacks are not a new threat but they remain a major security challenge in achieving a secure and guaranteed service and resources in cloud computing. Mitigating DDoS attack in cloud computing presents a new dimension to the solutions proffered in traditional computing, therefore, this work proposes DDoS defence solutions that identify and classify packet traffic as either legitimate or malicious, based on its attributes.

This thesis has three objectives. Firstly, it investigates a major attribute of DDoS attack, the spoofing of source IP address that hides its identity to disallow easy traceback or deceive the cloud provider to enjoy certain services accrued to a trusted host. Secondly, due to the increased number and sophistication of DDoS attacks against cloud services and the magnitude of traffic that needs to be processed, the analysis of feature selection methods and classification techniques was carried out. Feature selection has been identified as a pre-processing phase in cloud DDoS attack defence that could potentially increase the classification accuracy and reduce the computational complexity, by identifying important features from the original dataset, during supervised learning. Finally, this thesis studies the packet inter-arrival time (IAT) feature of traffic traces, in order to determine the presence of an attack using a change-point detection.

The DDoS attack pattern is detected by leveraging on the fact that most DDoS attacks are automated, thus exhibiting similar patterns.

The main contributions are as follows: (i) This thesis proposes an IP spoofing detection technique that uses a passive and active host-based operating system (OS) fingerprinting to detect the true source of a packet during a spoofed DDoS attack; (ii) this thesis proposes an ensemble-based multi-filter feature selection (EMFFS) method that combines the output of four filter methods to achieve an optimum selection, and a decision-tree classifier to detect DDoS attacks; and (iii) this thesis proposes a

change-point monitoring algorithm to detect DDoS flooding attacks against cloud services, by examining the packet IAT. A DDoS attack pattern is distinguished from normal traffic by using cumulative sum algorithm (CUSUM).

The results obtained show a high detection rate and classification accuracy, when compared with other classification techniques in the literature.

Chapter 1: General Introduction

The emergence of cloud computing has revamped the landscape of computing by presenting a change in the way information and communication technology (ICT) services are designed, developed, deployed, scaled, maintained, reformed and paid for. From the introduction of the first generation of computers, the current cloud computing technology has changed the computing paradigm by combining the existing computing concepts, such as service-oriented architecture (SOA), grid computing, virtualization, web 2.0, and other related technologies that can be accessed via an Internet-enabled device to give on-demand, scalable and reliable computing resources [1]. The term "cloud" was born from its remote infrastructure and services deployment, which can be seamlessly accessed by organisations or individual users from any part of the world – using an Internet-enabled "thin" client [2].

The cloud model comprises two main players: the cloud provider that delivers cloud services and the cloud users, who are individuals or organisations that access the provided cloud services. Other possible participants would include the cloud broker and the cloud auditor [3].

The emergence of modern applications with stringent requirements has necessitated the development of a cloud computing model to deliver software, hardware, platform, and infrastructure as a service, rather than the product [17]. These service models are often described by a multiple term XaaS, with X representing the type of service offered [4]. The services can be broadly categorised into three categories: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) [21] – See Figure 1.1.

- **Software as a Service (SaaS)**: These are applications hosted by the provider to deliver a direct service on demand to the cloud user via a web browser. This can be deployed either on top of PaaS, IaaS, or directly on the physical infrastructure. With this service, the cloud user is relieved of the burden of purchasing and installing applications on their local computer (thin client). Notables among the providers of this service include Google Maps, Zoho, and Salesforce.com.
- **Platform as a Service (PaaS)**: The PaaS level provides a platform, programming language, and software tools that enable cloud users to develop,

deploy and run their own applications – without installing any of these software tools on their local computer. The PaaS can be hosted, either on the IaaS, or directly on the cloud infrastructure [5]. Among the prominent providers of this service are: Microsoft Windows Azure and Google Apps.

- **Infrastructure as a Service (IaaS)**: The IaaS provides cloud users with the fundamental computing resources, storage and network as a service, based on virtualized technology. Users can scale down (up) these services dynamically, according to their demand [6]. Key providers of this service are: Amazon EC2 and Microsoft Azure.

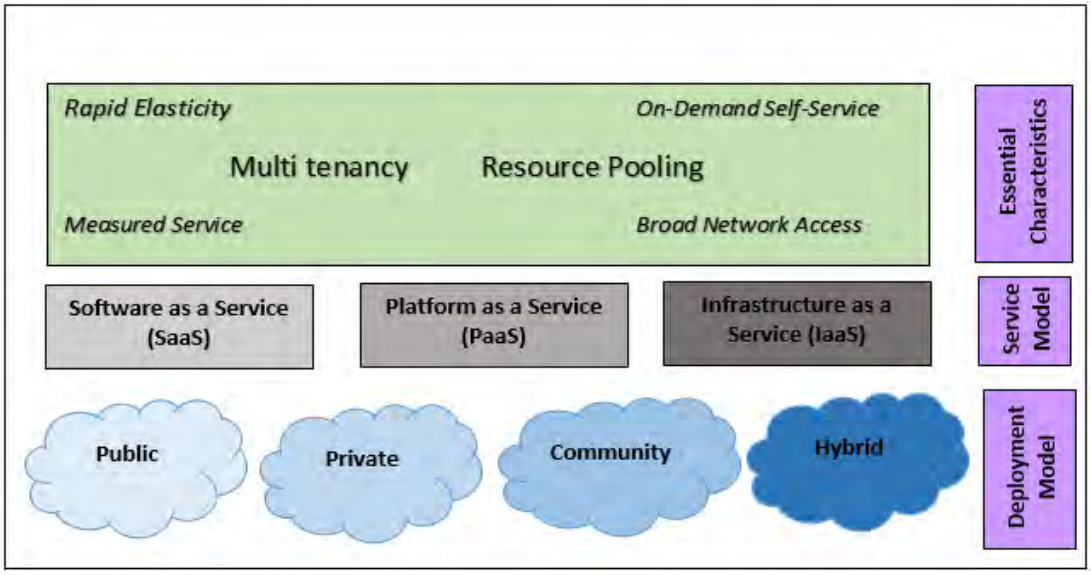

Figure 1.1 Cloud computing service model, development model, and essential features (adapted from ([22]).

The services discussed above can be deployed and offered to users either as a private, a public, a community or a hybrid cloud – See Figure 1.1.

- **Private cloud**: In a private cloud deployment model, both the infrastructure and the software, that needs to be delivered is located, owned and managed within an organisation or outsourced to a third-party cloud provider. It may exist either on the premises or off the premises; and it has the advantage of additional security, control and transparency over other deployment methods, however, it is capital-intensive.

- **Public cloud**: Here, the resources and the services are offered to the general cloud users; and they are available via the Internet browser service. This is owned and managed by a cloud provider located offsite – with no part of the cloud running on the user-end. Its services are usually low cost and pay-on-demand, however, this deployment can be subjected to security risks, as a result of multiple service users accessing the cloud over the public Internet.
- **Community cloud**: This model presents a deployment that is operated and managed by multiple organisations with similar interests, or by a third party. It can either exist on the premises or off the premises.
- **Hybrid cloud**: This deployment is made up of two or more cloud deployment models (i.e., private, public, or community). They remain unique in their own way, but bound together by standards that allow for application portability [7].

The deployment of cloud computing is primarily driven by virtualization technology that introduces a software abstraction between the hardware, the operating system, and the applications running on it [8]. This abstraction layer is called Virtual Machine Monitor (VMM) or hypervisor. The VMM acts as a controller of the hardware resources; and it enables multi-tenancy. This allows multiple OS to co-exist on the same physical hardware at the same time, in order to share the resources. The two common types of virtualization are full virtualization and paravirtualization [9]. The full virtualization offers a complete abstraction of computing resources by incorporating codes to emulate the underlying hardware, to enable unmodified guest OSs to run on top of the hypervisor, while paravirtualization, on the other hand, modifies the guest OS to enable communication with the hypervisor.

Due to the stringent requirements of the current cloud and the Internet of Things (IoT) applications, the deployment of cloud at the edge of the network has been proposed and this is called Fog Computing. The popularity of IoT applications and the increased digitalization of our society, where millions to billions of smart devices (e.g., in smart homes, smart cities, smart metering systems, intelligent vehicles and large-scale wireless sensor networks) are constantly exchanging information over the Internet – has resulted in large volumes of data that need to be managed and processed. To manage and process those data, cloud computing has become a popular option – due to its scalability, storage, computational and other capabilities to support the provision or de-provision of resources, according to the user requirements in real-time [197].

Therefore, in recent years, the cloud computing infrastructure has been extended from the core to the edge of the network.

Cloud computing is viewed as one of the most promising computing technologies, since it is characterised by resource pooling of virtualized computer resources, to offer a scalable IT infrastructure, workload deployment, as well as the sharing of resources. Its multi-tenancy feature enables the sharing of cloud resources among different tenants while its elasticity refers to the scaling of the resource usage, according to the users' demands. Other benefits derived from the deployment of cloud computing include: flexibility, scalability and pay per use.

1.1 Interaction between Cloud Computing, Fog Computing and the Internet of Things

Fog Computing's nomenclature was born from the fact that the Fog is a cloud close to the ground, with the intention of bringing cloud computing infrastructure closer to the Internet of Things (IoT) devices [198]. The advent of IoT has resulted in a number of use cases that generate a significant volume of data, compounding the challenges of dealing with big data from a number of geographically distributed data sources [199]. To efficiently analyse these time-sensitive data, Fog Computing, which extends the cloud to the edge of the network, was proposed. Billions of previously unconnected devices are now generating over two exabytes of data every day and it has been estimated that by 2020, 50 billion "Things" will be connected to the Internet [200].

Therefore, deploying the cloud at the edge of the network has been identified as a viable solution. Sehgal et al. [201] proposed a framework that combines IoT, cloud and Fog Computing for smart human security. This framework provides a wearable computing system by harnessing the pervasive nature of IoT, the omnipresent feature of cloud, and the extension of Fog Computing in order to provide security cover for people. In a similar vein, Yannuzzi et al. [202] integrated Fog Computing and cloud computing by considering the mobility, reliability control and actuation, in addition to scalability. The researchers, thereafter demonstrated that Fog Computing can be used as the underlying platform for IoT applications.

Suciu et al. [203] presented an architecture for secure E-health applications using big data, IoT and cloud convergence to enable a novel telemonitoring architecture. This

approach uses CloudView Exalead as a search platform that offers access to information present in the infrastructural level for search-based applications at both online and enterprise level. Cirani et al. [204] proposed a Fog node and an IoT hub, distributed on the edge of multiple networks, to enhance the network capability by implementing border router, cross-proxy, cache, and a resource directory. IoT operates in both the link layer and the application layer to enable resource discovery and seamless interactions among the applications.

Table 1.1 summarizes the features associated with Cloud Computing, Fog Computing and IoT.

Table 1.1 A summary of the features of Cloud Computing, Fog Computing and IoT.

Features	Cloud Computing	Fog Computing	Internet of Things
Target User	General Internet users	Mobile users	Stationary and mobile devices
Number of server nodes	Few	Large	Large
Architecture	Centralised	Distributed	Dense and distributed
Service Type	Global information collected worldwide	Localized information service limited to specific deployment location.	Information specific to the end device
Working Environment	Indoors with massive space and ventilation	Outdoors (i.e., streets, fields, tracks) or Indoor (i.e., home, malls, restaurants)	Outdoor and Indoor
Location awareness	No	Yes	Yes
Real-time interactions	Supported	Supported	Supported
Mobility	Limited Support	Supported	Supported
Big data and duration of storage	Months and years as it manages big data	Short duration as it transmits big data	Transient as it is the source of big data.
Major service provider	Amazon, Microsoft, IBM	Cisco IOx	ARM, Atmel, Bosch

1.2 Resource Availability in Cloud Computing

The high availability of cloud computing resources is essential; since impending attacks or the failure of its infrastructure rely on rule-of–thumb to over-provision resources in order to achieve availability [205]. Depending on the capacity of the cloud, over-provisioning might be limited and this could have a direct impact on the cost and the performance of other deployed user applications. This could result in a breach of the service level agreement (SLA): a binding agreement between providers and users, if the resource availability drops below the pre-agreed-upon threshold.

When accessing the availability of the cloud services, security, application failure, and infrastructural failure are the three main factors, which need to be considered. Security issues, such as malicious attacks from either an internal or external source can consume significant resources and network bandwidth. They can also disrupt the high availability of the cloud services to legitimate end-users (e.g., successful distributed denial of service attacks [101]).

The application and infrastructural failure of cloud components can either be physical, human, and/or operational, which can be a result of system failure, network failure, power cut, design error or a software bug.

1.3 Measuring Availability in Cloud Computing

To measure the availability in cloud computing, two key reliability metrics can be considered, namely: mean time to failure (MTTF) and mean time to repair (MTTR). During a component failure, resources and services offered are unavailable for use unless they are restored. MTTF is the average time estimated by the hardware manufacturer before a failure of the hardware module. For software, MTTF can be determined by multiplying the defect rate with the thousands of line codes executed per second. MTTF only unveils one side of the coin. To determine the time required to repair a failed component, the mean time to repair (MTTR) measurement is used [206]. For a hardware module, the MTTR is the mean time needed to replace a failed hardware; while the software MTTR can be determined by computing the time taken to reboot after detecting the software fault.

Measuring the rate of availability of Fog Computing can be determined by dividing the available service time by the total time.

$$\text{Availability} = \text{MTTF} / (\text{MTTF}+\text{MTTR})$$

1.4 Cloud Security Challenges

Despite the numerous benefits that cloud computing offers, some of its distinct features expose it to security and reliability risks. An example of such challenges faced by cloud providers is the provision of a high-availability cloud, which has hindered its general adoption [10]. The same way cloud computing has enjoyed the combined

benefits of related technologies, so also did it inherit their security issues [20] with regard to confidentiality, integrity and availability. Confidentiality in the cloud involves granting only authorized users permission to access the protected data. However, due to its resource sharing and multi-tenancy feature, the cloud environment can be exposed to data confidentiality threats [18]. Integrity in the cloud, on the other hand, involves the protection of the cloud against unauthorized data falsification, deletion, and modification. Lastly, the availability in the cloud involves the use of cloud services and resources by authorized users, according to their demands [91]. The effect of the non-availability of resources and services in the cloud is catastrophic; and this can lead to a total or partial disruption of service delivery, which can either be permanent or temporary.

Distributed Denial of Service (DDoS) attack has been identified as one of the biggest security threats to service availability in cloud computing. This prevents the legitimate cloud users from accessing the pool of resources provided by the cloud providers [130]. This is achieved by flooding the network bandwidth, to exhaust the server's computing resources, which subsequently results in the non-availability of cloud resources, thereby, leading to huge financial loss. High availability in cloud computing is essential. If an attack occurs without adequate defence measures, it relies on rule-of-thumb by over-provisioning the resources to achieve availability [11] before the entire resources is exhausted.

This often depends on the capacity of the cloud, which has a direct impact on the monetary cost. The availability of cloud services and resources is guided by service-level agreements (SLA) [19] between the cloud provider and the user. This is used to ensure high availability, which can be measured by the duration of uptime; and this is determined by the number of nines (i.e., from 99.9% to 99.999%).

The main motive of a DDoS attack is to overwhelm the cloud resources, and to break them down. The reasons for this attack can be attributed to competition among the cloud providers, protests by certain groups, revenge, extortion, and testing of proficiency among many others. The DDoS attacks are amplified and they can either be a direct attack or a reflection attack. While some of these attacks tend to exploit cloud’s vulnerability, such as, system weakness, protocol vulnerability, outdated patches, and misconfiguration, others simply direct malicious flood packets towards

their target. A DDoS attack on cloud computing can be from external to internal, internal to internal, or internal to external.

By external, this thesis refers to outside the cloud while internal means inside the cloud computing environment. The external attack has been identified as the most severe type of attack, therefore, research into cloud computing security with regard to DDoS attacks has focused mainly on protecting the legitimate cloud users against external attacks by malicious users [12].

1.5 DDoS Defence

DDoS defence in cloud computing can be categorized into two stages: detection and prevention. The main objective of a deployed defence measure is to detect attack as early as possible, therefore, the detection of a DDoS attack can be based on monitoring, classification, and traceback.

- Monitoring systems in the cloud computing environment can be deployed at different locations in the network, to screen the packet traffic. In core-based monitoring, a probe packet is sent by the ingress router along the same path as the data packet. This is picked up by the egress router to compute the state of the network [13]. An edge-based monitor, on the other hand, examines the packets coming into the cloud while the internal-based monitor inspects the packets within the cloud environment.
- Classifiers are often employed by a defence mechanism to sort the packet traffic. In signature-based defence, well-known DDoS attack signatures are stored and used to detect and classify any malicious traffic. The database is frequently updated, as soon as new attack signatures are detected. The main drawback of this technique is its inability to detect unknown attacks [14]. The anomaly-based classification method, on the other hand, stores the normal traffic-behavioural pattern and compares it with the incoming traffic, in order to detect an anomaly. Its advantage over signature-based is that it can detect an unknown attack.
- After detection, traceback can be carried out to locate the attacker by employing either probabilistic packet marking (PPM) or deterministic packet marking (DPM). The latter was developed to overcome the shortcomings of

the former that requires enormous packets and computational workload for the traceback process. It can also traceback a wrong source address, if the IP address is spoofed, when compared with the DPM that is simple to implement with less overhead [15].

Preventing DDoS attacks in the cloud ensures that the attack is mitigated to enable cloud users to access the services and resources. This can be categorized, according to its deployment location, into ingress, egress and route-based filtering.

- Ingress and egress filtering are preventive methods used to filter malicious packets. Ingress-based filtering denies spoofed IP address that does not match a domain prefix connected to the ingress router, while egress filtering is outbound and ensures that only allocated or assigned IP address space leaves the network [16].
- Route-based filtering removes specific routes from the routing protocol advertisement; and it filters spoofed IP packets by using route information to validate a packet.

1.6 Research Motivations

This research is motivated by the fact that cloud computing has become a scalable means by which valuable IT facilities and resources are delivered to cloud users on demand over the Internet – using the "pay-as-you-go" model. It is therefore anticipated that the various security issues pertaining to the confidentiality, integrity and availability (CIA) of cloud computing have to be addressed, in order to achieve a general adoption of the technology across the board.

Much attention has been placed on the confidentiality and integrity of the data being transferred and stored in the cloud while the availability of the services and resources it offers has not enjoyed much attention.

Ensuring service and resource availability in cloud computing is a major challenge to the general adoption of cloud. DDoS attack in the cloud has been identified as the biggest threat to cloud availability, therefore, prompt attention is needed in this area of research. This attack is perpetrated to prevent legitimate cloud users from accessing the desired service by recruiting and compromising vulnerable hosts on the Internet –

called zombies – and installing attack codes on them. These groups of zombies, under the control of the attacker, launch a combined attack on the targeted victim to exhaust its resources, such as, computing power and network bandwidth, which causes huge financial losses.

Mitigating DDoS attack comes with numerous technical difficulties, these include:

- The distributed nature of the attack: several hosts called zombies are compromised by the attacker to launch a collective distributed attack, which makes identification and defence difficult.
- Attack modification: malicious packets can modify their own pattern, in a bid to evade detection.
- Hiding IP address and mimicking legitimate traffic: most DDoS attacks spoof their IP address, in order to hide their identity from the victim and to avoid traceback, or to carry out a reflector attack. Another feature of this pattern is their attempt to mimic normal traffic – in order to avoid detection.
- False positives/False negatives: during attack detection, some legitimate packets are misclassified as malicious and discarded, while some malicious packets are also allowed into the cloud environment.

Solving the above-mentioned issues that affect the delivery of a quality cloud service has necessitated the need for thorough research in this area of study.

1.7 Research Objectives

Despite the increasing popularity of cloud services, as an effective alternative to owning and maintaining computer resources and applications, ensuring security and the availability of data remains an ongoing research challenge. Mitigating DDoS attacks presents a new dimension to solutions proffered in traditional computing – due to its architecture and features. Therefore, this thesis sets the following objectives:

- Objective 1: To analyse TCP/IP header features of incoming packets in cloud computing, in order to detect and classify spoofed IP addresses during DDoS attack.

- Objective 2: To implement a DDoS attack packet classifier that identifies legitimate packet traffic from malicious traffic with a high degree of accuracy and reduced computational complexity.
- Objective 3: To study and analyse the number and frequency of hits (i.e., patterns) of normal and malicious packet traffic in the front-end of cloud computing, and design a profile for classifying packets that deviates from a normal behavioural pattern.

1.8 Thesis Organization

The rest of the chapters in this thesis are structured as follows:

Chapter 2 – This chapter is the literature review chapter that discusses the DDoS attacks in cloud computing and their taxonomy. DDoS attacks in the cloud have been classified into application-bug level DDoS and infrastructural level DDoS. Thereafter, infrastructural level DDoS are further classified into direct and reflective attacks. Common DDoS attack tools have also been presented. To mitigate DDoS attacks in the cloud, the existing DDoS attack mitigation techniques and their taxonomy are discussed by presenting three types of DDoS detection: signature-based, anomaly-based and hybrid, and their deployment location: source-end deployment, access point deployment, intermediate-network deployment and distributed defence.

The material presented in this chapter has appeared in the publication below:

- **Osanaiye. O.,** Choo K-KR., and Dlodlo. M. "Distributed Denial of Service (DDoS) Resilience in Cloud: Review and Conceptual Cloud DDoS Mitigation Framework," *Journal of Network and Computer Applications*, vol. 67, pp. 147-165, 2016.

Osanaiye. O. reviewed and conceptualized the DDoS attack and defence taxonomy in cloud computing environment in the journal paper. Also, the manuscript was prepared by **Osanaiye. O.** and supervised by Choo. K-KR. and Dlodlo. M.

Chapter 3 – This chapter describes the IP spoofing detection algorithm that uses host-based OS fingerprinting in both the active and the passive stage – in order to detect spoofed IP during a DDoS attack. This is achieved by analysing the TCP/IP header feature of incoming traffic of connecting hosts by considering both TCP SYN and

SYN+ACK packets during the passive and the active stage. The interesting TCP/IP header feature considered in determining the OS are time to live (TTL) value, window size, the IP Don't Fragment (DF) option, and the total length. A detailed analysis and evaluation is presented in the Xen Cloud Platform testbed.

The material presented in this chapter has appeared in the following publications:

- **Osanaiye. O.** "Short paper: IP Spoofing Detection for Preventing DDoS Attack in Cloud Computing," in *Proceedings of 18th IEEE International Conference on Intelligence in Next-Generation Network (ICIN)*, February 17-19, 2015. Paris France, pp. 139-141.
- **Osanaiye. O.** and Dlodlo. M. "TCP/IP Header Classification for Detecting Spoofed DDoS Attack in Cloud Environment," in *Proceedings of the 16th IEEE International Conference on Computer as a tool EUROCON*, Sept 8th -11th 2015, Salamanca, Spain, pp. 1-6.

Osanaiye. O. conceptualised and implemented the model. He presented the results obtained, and prepared the manuscript. The research and manuscript preparation was supervised by Dlodlo. M.

Chapter 4 – This chapter describes the ensemble-based multi-filter feature selection (EMFFS) method that combines the output of four filter methods: information gain, gain ratio, chi-squared and reliefF to select the important features for DDoS attack detection. The key objective of EMFFS is to significantly reduce the feature set of the dataset during the pre-processing phase, in order to improve the classification accuracy by using a decision tree classifier. An intrusion-detection benchmark dataset, NSL-KDD, consisting of 41 features is used to evaluate the efficiency of the EMFFS in Weka.

The material presented in this chapter has appeared in the publication below:

Osanaiye. O, Cai. H., Choo K-KR., Dehghantanha. A., Xu. Z' and Dlodlo. M. "Ensemble-based Multi-Filter Feature Selection Method for DDoS Detection in Cloud Computing," *EURASIP Journal for Wireless and Communications Network*, no. 1, pp. 1-10, 2016.

The ensemble-based multi-filter feature selection method in this journal paper was conceptualised and implemented by **Osanaiye. O.;** while Ali Dehghantanha was

involved in the result verification and performance comparison analysis of the EMFFS method. Zheng Xu analysed the features of the dataset deployed. The feedback section of the EMFFS method was supervised by Haibin Cai; while the research and manuscript preparation were supervised by Choo K-KR. and Dlodlo. M.

Chapter 5 – This chapter explores the potential of the change-point detection, using the cumulative sum (CUSUM) algorithm. The change-point detection, a statistical anomaly detection technique, monitors and compares the features of observed packet sequence against the normal behavioural profile pattern obtained over a pre-determined period – with the aim of detecting a significant change when using the packet inter-arrival time attribute. This method uses both the flow-based classifier and the CUSUM algorithm to detect a DDoS flooding attack.

The material presented in this chapter has appeared in the publication below:

Osanaiye. O. Choo. K-KR. and Dlodlo. M. "Change-Point Cloud DDoS Detection using Packet Inter-Arrival time," *8th Computer Science & Electronic Engineering Conference (CEEC'16)*.

Chapter 6 – This chapter concludes the entire thesis, and discusses some possible future works.

Chapter 2: Literature Review*

2.1 Introduction

This chapter reviews common DDoS attacks targeting cloud computing; and it categorizes such attacks into application-bug level and infrastructural level attacks. Furthermore, this chapter discusses the various tools that can be used to conduct or facilitate DDoS attacks, as well as reviewing the mitigation strategies (e.g., how are DDoS mitigation strategies for cloud computing different from those designed for traditional computing?). There are two key differences in the DDoS mitigation strategy between cloud computing and traditional computing, as identified by [24]. They are as follows:

(i) The cloud provider is in control of the network and computational resources, rather than the user, as in the case of a traditional DDoS mitigation strategy.

(ii) The network infrastructure and resources are shared by the users in the cloud, which results in a reliability network segregation requirement (differing from traditional computing).

This chapter reviews 96 publications, between January 2009 and December 2015, from ScienceDirect, Springer, IEEE Xplore, Google Scholar and the ACM digital Library, using keywords, such as "DDoS in Cloud", "Detecting DDoS in Cloud Computing", "Cloud Availability", and "Cloud computing Security".

The rest of the chapter is organized as follows: Section 2.2 discusses DDoS attacks and presents an attack taxonomy. Section 2.3 describes the existing DDoS mitigation techniques and their taxonomy. Section 2.4 presents a general discussion, while Section 2.5 finally concludes the chapter.

2.2 DDoS Attacks

As remarked by Bruce Schneier, "the only secure computer is one that's turned off, locked in a safe, and buried 20 feet down in a secret location – and I'm not completely confident of that one either" [28].

*Material presented in this chapter has appeared in [130].

This has attracted the attention of security experts: both in academia and in industry. Due to the advances in technology and the tools for launching this attack, the defences proffered are not static, therefore, defenders need to stay up-to-date on the most recent attack trends and the state-of-the-art defences. As opposed to other network security attacks that seek to infiltrate or alter information, a DDoS attack is perpetrated by one or more compromised systems controlled by an attacker, who seeks to flood a predetermined target by using a series of malformed or malicious packets that overwhelm the allocated resources. The consequences of a successful attack would result in the unavailability of cloud services [29]. A recent survey [30] identified: revenge, extortion, political issues, proficiency testing by cybercriminals and competition between cloud providers, as being common motivations for DDoS attacks – See Figure 2.1. Anwar and Asad [114] highlight the consequences of DDoS attacks against a cloud data centre.

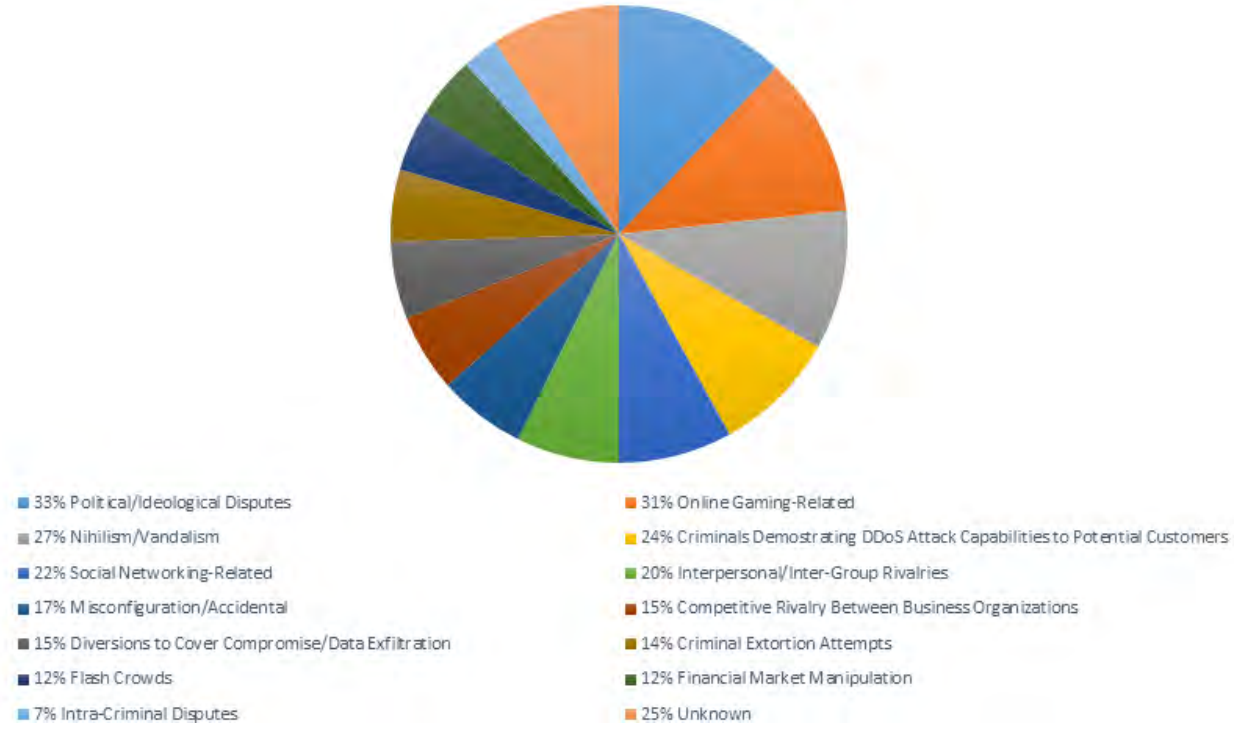

Figure 2.1 Motivations behind DDoS attacks (adapted from [31]).

2.2.1 DDoS Attack in Cloud Computing

Cloud security deployment policy is guided by the confidentiality, integrity and availability (CIA) triad model.

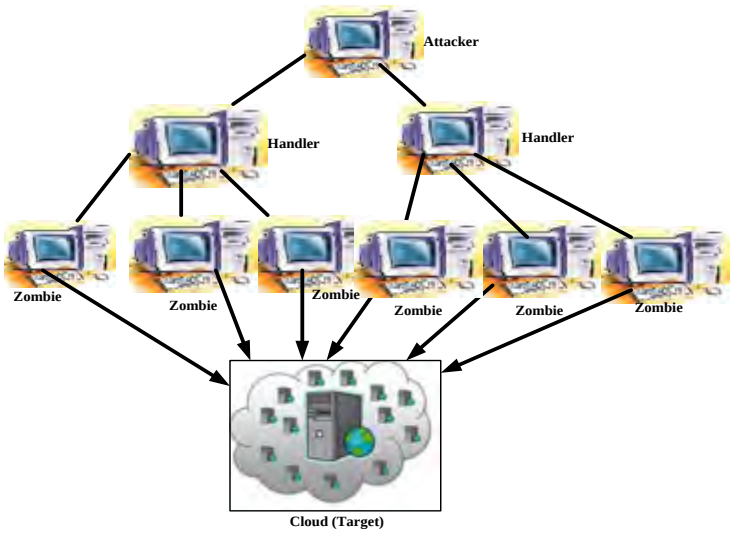

Figure 2.2 DDoS attack in cloud (adapted from [90]).

In its simplest form, DDoS can be conducted by using compromised vulnerable nodes on the Internet (also known as zombie computers) [32] – See Figure 2.2. Upon receiving malformed packets, the targeted system may not know how to handle such packets, consequently resulting in the freezing or rebooting of the system [33]. When this occurs, the cloud user will be denied access to the respective cloud services and resources.

Several cases of such DDoS attacks have been reported in recent months (e.g., [95]); and they can take different forms, targeting different cloud components. Merlo et al. [106] explain how DDoS attacks can be carried out on a cellular network, which uses a properly crafted SIM-less device. The attacks can result in service degradation in universal mobile telecommunications system (UMTS) networks; and they can disrupt the mobile network coverage. Ficco et al. [102] describe a low-rate DDoS attack that affects the pricing model of the cloud by evading early detection to incur cost.

A similar type of attack, fraudulent resource consumption (FRC) that could exploit the utility and pricing model (pay-as-you-go) of cloud services has been suggested in [103]. A new and subtle denial of service attack on the cloud data centre infrastructure has also been reported. Palmieri et al. [104] [109] describe this attack that exploits computing resources to waste energy and increase cost. In the worst case scenario, it

achieves the denial of services – due to a power outage having exhausted the power budget. A similar type of DDoS attack against cloud computing infrastructures was presented in [110], which affects both the quality of service delivered and the energy consumption. DDoS attacks on web service hosted in cloud by using HTTP and XML, were described in [78][99].

From the reported DDoS attack cases, resources in the form of bandwidth, computer application and infrastructure are the most targeted in the cloud – in an attempt to alter service models (e.g., pricing model, business model, and security model) – to the detriment of both cloud providers and users.

2.2.2 DDoS Attack Taxonomy

There have been limited attempts to draw up a taxonomy for DDoS attacks in the cloud. Deshmukh et al. [32] grouped DDoS attacks into bandwidth and resource depletion. In [35], DDoS attacks targeting cloud web services were categorized into: oversized payload, coercive parsing, and flooding attacks; while [25] and [36] categorized DDoS attacks into infrastructural level attacks (OSI Layers 3 and 4) and application-level (OSI Layer 7) attacks. In this section, DDoS attacks have been categorized into application-bug level attacks and infrastructural level attacks, similar to the approach undertaken in [37]. It is believed that classification, based on a layered structure would simplify the reader's understanding of the attack process – See Figure 2.3.

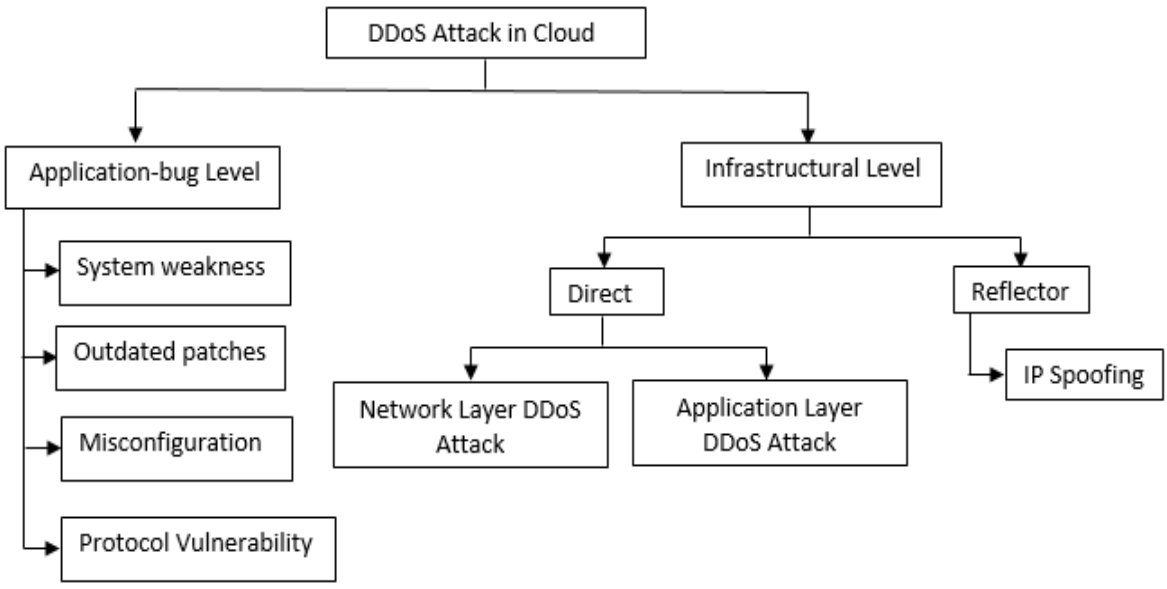

Figure 2.3 DDoS attack taxonomy in the cloud.

2.2.3 Application-bug Level DDoS

In carrying out application-bug level attacks, attackers exploit the system vulnerabilities or weaknesses to render cloud resources unavailable for users. Among the common attack vectors are: protocol vulnerability, system weakness, outdated patches, and misconfiguration. An example of this is the vulnerabilities in the protocol of target applications that can be exploited by attackers by sending specially crafted packets to overload the application, thereby crashing it. Dantas et al. [38] discussed two types of such attacks, namely: HTTP PRAGMA and HTTP POST attacks. Beitollahi and Deconinck [37] also described the ping-of-death attack, which uses a ping packet size of 65,535 bytes. The latter exceeds the maximum IPv4 packet size, therefore, when most modern operating systems try to handle such packets, they generally freeze, crash or reboot – due to buffer overflow.

2.2.4 Infrastructural Level DDoS

Infrastructural attacks (also known as flooding attacks) target cloud components, such as storage, network bandwidth, CPU circles and TCP buffers, to render them unavailable to legitimate cloud users. In infrastructural level DDoS attacks, the attackers only need the IP address of the target – without the need to exploit any vulnerability. A DDoS flooding attack can be carried out in two different forms: a direct attack and a reflector attack.

2.2.4.1 A Direct Attack

A direct attack involves the use of compromised victim hosts / zombie computers to send massive malicious packets aiming to overwhelm the target system by consuming all the available resources, resulting in the system being unavailable to legitimate users. Such attacks can be further classified into: network layer DDoS and application layer DDoS attacks.

2.2.4.1a Network Layer DDoS Attack

In carrying out a network layer DDoS attack, research has shown that the protocols that exist in the network and transport layers can be used to flood the target host [30]. Examples of these type of attacks are: TCP SYN flood, UDP flood, and ICMP flood.

TCP SYN flooding attack

The Transmission Control Protocol (TCP) is a connection-oriented protocol that exists on the transport layer of the TCP/IP model stack. The connection-oriented feature is derived from the three-way handshaking established, prior to the packet transmission between hosts. During the connection process, a SYN message is sent by the connecting host and this is acknowledged by the remote host by sending a SYN + ACK message. To complete the handshaking process, the connecting host responds with a final ACK and the connection is established between the two hosts. Attackers have exploited this connection feature by initiating half-opened connection, which exhausts the kernel memory by creating too many transmission block allocations [25]. This can be accomplished by infiltrating vulnerable nodes on the Internet, in order to carry out a co-ordinated attack. DDoS attacks using TCP SYN flooding can also be facilitated by using spoofed IP addresses. During a spoofed attack, the final ACK required to complete the connection process would not be received; as the host whose IP address was spoofed would respond with a RST flag, or the host might not exist.

Cha and Kim [39] have reported a case of successful TCP SYN-flooding attacks affecting Amazon's cloud services.

UDP flooding attack

The UDP protocol is also a transport layer protocol that is connectionless; and it is often used when the reliability of the packet transfer is not mandatory. An example of this is during the transfer of real-time applications, such as voice and video. On the Internet, UDP can also be used for online gaming and instant messages [40]. The protocol vulnerability can be exploited to launch DDoS attacks, such as flooding attacks. UDP flooding can be initiated by generating excessive amounts of UDP packets – to random ports of the cloud target [40]. The attack exploits UDP's connectionless and unreliability features by directing a high volume of malicious traffic towards the target, to fill up the response queue, thereby, preventing the responses from reaching the legitimate users [25].

The unreliable feature in UDP does not allow the target system to regulate the attackers' sending rate [25].

ICMP flooding attack

ICMP is an IP protocol, which can be used to check the current status of a host's network connectivity. Attackers have used ICMP to launch DDoS attacks in the form of smurf and ping flood attacks [25]. These are carried out by directing enormous ICMP packets to a target – with an attempt to consume the bandwidth and crash the target. Consequently, the target would not be able to respond to incoming request from legitimate users.

2.2. 4.1b Application Layer DDoS

Application layer DDoS attacks in cloud computing have continued to increase over the years, both in volume and complexity. These attacks adversely impact the productivity, the quality of service, the quality of experience, the reputation and the revenue of the cloud provider. Attacks on the application layer target cloud services by using flood packets with significant amounts of HTTP floods at high rates to overwhelm a target web server hosted in the cloud. This consumes the target cloud webserver's resources, and prevents legitimate users from accessing the target. Such application layer DDoS attacks are challenging to mitigate since such attacks consume less bandwidth, and are stealthier in nature. The attacks flood the target server with what appears to be legitimate requests [25]. Common among such attacks are HTTP flood attacks and XML flood attacks.

HTTP flood attacks

HTTP flood attacks (also known as H-DoS) are designed to flood the web servers and applications in the cloud by using malformed HTTP packets ('impersonating' HTTP GET or POST requests) [41]. Such attacks do not necessarily require a high rate of traffic flow. For example, an HTTP GET attack can be carried out by compromising several nodes on the Internet to create several request sessions to the victim in order to disable the victim. A recent report on global DDoS attack reveals that close to a quarter of current DDoS attacks target the application layer [25], and one-fifth of the HTTP DDoS attacks are HTTP GET floods.

XML Flood attacks

When requesting resources, cloud users and providers use a SOAP message to start the communication. SOAP messages work with HTTP and are written in XML because the latter is a universally acceptable language that runs on any platform [42].

X-DoS, an Extensible Markup Language DoS attack, can be carried out by using less sophisticated tools – due to its ease of implementation. The distributed version of X-DoS is known as DX-DoS. In the XML-wrapping attack on Amazon EC2 services described by Gruschka et al. [43], SOAP message request validations are exploited by changing the XML tags. Hence, any unauthorized user can have access to Amazon's EC2's services, which can be abused to send spam emails by using multiple virtual machines.

2.2.4.2 Reflector Attack

In a reflector-based DDoS attack, the attacker spoofs an IP address and sends the request to a large number of reflector hosts (see Figure 2.4). When the requests are received, the reflector hosts send the response to the target, resulting in the flooding of the target [36]. An example of this attack is a smurf attack, which is carried out by sending an ICMP echo request as a broadcast message to hosts on the Internet with a spoofed IP address (the target's IP address) [27]. These hosts amplify the attack by directing their ping response to the target. In the study of reflector attacks, Arukonda et al. [44] proposed various mitigation strategies. Other examples of reflector attack are: SYN ACK RST flood, and DNS flood [36].

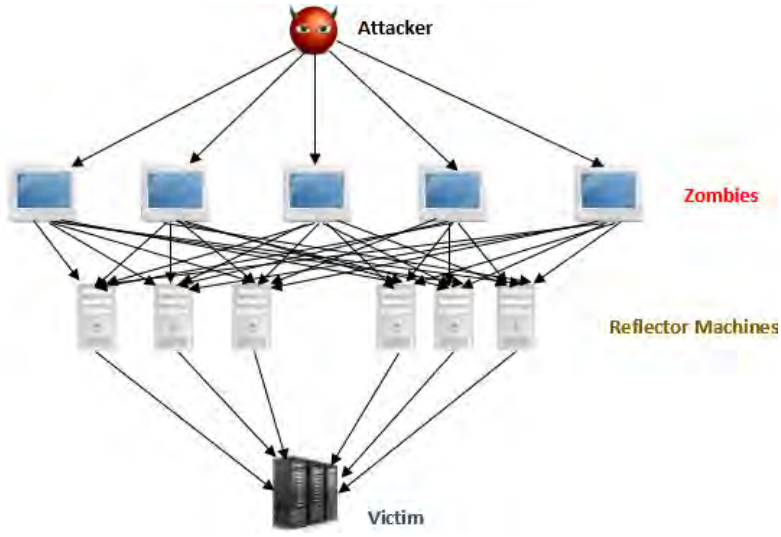

Figure 2.4 DDoS reflector attack (adapted from Beitollahi et al. [3]).

2.2.5 Other Cloud Resource Consumers

Flash crowd is when a group of legitimate packets attempt to access a resource concurrently. Ali-Eldin et al. [45] describe a flash crowd as a surge in traffic, known

as the Slashdot effect, which results in the service being unreachable by other legitimate users during the period. A Flash crowd is not a new phenomenon; and a high-profile example is the attack targeting the FIFA website during the world cup in 1998. Other related attacks targeting cloud resources, without necessarily denying legitimate user access, include the Economic Denial of Sustainability (EDoS) and the Fraudulent Resource Consumption (FRC). An EDoS attack targets the cloud service provider's billing system by fraudulently consuming resources [46]. FRC is another attack variant [47], which exploits the utility pricing model of operating in the cloud over an extended period of time.

2.2.6 DDoS Attack Tools

Recent tools that have been used by attackers include the following:

- LOIC (Low Orbit Ion Canon): LOIC, a popular tool used to launch a flooding attack, is readily available over the Internet; and it has been used by cybercriminal groups, such as Anonymous [48]. The co-ordinated attack was perpetrated by sending requests to other Internet users to join the attack via Internet Relay Chat (IRC). As the tool is automated, all that is required for an Internet user to participate is the URL or IP address of the victim. The tool would use the participating user's bandwidth to send UDP, TCP or HTTP requests to flood the target [48].
- XOIC: XOIC is a tool closely related to LOIC and it has a user-friendly graphical user interface (GUI) that can be used during a DDoS attack to a specified target IP address or port number. XOIC has three attack modes, namely: test mode, normal mode and DoS mode – with an inbuilt TCP/HTTP/UDP/ICMP messages [49].
- DDoSIM: This is a DDoS facilitating tool that creates zombie hosts with random IP addresses to launch a full TCP connection against a target host. The tool is written in C++ and can conduct HTTP DDoS, SMTP DDoS and Application-Layer DDoS [50].
- DAVOSET: This is a tool that abuses the vulnerability of the target to carry out DDoS attacks. Its latest version includes features, such as support for cookies [51].

- PyLoris: This DoS tool uses SOCKS proxies and SSL to perform DoS attacks on a target. It allows the attacker to own the HTTP request header, which would help keep the connection opened for as long as possible to exhaust the server resources, and thereby deny access to the legitimate users [52].

Other notable tools include HULK, R-U-Dead-Yet, and GoldenEye HTTP DoS tool, which can be used to generate different forms of DDoS attacks targeting cloud services. Common DDoS attack, features, and tools for perpetrating attacks are thus presented (see Table 2.1).

Table 2.1 DDoS attacks, features, and tools

Attack name	Characteristics				Tools
	Application	Infrastructure	Direct	Reflector	
SYN flooding		✓	✓		LOIC, XOIC
ICMP flooding		✓	✓		TFN, XOIC
UDP flooding		✓	✓	✓	LOIC, XOIC
HTTP (H-DoS) flooding	✓		✓		DDoSIM
XML (X-DoS) Flooding	✓		✓		DAVOSET
Ping of Death (POD)		✓	✓		Ping
Slowloris	✓		✓		PyLoris, Goloris
Zero-day*	✓	✓	✓	✓	Any tool
Smurf		✓		✓	Nemesis, Ping

*Unknown or new DDoS attacks that exploit the vulnerabilities without patch or fix.

2.3 Cloud DDoS Defences

Several DDoS defences have been proposed since the inception of the attack in 1999, when a DDoS tool, Trinoo was deployed on approximately 227 hosts to flood a single computer in the University of Minnesota [53]. A number of these proposed defences for cloud computing are based on software-defined network (SDN). Yan et al. [113] reviewed the recent SDN literature and the potential of using SDN to defeat DDoS attacks targeting cloud computing. Similarly, Wang et al. [115] examined the security impact of DDoS attack defence techniques in an enterprise network, in which SDN and cloud computing had been adopted.

Varadharajan and Tupakula [117] examined various attack scenarios on cloud hosted services and they proposed a trust-enhanced security model. This section focuses only on DDoS defences proposed for cloud services by presenting the DDoS defence taxonomy, as outlined in Figure 2.5.

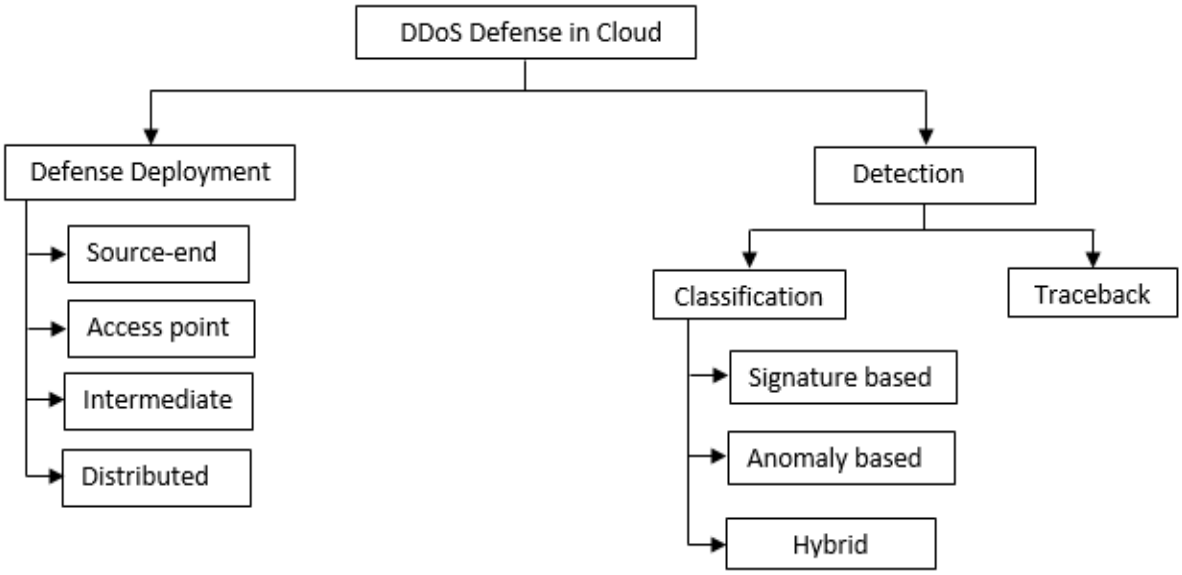

Figure 2.5 Cloud DDoS defence taxonomy.

2.3.1 Cloud DDoS Defence Deployment

DDoS defences for cloud services can be deployed in four key locations: source-end, access point, intermediate network and distributed.

2.3.1.1 Source-end Deployment

The advantages of source-end deployment include more effective protection of network resources and bandwidth. For example, defences deployed at the source of a potential attack usually use a throttling component to limit the rate of outgoing packets during such DDoS attacks [53]. This would preserve the resources of both the intermediate network and the target victim.

2.3.1.2 Access Point Deployment

Access point deployment is usually deployed in the front-end, back-end, or on each virtual machine (VM) in the cloud computing environment. The front-end is typically the administrative domain of the cloud service that serves as an interface between the cloud user and the various cloud components. In Eucalyptus, for example, this is

referred to as the cloud controller, while in Xen, it is known as dom0. DDoS defences deployed at the access point distinguish legitimate packets from malicious packets – before granting access to the cloud computing resources and services. A key limitation of this deployment is that the access point is generally not the most suitable place for filtering or rate-limiting since the bandwidth might be saturated. However, this approach is most commonly deployed, due to the ease of deployment. A popular example of access point deployment is SBTA [15].

2.3.1.3 Intermediate-network Deployment

These comprise the defences deployed on network nodes to limit the impact of DDoS attacks on the network before the attacks affect the intended target. This is achieved by imposing rate limits on the traffic passing through the nodes after comparing it against a normal profile pattern [53]. Such a deployment can be effective but it is impractical in a cloud computing environment as the nodes are not controlled by the same provider; and they are in different administrative domains. This might, perhaps, work in a private cloud deployment.

2.3.1.4 Distributed Defence

Distributed defence is a hybrid deployment model comprising source-end, access point and/or intermediate network deployments. Depending on the configuration, this deployment model can be tuned to achieve a high DDoS attack detection rate (e.g., good co-operation between various administrative domains and providers). MTF [54] is an example of a distributed defence deployment.

2.3.2 DDoS Detection

Typical DDoS detection techniques classify packet traffic as either legitimate or malicious, and can be broadly categorized into signature-based, anomaly-based and hybrid.

2.3.2.1 Signature-based Detection

The signature-based detection technique uses a set of rules and a known signature attack pattern stored in a knowledge database. Traffic patterns are monitored and compared against these existing signatures, in an attempt to detect malicious traffic (somewhat similar to a typical signature-based anti-malware solution). Signature-

based is known for its accuracy in detecting known attack signatures, provided the database is always up-to-date, however, its major flaw is its inability to detect unknown attacks or variations of the known attack signatures. This can lead to high false negatives.

A signature-based DDoS detection in the cloud was proposed by Bakshi and Yogesh [55]. They used an intrusion detection system (IDS) in VMs to counter DDoS attacks by deploying IDS sensors (SNORT) on the virtual interface of the VMware virtual ESX machine to analyse both inbound and outbound traffic in real time. The defence is designed to counter DDoS attacks in the network/transport layer, by identifying the IP addresses used for the attacks, and automatically generating an access control list to drop the entire packet from the blacklisted IP addresses. If the generated attacks are from compromised zombie machines, the approach can be configured to block such traffic and transfer the targeted application to a VM hosted in another data centre.

Lonea et al. [56] deployed a VM-based IDS, which has a graphical interface. The researchers configured the MySQL database to monitor the alerts of cloud fusion unit in the front-end of the Eucalyptus cloud architecture. Their solution used the Barnyard tool to capture such attacks while the signature-based snort was configured with predefined DDoS rules to defend against known DDoS attacks. The captured attack packets by Barnyard are stored in the binary unified file and transmitted by using a secure tunnel to a centralized MySQL database at the front-end. DDoS attacks were simulated by using Stacheldraht, a DDoS attack tool that generates an infrastructural resource depletion attack consisting of ICMP flooding, UDP flooding, and TCP SYN. Even though the system has a high detection rate, with a low false positive, the major limitation of this approach is the inability to detect unknown attacks – an inherent weakness of the signature-based approach.

Karnwal et al. [42] [57] proposed the use of a filter tree approach to defend against application layer flooding (see Figure 2.6). The five modules in the proposed approach were designed to detect and resolve XML and HTTP based DDoS attacks that occur in requests for resources when using SOAP messaging. The IP marking module uses the Flexible Deterministic Packet Marking (FDPM) scheme to mark the SOAP messages at the edge router, while the IP traceback is a logical file that stores a blacklist of IP addresses provided by the Cloud Defender. Finally, the Cloud Defender filters the attacks by going through five different stages: Sensor Filter, Hop Count

Filter, IP Frequency-Divergence Filter, Confirm legitimate-user IP Filter and Double-Signature Filter. The first four stages are used to detect HTTP DDoS attacks while the XML DDoS attack is detected in the fifth stage.

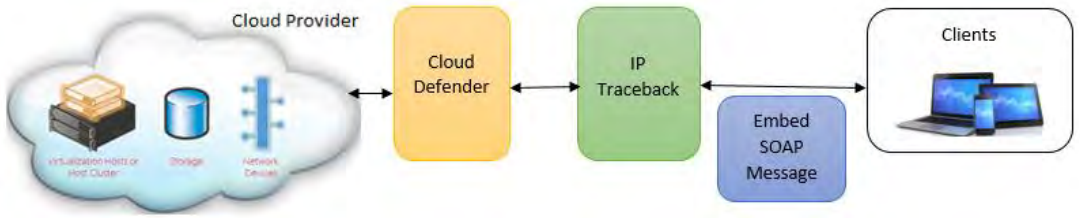

Figure 2.6 Filter tree approach against XML and HTTP DDoS attack in cloud (adapted from Karnwal et al. [42]).

Gul and Hussain [58] proposed an intrusion-detection model to handle large flow of packets by analysing and producing reports on these packets. These reports are distributed to the various parties in the cloud system. The proposed model employs multi-threading techniques to improve the IDS performance. The NIDS senses and monitors the traffic in the network to check for malicious packets. As soon as a malicious packet is detected, an alarm is sent to a third-party monitoring system that reports this to the cloud management system. The NIDS model has three modules, namely: capture and queueing module, processing/analysis module, and reporting module. This model was implemented by using .NET in a Windows environment. The evaluation carried out by the researchers suggests that a multiple-threat deployment mode is more efficient than a single threat-deployment mode.

Lo et al. [59] proposed an IDS-based mechanism distributed within the cloud environment. Alerts would be exchanged with the IDS nodes distributed throughout the cloud environment, when an attack is detected. The IDS system comprises four components, namely: co-operative operation, response and block, threshold check and alert clustering, and intrusion detection. A co-operative agent is contained in each IDS, which is used to determine whether to accept or reject the alert sent by other IDS nodes. Such an approach can mitigate attacks detected and reported by other IDS nodes.

Gupta et al. [60] proposed the use of an attack pattern detection scheme based on VM profile optimization. Rule-based detection was used to match packet during TCP SYN

flooding attacks, by extracting the threshold for rule patterns during the initial rule-establishment phase.

The advantages of the signature-based detection method include:

- Accuracy in detecting a known attack signature with a low false positive rate.
- The presence of DDoS attack labels allows the system administrator to determine the exact type of DDoS attack the victim is experiencing.

The disadvantages of signature-based detection include:

- Maintaining an up-to-date signature is an uphill and costly, if not impossible, task.
- Misrepresentation of a signature pattern would result in a high false negative rate.
- The inability to detect unknown and zero-day attacks.

2.3.2.2 Anomaly-based Detection

The anomaly-based or behavioural classification approach involves the collection of normal traffic behavioural profile patterns over a pre-determined period. Its main objective is to detect subsequent patterns that deviate from the expected behaviour. Chandola et al. [92] group anomalies into three main categories, namely: point anomalies, contextual anomalies and collective anomalies.

Point anomalies occur when an individual data instance is considered anomalous with respect to the rest of the data. A typical example of this is an application-bug level attack (see [105] for an overview of web application attacks and mitigation strategies) in packets resulting in the DoS attacks. An anomaly is referred to as being contextual if the data are anomalous in a specific context – but not in any other context. This is mainly determined by the structure of the dataset. In a collective anomaly, a group of data instances is anomalous with respect to the whole dataset. An example of this is DDoS flooding attacks, where individual data become anomalous and harmful in coordination.

The anomaly-based classification approach is typically carried out in two phases, namely: training and detection phases.

Training phase: The efficiency of anomaly detection depends on the nature of the input data during the training phase. The input comprises a collection of data instances in the form of patterns, samples and observations described by a set of attributes represented in binary, categorical or numerical type. Each data instance may consist of a single attribute (univariate) or multiple attributes (multivariate) [92]. In a multivariate data instance, the data can either be of the same type, or a combination of data types. Data labels are used to specify whether a particular instance is normal or anomalous. There are research datasets consisting of both anomalous attack labels and normal data instances. A popular (but out-of-date) example is KDD'99, which consists of approximately 4,900,000 single connection vectors. Each of the vectors contains 41 features; and these can be labelled as either attack or normal. The simulated attack falls into four categories, namely: Denial of Service (DoS), User-to-Root (U2R), Remote to Local (R2L), and a Probing attack [93].

Detection phase: Anomaly detection can exist in three modes, based on the number of labels available, namely: supervised, semi-supervised, and unsupervised.

The supervised mode assumes the availability of labelled instance training datasets for both normal and anomaly classes. This approach is used to build a predictive model for normal versus anomalous classes. Previously unseen data instances are compared with the model, to determine into which class it falls. There are two key issues with supervised anomaly detection. Firstly, anomalous instances are far fewer when compared with normal instances in training data; and secondly, the classification challenge for an anomalous class is the lack of an accurate and representative label [94]. Semi-supervised anomaly detection assumes the training data have only label instances for the normal class. They are much more practical than supervised techniques since they do not require the use of labels for anomalous classes [92]. Lastly, there is the unsupervised anomalous detection approach that does not require any training dataset, hence, it is one of the most widely deployed techniques [94]. The latter assumes that normal instances are significantly more frequent than anomalies in a typical test dataset. If this assumption is not true, the technique suffers from a high false alarm rate.

The reporting of detected anomalies is a very important aspect of anomaly detection. According to the literature, the two common detection output types are scores and labels. Using scores involves assigning an anomalous score to each data instance, in

order to reflect the degree of anomaly. A cut-off threshold is set to determine whether to accept or reject the data instance. Labels, on the other hand, involve assigning a label to each test instance: indicating either normal or anomaly.

2.3.2.2a Anomaly Detection Techniques

In categorizing anomaly detection during a DDoS attack in cloud, this thesis groups the existing techniques into five "ring" classes, based on the algorithm(s) used. Coming up with this ring scheme is not trivial as the proposed techniques are highly likely to overlap. The five classes are: machine learning, statistical anomaly detection, data mining, classifiers and artificial intelligence. (see Figure 2.7).

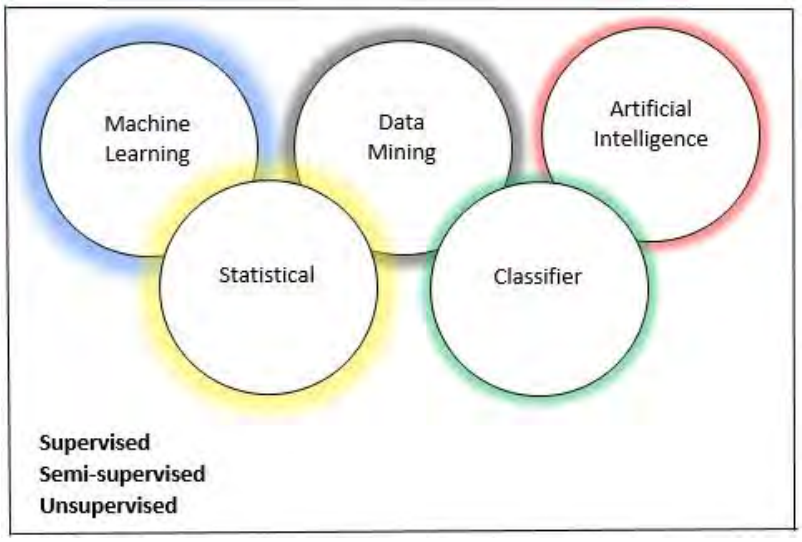

Figure 2.7 Ring classification of anomaly detection methods.

Statistical anomaly detection

In statistical anomaly detection, the statistical features of normal traffic are compiled to generate a normal traffic pattern, which will be compared with incoming traffic, in order to detect any anomalous packets. During detection, statistical inference tests (parametric or non-parametric techniques) are applied to determine the legitimacy of the behaviour [61].

Vissers et al. [64] propose a parametric technique using the Gaussian model to defend against application layer DDoS attacks on cloud services by using malicious XML content contained in a Simple Object Access Protocol (SOAP). A normal profile

model is constructed from the dataset during the initialization stage prior to activating the proxy to listen to requests.

During detection, there will be several phases. In the first phase, an HTTP header inspection will be carried out to prevent HTTP flooding. It also undertakes SOAP action check and size outlier inspection. In the next phase, the XML content is processed before checking whether the $R_{SOAPAction}$ is/are spoofed by consulting previous maps. Finally, the SOAP feature outlier detection (a filter process that evaluates each feature to its corresponding Gaussian Model) is carried out. The drawback of this model is its inability to detect request schematics resulting from new DDoS techniques, without implementing additional features.

Shamsolmoali and Zareapoor [62] proposed a statistical-based filtering system: Cloud confidence DDoS Filtering, which uses two levels of filtering. In the first level, it removes the header field of the incoming packet and compares the TTL value with the stored value in the IP to hop-count (IP2HC) table. If these values are not equal, the packet is dropped and categorized as spoofed. The second level is based on the Jensen-Shannon divergence concept, which uses a stored normal profile in its database to compare the incoming packet header information. This helps to check for information divergence. Zakarya [63] introduces an entropy-based detection technique that uses an attack packet dropping algorithm to detect DDoS attacks in the cloud computing environment. The entropy rate is used to identify the attack flow based on the distribution ratio. The anomaly detection system is deployed on each edge router, which subsequently transfers the flow to an adjacent router for a confirmatory check, when DDoS is detected. If confirmed, DDoS packets are dropped. The findings from the CloudSim simulation suggested an accuracy rate of 90%.

Girma et al. [66] propose a hybrid statistical model, which uses a covariance matrix and an entropy-based system to classify the DDoS attack pattern, by measuring heightened dependency in the data. Ismail et al. [70] used a mathematical model, as well as a covariance matrix approach to detect flooding based DoS attacks against cloud services. In the first phase of their approach, a model for profiling the normal traffic pattern was used as the baseline, by mapping captured normal traffic into a matching covariance matrix. The second phase is the intrusion detection stage, where the covariance matrix obtained from the first phase is compared with the covariance

of the currently captured traffic. The last phase, the prevention stage, implements the two earlier phases.

A confidence-based filtering (CBF) method that uses correlation characteristics was proposed by Dou et al. [71], which could be deployed in both attack and non-attack situations. During the non-attack period, a nominal profile is created by extracting an attribute pair from the network and the transport layers. The frequency of occurrence of these value pairs would be extracted and used to calculate their confidence value. The correlation characteristics that exist between these two layers were used to determine the legitimacy of a packet during the traffic flow. During the attack period, attribute value pairs of incoming packets would be collected and compared with the nominal profile to determine their confidence value in a legitimate flow. The Packet discarding strategy uses the CBF score and filtering criterion to ascertain the legitimacy of a packet, by checking whether the CBF score is above the pre-defined threshold. This would be used to determine whether the packet will be granted access to the cloud environment, or not.

Negi et al. [72] presented an enhanced CBF packet filtering method of Dou et al. [71] to improve the processing speed and the utilization of storage, based on the correlation pattern.

More recently in 2015, Wang et al. [24] proposed a cloud DDoS attack defence (DaMask). DaMask employs a highly programmable network monitoring technique that detects any attack and responds by using a flexible control structure. DaMask has three layers (i.e., network switches, network controllers and network application) and two modules (i.e., an anomaly-based network attack detection module – DaMask-D – and an attack mitigation module – DaMask-M) – See Figure 2.8. When an attack is detected by DaMask-D, an alert is issued. Both the alert and the packet information will then be forwarded to the DaMask-M module. If the packet is determined to be legitimate, it will be forwarded to its destination. DaMask-M performs two functions, namely: countermeasure selection and log generation. After receiving an alert, DaMask-M decides on what countermeasure to undertake. A typical countermeasure is to drop the packet. Evaluations, using Amazon Web Service (EC2) public cloud and a private cloud running Ubuntu 12.10 with the UNB ISCX dataset, suggested that DaMask's performance is similar to a continuous-time Bayesian network detection scheme, but the DaMask reportedly has a lower computational cost.

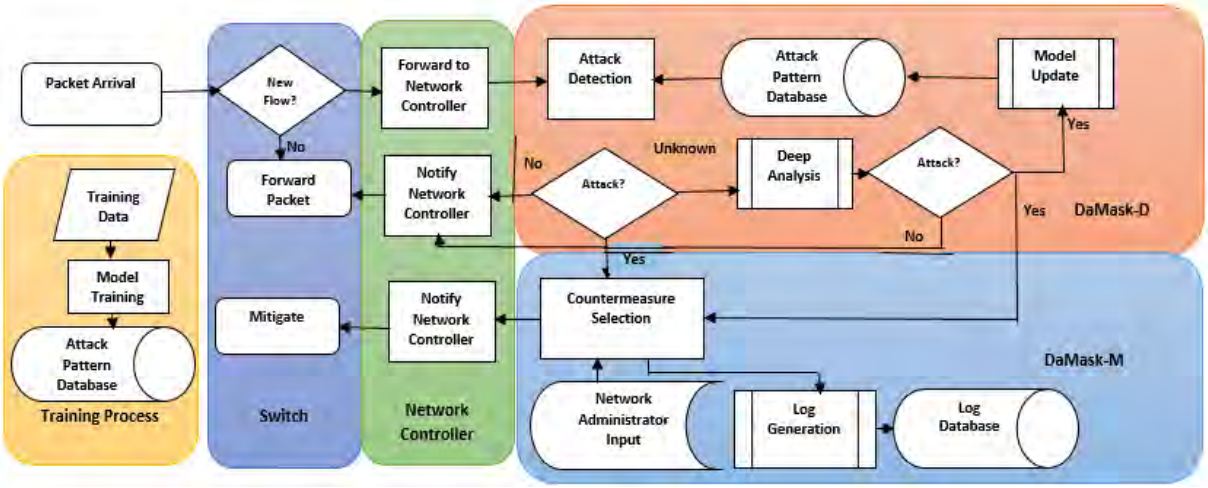

Figure 2.8 Workflow of DaMask. (adapted from Wang et al. [24]).

Bedi and Shiva [74] proposed a mechanism to secure cloud infrastructures from co-resident DoS attacks by using the game theory. This mechanism functions by modelling both the legitimate and the malicious VM behaviours that co-reside on a physical machine. A game inspired firewall defence was also modelled.

Marnerides et al. [112] have presented an anomaly detection technique, Ensemble Empirical Mode Decomposition (E-EMD), which can be used to conduct statistical characterization and the decomposition of measured signals. E-EMD can be implemented on the hypervisor level and functions by jointly considering systems and network information from every VM. The proposed approach was motivated by the fact that most monitored traffic exhibit non-linear and non-stationary properties.

In summary, the advantages and disadvantages of statistical anomaly detection approaches are as follows:

Advantages:

- Statistical anomaly detection approaches allow the learning of expected behaviour from observations – without any prior knowledge of the normal activities of the target system. This can potentially result in more accurate detection of malicious activity.
- Anomaly scores associated with statistical detection can be used as a confidence interval during decision making.

Disadvantages:

- Setting an optimal threshold, without resulting in an extreme false positive or false negative can be challenging.
- Statistical anomaly detection techniques involve assumptions and hypotheses. If not justified reasonably, this can lead to high misclassification rate.

Data mining

The significant increase in Internet traffic complicates efforts to detect a DDoS anomaly pattern. To address this challenge, Choi and Choi [41] present a data mining approach that uses a MapReduce model to mitigate the application layer HTTP GET DDoS attacks. Mapreduce is a parallel-processing model that has been used to expedite batch job operations. The proposed framework consists of three parts, as shown in Figure 2.9. The packet and log module analyses packet transmission and web server logs, and the pattern analysis module creates the attack pattern for DDoS detection. The parameters to be analysed include: CPU usage, packet size, load, and information distribution of the packet header. The detection module uses a normal behavioural pattern to detect any DDoS attacks. To evaluate the model, a mapreduce algorithm was used to measure the rates between the pattern rule and the detection time of the proposed system to external signatures. The results suggested that the proposed method performs better than Snort, since it can identify any new attack profile and it requires a shorter processing time.

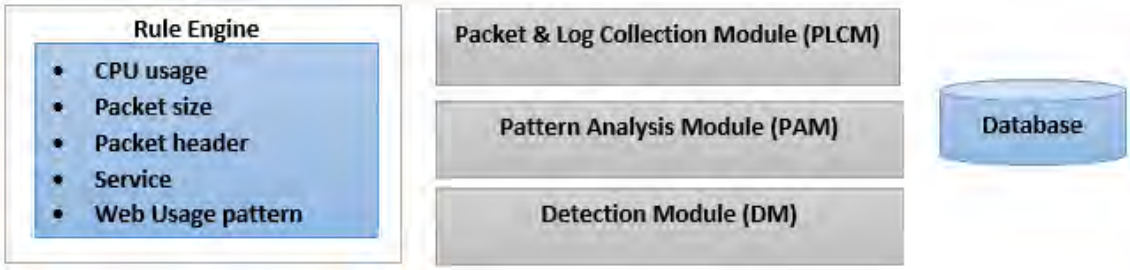

Figure 2.9 Packet analysis and control function block. (adapted from Choi et al. [41]).

In another related work to detect application layer HTTP GET flooding attacks, Choi et al. [29] used MapReduce to classify the parameters (i.e., CPU usage, load, packet size, protocol distribution and information distribution of packet header) prior to employing entropy statistics to measure the reliability of the parameters. Alqahtani and Gamble [65] proposed a solution to detect DDoS attacks at the service level, tenant

level, application level, and cloud level. The anomaly detection is performed locally at the service level by using a hash map to summarize the data stream. During the monitoring process of the service-level detection, an alarm is raised at the cloud interface, when the ingress flow increases significantly.

The requesters contributing to the high flow rate are identified and tagged. Their flow rate is measured by using an abstract information distance metric and thereafter compared with a pre-determined threshold. If the measured rates are higher than the pre-defined threshold, then the activity is classified as an attack. In the tenant level detection, detectors identify any potential attack by combining hash maps received from the local services. The application level detection correlates the DDoS attacks, the flow rate and the performance degradation of web services to detect the spread of DDoS attacks. The results from the tenant and application levels are sent to the cloud level for further verification. Key performance metrics were used to monitor the detection rate and the extent of the damages.

Kwon et al. [69] proposed a lightweight IDS based on the self-similarity feature. It was determined that the behavioural patterns of normal traffic are similar; and they can be differentiated from the behavioural patterns of malicious traffic (i.e., the outlier). The cosine similarity was used in the approach, and the optimal time interval was used to estimate self-similarity. During the evaluation, the authors used a pre-processor to extract the events from the windows security event log. If the self-similarity is not valid, a system alert is generated and the deployed IDS examines the outlier points, while the IP address of the source before an incident is reported to the system administrator. A key feature of this approach is that it does not require a lengthy learning process and the self-similarity can be determined in real time. Chen et al. [116] proposed a network monitoring and a threat detection system to secure the critical infrastructures in cloud computing. This system is made up of three components: monitoring agents, cloud infrastructure and the operation centre; and it uses Hadoop MapReduce and Spark to enhance the speed of the data processing.

From the review, it can be observed that the DDoS anomaly detection, based on data mining is increasingly popular and the main benefits and limitations are as follows:

Advantages:

- This approach is able to address the limitations of other (non-data mining) approaches in dealing with large databases by extracting information sets and transforming them into an understandable structure.
- It adds a level of focus that helps to improve the process of detecting DDoS anomalies.
- It enhances the network administrator's ability to differentiate between attacking and normal traffic by identifying the bounds for a valid network activity.

Disadvantages:

- Cases of missing or bad values from the dataset would affect the detection efficiency.
- Attribute selection can be an issue for large datasets as the selection of all attributes may worsen the performance.

Artificial Intelligence

The artificial intelligence-based approach (also referred to as soft computing approach), such as Genetic Algorithm, Artificial Neural Network, and Fuzzy Sets, requires a continuous learning process to effectively detect any new anomalies. Joshi and Joshi [89], for example, proposed a Cloud TraceBack model (CTB) and a cloud protector to deal with DDoS attack on web services by using a back propagation neural network. The CTB applies the service-oriented architecture (SOA) approach to the traceback methodology, to determine the true source of the attack. CTB is deployed at the edge router closer to the cloud; and it uses a deterministic packet marking algorithm to mark the reserved flag and ID fields of the IP header packets. However, CTB does not directly eliminate DDoS attacks since the elimination is undertaken by the cloud protector. The implementation phase consists of five stages, namely: dataset training/testing, processing the dataset, determining the NN architecture, training the system, and testing the system. An entropy approach has been proposed by Jeyanthi et al. [84], similar to [85], however, the former uses enhanced entropy to detect the cause of overload by determining the network condition. The simulation results suggest a reduction in traffic and a better response time.

Huang et al. [67] proposed a system that uses Multi-stage detection and a text-based turning test to mitigate any HTTP request flooding attacks. The proposed system is made up of five modules, namely: source checking, counting, attack detection, Turing test, and question generation. The source checking and counting module intercepts any incoming packets, thereafter, the checked packet will be challenged by the Turing test module, if the packet is deemed suspicious by answering several text-based questions. The DDoS attack detection module retrieves and records the traffic behaviour of each virtual cluster (VC), before subsequently being used as a profile to analyse every VC's instant traffic – to check for possible abnormal traffic behaviours by malicious packets.

The text-based Turing testing module receives redirected blocked packets and randomly selects a question to be answered by the requester. Access to the destination is only granted if the question is correctly answered. Question-generation modules periodically update the pool of questions. The system was implemented in Linux kernel and user spaces, according to the requirements with a performance test suggesting a low reflection ratio and high efficiency.

The advantages and disadvantages associated with the deployment of artificial intelligence are as follows:

Advantages:

- The use of neural networks for unsupervised learning can be relatively effective in detecting a DDoS attack packet.
- The adaptive nature of artificial intelligence techniques allows the training and testing of instances in an incremental fashion.

Disadvantages:

- Such approaches may not be easily scalable.
- Over-fitting can occur during the training phase.
- Inadequate access to the required amount of normal traffic data compounds the training of the underlying algorithm, thereby, impacting on the algorithm's effectiveness and efficiency.

Classifier

Classifiers are techniques that learn from a set of labelled data instances, in order to classify a test instance into one of the classes. Such techniques generally operate in two phases, namely, the training phase and the testing phase. The training phase classifier learns by using available training data labels, while the testing phase classifies a test instance as either normal or anomalous by using the classifier [92]. Chonka and Abawajy [78] and Chonka et al. [85] propose a decision tree classification technique: Pre-Decision Advanced Decision and Learning System (ENDER) – to detect and mitigate any HX-DoS attacks against the cloud web services. HX-DoS is an application layer attack that combines both HTTP and XML messages to flood the resources of the target cloud provider service. ENDER detects and marks attack traffic by using two decision theory methods.

In the first method, a rule set (CLASSIE) that has been built over time by using the decision tree to look for both known and unknown attributes. The second method introduces Added Decision Making and Update (ADMU) that decides on the likelihood of a previously classified message. When an attack is detected, a '1' bit mark from CLASSIE will be appended to the message to allow Reconstruction and Drop (RAD) to discard the message before it harms the victim. The RAD is located one-hop away from the victim. The authors used a Gaussian Distribution Traffic Model (GM-TM) to demonstrate that ENDER has a 99% detection rate, compared with the GM-TM's 88% detection with 15 % false positives.

Lonea et al. [68] suggest an IDS-based technique. The IDS was deployed in the cloud's VMs, and a data fusion methodology is hosted at the front-end server. During detection, the alert generated by the VMs will be stored on a MySQL database located in the cloud-fusion unit of the front-end server. Analysis of the alert generated by each of the VM-based IDS uses a quantitative solution classifier: the Dempster-Shafer theory (DST) in 3-valued logic and fault-tree analysis (FTA) for the mentioned flooding attack.

Evaluations suggested that the proposed solution can reduce the false negative rate and increase the detection rate, without any associated complexity.

The multilevel thrust filtration (MTF) mechanism of Iyengar et al. [54] contains four detection and prevention modules designed to shield attackers from gaining access to

the cloud environment. These modules are: traffic analysis, abnormality detection, abnormality classification, and attack prevention. The MTF functions by authenticating incoming packets and detecting four types of traffic congestion (i.e., spoofed attack, DDoS attack, flash crowd, and aggressive legitimate traffic) that affect cloud availability at different levels. The proposed technique uses both host-based and router-based techniques to detect attacks in the early stages, to avoid any influx of malicious traffic to the data centre. In the MTF architecture, an intermediate web server that resides in the cloud acts as a look-up server, which offers a unique ID to cloud users on request. The authorized scrutinized node's IP address would then be sent to registered users.

The escape-on-sight mechanism proposed by Jeyanthi and Iyengar [75] is designed to provide an efficient scalable mechanism for escape during DDoS attacks. Incoming traffic to the cloud is verified by the traffic analyser, which determines whether the traffic is normal or an attack. Once an attack is detected, the firewall filters and blocks the attack traffic to prevent the cloud service from being overwhelmed.

Application layer DoS attacks exploiting REST (Representational State Transfer) API in cloud services have been reported in recent times. REST, an abstraction for distributed communication over a network, can be used as an alternative to SOAP. The vulnerability is due to its exposure since services using REST do not require authentication. This allows (cloud) services to be overloaded, resulting in resource depletion during an attack. To defend against this attack, Michelin, Zorzo, and De Rose [73] proposed a defence strategy that uses an authentication token.

This approach has two different modes, namely: monitoring and filtering (see Figure 2.10). During the monitoring mode, a stress test is carried out on the system to check for overloading. It verifies the token from each user; and if an invalid token is detected, the user would be placed on a grey list as a potential attacker. During overloading of the service, the mechanism moves from monitoring to filtering mode; and the grey list is changed to black. Users on this blacklist would have their REST dropped, thereafter, the system would revert to monitoring mode, whenever the system is decongested. The mechanism was evaluated on OpenStack Grizzly running on VMware workstation 10.0.1.

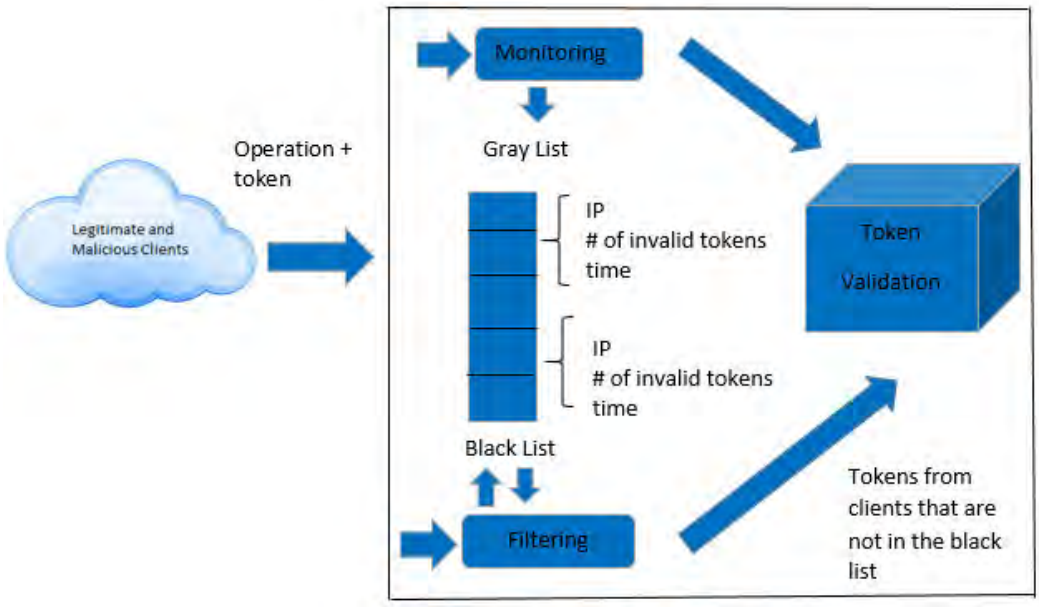

Figure 2.10 Client control solution architecture (adapted from Michelin et al. [73]).

In summary, the classifier anomaly detection presents some advantages and a few drawbacks.

Advantages:

- Classifier-based anomaly detection techniques have a high detection rate subject to precise threshold settings.
- They are characterized by a high adaptation rate for updating detection strategies.

Disadvantages:

- They require adequate information training to detect any unknown attack event.
- Resource consumption is high, compared with other existing techniques.

Machine learning

Deploying machine learning, in order to detect cloud DDoS attacks encompasses techniques, such as statistics and data mining; but these techniques have a subtle difference from statistical techniques. The latter requires understanding the process

that generates the data while machine learning involves building a system to improve the performance, based on past results. Gupta et al. [77] propose a profile-based network intrusion detection and prevention system that secures the cloud against malicious insiders and outsiders. It combines both fine-grained data analysis and the Bayesian technique approach to detect DDoS attacks using an unsupervised learning algorithm. The latter aims to detect network-based attacks, such as TCP SYN flooding. As in the case of [56], the network profile is deployed on each VM in the cloud environment. The first stage of the system entails passing a newly created VM through a rigorous attack test, using the attack signature database. Thereafter, a profile is created for the VM that identifies any possible attacks against other VMs. The second stage, the anomaly detection stage, uses a normal traffic pattern, obtained from the virtual bridge of the VM to check for consistency against other probable attack behavioural patterns. The final stage is the network intrusion detection, which analyses the normal traffic flow path obtained from the virtual bridge for the attack pattern from VM profile database. Any detected anomaly would be reported to the alert module, before going for further analysis.

A generic approach to detecting network anomaly through independent component analysis was proposed by Palmieri et al. [100]. The distributed approach employs a two-phase machine learning scheme, which consists of Blind Source Separation (BSS) and a rule-based classifier to detect zero-day attacks that alter the flow characteristics and the traffic volume rate. The BSS extracts the traffic features from the distributed sensors to be used by the decision tree classifier to build a baseline traffic profile.

The advantages of the machine learning technique include:

- Relatively high efficiency in detecting a DDoS attack pattern.
- The ability to change their execution strategy during detection, having acquired additional information.

The disadvantages of the machine learning technique include:

- Such an approach requires significant computing resources during the training and testing phase.
- Bottlenecks can be introduced into the system, as a result of high overheads, which lead to a performance degradation of the monitored system.

2.3.2.3 Hybrid Detection

The hybrid-based detection approach involves the use of both signature-based and anomaly-based techniques. This approach uses the complementary features of both techniques to achieve a higher detection rate. For example, Krishnan and Chatterjee [76] proposed an adaptive distributed IDS that combines anomaly-based and knowledge-based techniques to defend against cloud DDoS attacks. The solution has a service agent, an alert agent and a storage agent, which communicate with each other and the pair nodes. This adaptive hybrid solution is designed to improve the detection rate by lowering the false positives. The system also implements an alert clustering and analyser that helps all co-operating nodes to differentiate between false alarms and malicious nodes. Cha et al. [39] designed a three-stage anomaly detection method. The first stage is the monitoring stage, which uses a rule-based system to pre-process the known DDoS attack patterns. The second stage presents a lightweight anomaly detection that predicts the expected future load on each customer interface by using time-series modelling.

The traffic volume over the network is divided into large and small volumes along the time-axis; and the Bayesian technique is used to analyse the DDoS attack candidate on the network topology. The last stage uses a focused anomaly to detect both known and unknown DDoS attack patterns, using an unsupervised learning algorithm.

Mordi et al. [79] designed a hybrid network-based intrusion to detect cloud DDoS attacks. Snort, an open-source signature-based detection method that stores the rules of known DDoS attack patterns, and a Bayesian classifier (a statistical classifier that predicts the probability of a network event belonging to a class, such as normal or malicious, with high accuracy) were used in this solution. Eucalyptus, an open-source cloud was used for the experimental set-up, where an intrusion detection system was installed on each node controller and all ports were opened for the testing purpose. Scapy, a powerful interactive packet manipulation program that can forge or decode packets of a wide number of protocols, was employed to generate custom packets; while performance and quality were evaluated by using the KDD ’99 dataset.

Teng et al. [80] proposed a cooperative intrusion detection architecture, modelled with E-CARGO to defend against cloud DDoS attacks. The proposed architecture is made up of four layers. The first layer, an event generator, collects the network packets and

generates suspicious intrusion events. The feature detector works in a similar fashion as Snort, and is used to separate events, according to the network protocols (e.g., ICMP and TCP). The statistical detector uses the data packets from the feature detector to determine the attack event. If the number of packets obtained within a certain time range is higher than the threshold set, it is considered to be an attack.

Finally, the fusion centre uses agents to perform different roles, such as data pre-processing, space-time fusion and content fusion. The findings from the experiment showed that the proposed method is a viable solution to detect DDoS and slow scanning attacks. However, it has the limitation of differentiating flash crowd traffic from malicious attacks.

A hybrid hierarchical correlated approach, proposed in [108], uses security probes to collect and analyse information at different cloud architectural levels. The correlations of intrusion symptoms to the identified cause and target are driven by a knowledge-based method, and represented by an ontology.

While the hybrid approach provides the advantages offered by both signature and anomaly-based approaches, they are associated with overheads and complexity in getting different algorithms to interoperate efficiently and effectively. A comparative summary of the reviewed detection approaches is presented in Table 2.2.

Table 2.2 Comparative summary of DDoS defence approaches

Approach	Efficiency	Overhead	Adaptive	Overfitting	Scalability issues
Statistical	✓				
Data Mining	✓				
Artificial Intelligence	✓		✓	✓	✓
Classifier	✓	✓	✓		
Machine Learning	✓	✓	✓		
Signature-based		✓			✓
Hybrid	✓	✓	✓		

2.3.3 Traceback and IP Spoofing Detection

The Traceback technique can help to locate the true source of DDoS attacks, as most of these attacks tend to spoof their addresses (e.g., launching a reflector attack). In proposing a defence strategy for application layer DDoS attacks against cloud services, Yang et al. [15] proposed a SOA-based technique called the SOA-Based Traceback Approach (SBTA) and a cloud filter. SBTA performs DDoS attack

traceback by deploying the technique before the web server. The SBTA uses advance packet marking based on the Compressed Edge Fragment Sampling (CEFS) to determine the path reconstruction. The cloud filter, on the other hand, is deployed on the edge router; and it is used as a control mechanism for filtering and rate-limiting purposes. The cloud filter gathers cloud traceback mark tags and sources the IP addresses during the attack; and it uses the database to filter out packets with spoofed IP addresses. A significant drawback of this technique is its reactive approach and the high rate of false negatives.

The defence against spoofed IP addresses in cloud DDoS attacks that hide the true source has also been proposed. In [85], a technique is proposed to identify any spoofed IP addresses in DDoS attacks. The authors also proposed an algorithm, which is only activated, whenever there is a sudden rise in the packet traffic greater than a pre-defined threshold. The approach also consists of a cloud authentication system (CAS) that verifies the legitimacy of the connecting cloud user. The CAS, hosted in the cloud environment, has two tables and three procedures. The tables are a spoofed address table and the current connection table. The procedure, on the other hand, comprises a check flood, a packet check, and the final check. These are collectively used to determine the occurrence of a flooding traffic for subsequent filtering.

The OPNET Modeller 14.0 was used to simulate the approach; and the performance was improved when a buffer was employed for the overloaded packet. A similar approach, using IP spoofing to defend against DDoS attacks in the cloud environment was proposed in [90]. Motivated by the fact that most DDoS attacks are characterized by the spoofing of IP addresses, Osanaiye proposed an operating system (OS) fingerprinting technique that monitors incoming packets to the cloud environment, to determine its source OS. The algorithm has two stages, namely: active and passive stages. During the passive stage, the packet headers of incoming traffic are captured and analysed to determine the running OS.

In the active stage, specially crafted packets are sent to the source IP address of the connecting packet. A matching is performed to compare the passive OS and probed OS. If both OSs are not the same, the packets would be considered spoofed and dropped. Due to OS distribution, an extension of [90] was presented in [101], where the final TTL value is used to determine any spoofed DDoS attacks, when the spoofed and the true source run similar OSs.

2.3.4 Other Forms of DDoS Attack Defences

In solving DDoS attack issues in cloud computing, Yu et al. [81] considered the scenario where an individual cloud user is being targeted. In their approach, an intrusion prevention system (IPS) was deployed at different access points of the cloud environment to monitor any incoming packets during DDoS attacks. This is a reactive method that dynamically allocates the available resources during DDoS attacks, in order to compensate for the attacks. A queuing model is developed to establish a relationship between the resource allocations and various attack strengths. The proposed DDoS mitigation algorithm extracts non-attack parameters for a protected server. It identifies the resources for a current IPS and the available resources from the cloud. The IPS is cloned from the original IPS when an attack is detected. All IPSs work in co-ordination to filter out attack packets and to provide quality of service (QoS) for the users. When the volume of attack decreases, the system would automatically reduce the number of IPS and de-provision the resources previously allocated back to the pool.

Guenane et al. [82] presented a firewall-based approach that reduces the effect of cloud DDoS attacks, based on a Security-as-a-Service model (SecaaS). The hybrid feature in the proposed solution is derived from its architecture, which consists of two parts, namely: the virtual and the physical. The virtual part is made up of virtual firewalls deployed as VMs that execute firewall functionalities, such as monitoring, analysing and reporting with dynamic resource provisioning. The physical part is the organization's physical IT resource infrastructure that agrees to acquire a security service offered by the cloud provider.

The DDoS mitigation system redirects and load balances traffic on overloaded physical firewalls, based on its hybrid architecture operation. The localized firewall in the cloud receives the redirected traffic, which is managed by the virtual part of the hybrid architecture to achieve the two key objectives of decision management and availability performance.

Iyengar and Ganapathy [83] proposed a trilateral trust-based defence mechanism, which identifies different attack groups prior to separating the legitimate packets from the incoming traffic. The TTM authenticates the incoming cloud user requester as

either a trusted client or a threat, using three sequential traffic threat notification levels. Each level detects different threats, thereby, reducing the threat traffic for the successive levels. Early stage detection reduces the traffic congestion at the data centre, which improves its availability for legitimate cloud users. The TTM protocols include: client ID acquisition, mutual trust establishment, historical behaviour monitoring, credit points updating, service provision, and attack exclusion.

Chopade et al. [86] proposed an average distance estimation technique to defend against cloud DDoS attacks. An exponential smoothing estimation was used to determine the mean value of the distance in the next time period. The Minimum Mean Square Error (MMSE), a linear predictor, was used to determine the distance-based traffic separation for DDoS detection by estimating the traffic rates from different distances. The distance value was determined from the TTL field of an IP packet. The technique was implemented by using NS2 on more than 100 nodes; and the evaluation suggested that the technique has a high detection rate with low false positives.

Liu [87] described a new form of DoS attack, coined Loaded Spread Pair estimator (LSP), which exploits the under-provisioning of network resources in the cloud infrastructure. The author proposes a solution that does not detect nor prevents the attack, but dynamically transfers the cloud servers to a different infrastructure, in order to achieve the desired QoS. The technique does not only cater for DoS attacks, but also compensates for performance degradation due to resource constraints.

Aishwarya and Malliga [88] used IDS to defend against transport layer DDoS attacks, using SYN cookies. In this approach, connecting users with malformed ACKs can be ignored; and the packets are checked to determine whether they are spoofed, using hop count filtering (HCF). This serves as a first layer of security. In the second layer of security, the sequence number of the SYN packet is encoded, therefore, only legitimate cloud users can decode it. An open-source distributed architectural framework that provides an application program interface and the tools to develop multiple probes, was proposed in [107]. The framework can be dynamically deployed to collect security information at different cloud architectural levels for analysis of intrusion in the cloud system.

A shuffling-based moving target approach for DDoS cloud defence was proposed by Jia et al. [111]. In their work, the system architecture uses a selective server replication

and intelligent client reassignment to turn victim servers into a moving target to isolate any DDoS attacks.

2.4 Discussion

In the preceding section, DDoS attacks were categorized into application-bug level and infrastructural level attacks. Most research efforts appear to be directed towards the infrastructural level DDoS attacks, which include both network and application flooding attacks. The reason for several reported cases is because of the ease with which infrastructural level DDoS attacks are carried out. In infrastructural level DDoS attacks, the attacker does not seek to exploit cloud vulnerability; as malicious flood packets are merely directed towards the target to clog and consume its resources – to the detriment of legitimate users. Application-bug level DDoS attacks, on the other hand, have also been reported [57] [78].

In these attacks, system vulnerabilities are exploited to carry out the attack. Typically, the exploited vulnerabilities are: misconfiguration, outdated patches, protocol vulnerability and system weakness. However, very few publications on application-bug level DDoS attack mitigation have been published.

The most common deployment location of DDoS defence in the cloud environment is the access point (similar to IDS deployment – see Table 2.4); and it is observed that some have proposed a distributed deployment for efficiency. The choice of the access point is, probably, due to the distributed nature of the Internet, and also the cloud architecture. Source-end deployment would have been the most ideal location to detect DDoS attacks but it would be challenging to enforce all the hosts on the Internet when adopting a general policy.

It also appears that earlier techniques generally used signature-based classification to identify any known DDoS signatures. While this is an effective solution against known DDoS attack patterns, such a solution is increasingly irrelevant in today's threatened landscape – due to their inability to detect unknown DDoS attack signatures. For example, there is an abundance of tools that can be used to generate DDoS attacks to defeat signature-based approaches, thereby, leading to a high false negative rate. Anomaly-based solutions are increasingly popular since such approaches have been shown to be effective against both unknown – and derivatives – of known attack patterns.

Normal traffic behaviour is modelled to obtain a normal behavioural profile by using modelling techniques, such as data mining, machine learning, artificial intelligence and statistical methods. This involves extracting packet attributes during the non-attack period and profiling the normal behaviour. During the attack period, incoming packet events are analysed – together with the profiled normal behaviour – to detect any DDoS attacks against the cloud services. Hybrid solutions attempt to take advantage of the complementary nature of these approaches, by integrating both signature-based and anomaly-based techniques to achieve a better detection rate. Figure 2.11 depicts the research trend between January 2010 and December 2015.

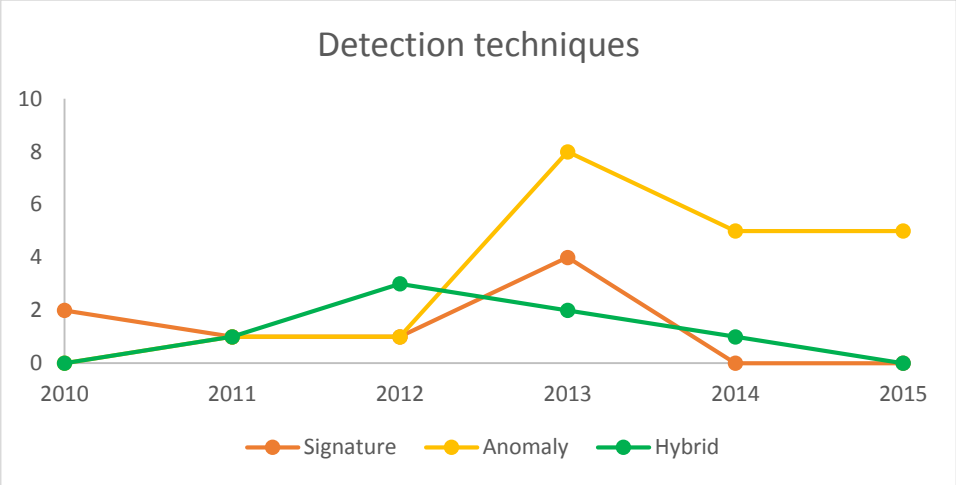

Figure 2.11 DDoS detection techniques for cloud computing trends between January 2010 and December 2015.

In DDoS defence, most techniques used for generating normal profile patterns fail to consider low-rate DDoS attacks, which can result in a high false negative rate. An example is the work of [80] [85], where a pre-defined threshold is used for determining the presence of a DDoS attack. In such settings, low-rate DDoS attacks would not be detected.

During the evaluation phase, it is apparent that evaluating the proposed solutions remains a challenge – due to the lack of up-to-date real-world datasets for training. Commonly used datasets (see Table 2.3) in the literature include: the UNB ISCX 2012 dataset, the CAIDA DDoS 2007 dataset, the DARPA 2000 LL-DDoS from Lincoln laboratory, MIT, and the KDD'99 dataset.

As an example, the CAIDA DDoS dataset (one of the most current datasets) is made up of an hour of anonymized trace from DDoS attack on August 4, 2007, between the

hours of 20:50:80 UTC to 21:56:16 UTC split into 5 minutes pcap files. Another key issue is the dearth in the availability of labelled datasets. This is evident as KDD'99 represents one of the few publicly available labelled datasets currently in use today by researchers.

Table 2.3 Summary of commonly used metrics and datasets

Reference	Performance evaluation metrics	Benchmark/Datasets
Lo et al. [59]	Detection rate and computation time	Snort[1]
Shamsolmoali et al. [62]	Detection rate and false alarm rate	CAIDA "DDoS Attack 2007"[2]
Choi et al. [41]	Detection rate and average detection time	Simulated
Huang et al. [67]	Throughput during attack and performance overhead.	Simulated
Lonea et al. [68]	Detection rate and computational time	DARPA 1999 Dataset[3]
Wang et al. [24]	Detection rate and computational cost	UNB ISCX dataset[4]
Michelin et al. [73]	Response time and CPU usage	Simulated
Chonka et al, [78]	Detection rate	StuPot project dataset[5]
Modi et al. [79]	Detection rate, computational cost, scalability	KDD'99 dataset[6]

[1]https://www.snort.org, [2]http://www.caida.org, [3]http://www.ll.mit.edu/ideval/data/, [4]http://www.unb.ca/research/iscx/dataset/, [5]http://www.deakin.edu.au/~ashley/, [6]http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Identification of normal and other attack patterns is crucial as this is the key metric to determine the utility and efficiency of the proposed techniques. Other commonly used metrics that have been considered in the literature include the detection rate, the average response time, the percentage of bandwidth utilization and the normal packet-survival ratio.

- Detection rate: The detection rate measures the accuracy of the defence technique, in terms of identifying attack patterns in a traffic flow.
- Average response time: This is the average time it takes a legitimate cloud user to request and receive cloud services during an attack. The service can be in the form of an application hosted in the cloud or an infrastructural service.
- Percentage bandwidth utilization: This represents the number of bits transmitted or received by the cloud user per unit time in bits per seconds.

During a DDoS attack, this may include both legitimate and attacking traffic, however, only legitimate traffic will be considered here.

- Normal packet-survival ratio: This is determined by measuring the ratio of the total number of legitimate packets that successfully access the cloud environment against the total number of attempted accesses.

A summary of the reviewed DDoS attack defence categories in cloud computing is presented in Table 2.4.

Table 2.4 Summary of some existing DDoS attack defence mechanisms in cloud computing

Year	Reference	Detection technique			Deployment location					DDoS attack type		
		Signature	Anomaly	Hybrid	S[1]	AP[2]	I[3]	D[4]	NS[5]	App[6]	Infra[7]	NS[5]
2010	Bakshi et al. [55]	✓				✓					✓	
	Lo et al. [59]	✓				✓					✓	
2011	Gul et al. [58]	✓				✓						✓
	Kwon et al. [69]		✓			✓						✓
	Cha et al. [35]			✓		✓					✓	
2012	Karnwal et al. [42]	✓						✓		✓		
	Bedi et al. [74]		✓			✓						✓
	Krishnan et al. [76]			✓		✓						✓
	Chonka et al. [78]			✓			✓			✓		
	Modi et al. [79]			✓		✓						✓
2013	Lonea et al. [68]		✓			✓					✓	
	Karnwal et al. [57]	✓						✓		✓		
	Gupta et al. [60]	✓				✓					✓	
	Modi et al. [61]			✓		✓						✓
	Zakarya et al. [63]		✓				✓					✓
	Huang et al. [67]		✓			✓					✓	
	Lonea et al. [56]	✓				✓					✓	
	Choi et al. [29]		✓						✓		✓	
	Ismail et al. [70]		✓			✓						✓
	Dou et al. [71]		✓			✓						✓
	Negi et al. [72]		✓			✓						✓
	Jeyanthi et al. [75]		✓					✓				✓
	Gupta et al. [77]			✓		✓					✓	
2014	Shamsolmoali et al. [62]		✓			✓					✓	
	Vissers et al [64]		✓			✓				✓	✓	
	Choi et al. [41]		✓					✓			✓	
	Iyengar et al. [54]		✓					✓				✓
	Michelin et al. [73]		✓			✓				✓		
	Teng et al. [80]			✓		✓						✓
2015	Alqahtani et al. [65]		✓			✓					✓	
	Girma et al. [66]		✓					✓				✓
	Wang et al. [24]		✓			✓						
	Marnerides et al. [112]		✓			✓						
	Chen et al. [116]		✓			✓				✓		

[1]Source-end, [2]Access point, [3]Intermediate, [4]Distributed, [5]Not Stated, [6]Application-bug level [7]Infrastructural level.

2.5 Summary

This chapter has reviewed the academic literature on DDoS attacks against cloud services, and the mitigation strategies published between January 2010 and December 2015. Also presented in this chapter is a taxonomy of the different types of cloud DDoS attacks and a corresponding DDoS defence taxonomy. Anomaly-based detection and access point deployments were identified as the most popular defence techniques and the deployment location proposed in the literature.

Despite the amount of research efforts in this area, there are a number of challenges that need to be addressed. For example, the need for a defence solution to detect both application-bug level and infrastructural level DDoS attacks, as contemporary DDoS attack tools are capable of launching such attacks targeting different cloud components. More research should also be directed towards efficient defence solutions that can detect both high-level and low-level DDoS attacks. There is also a need for an effective approach for selecting optimal thresholds in determining DDoS attack patterns for existing and future defence techniques; otherwise, this could lead to a high rate of false positives and false negatives.

This thesis, therefore, provides a solution to some of the challenges mentioned above. In mitigating DDoS attacks in cloud computing, this thesis presents an IP spoofing detection technique that detects spoofed source IP addresses. In addition, to solving the issue of the increase in Internet traffic that complicates efforts to detect DDoS anomaly pattern, an EMFFS method that selects the important features for detecting DDoS attacks to achieve optimum classification has been proposed. Finally, to address the detection issue for unknown and a variety of known attacks, a change-point cloud DDoS detection technique that detects different forms of DDoS attack, which deviates from the normal pattern, using a packet inter-arrival time feature, is presented.
In the next chapter, this thesis presents an IP spoofing detection technique that uses host-based OS fingerprints to determine a spoofed source during DDoS detection.

Chapter 3: TCP/IP Header Classification for Detecting Spoofed DDoS Attack in Cloud Environment*

3.1 Introduction

In the previous chapter, the literature review chapter, a survey of common DDoS attacks targeting cloud computing and its defences was presented. From the review, the spoofing of source IP address has been identified as a common attribute of current DDoS attacks to disguise their identity, in order to frustrate easy traceback. Therefore, this chapter discusses different methods for detecting spoofed IP packets in cloud computing, and propose a host-based operating system (OS) fingerprinting that uses both passive and active methods to match the OS of incoming packets from its database, in order to detect spoofed sources during DDoS attack.

Various techniques have been proposed by researchers to defend against IP spoofing DDoS attack. This can be broadly categorized into host-based, router-based, and a combination of the two that renders a hybrid solution [97]. Host-based can be classified as being either passive or active while the deployment of router-based solutions can be basic or distributed [97].

A router-based filtering mechanism called Virtual Anti-Spoofing Edge (VASE) was proposed in [98]. VASE is used to filter spoofed IPs with agility by configuring filtering rules to reduce unnecessary overhead costs on the router in the absence of any spoofed attack. It uses sampling and on-demand configurations. VASE is an open deployment that is compatible with commercial routers for deployment in a real network. This mechanism is not a state-of-the-art solution, as the outer-sync, sampling cost, and filter generation introduces complexity into the system.

Shiaeles et al. [128] proposed a Fuzzy hybrid-spoofing detector (FHSD), a multi-layered IP spoofing detection technique, which is based on five features namely: a source-MAC address, the hop count, the OS passive fingerprint, the GeoIP and the web browser user agent. It functions by using fuzzy empirical rules and the fuzzy largest of maximum operator for offensive-traffic identification and mitigation.

*Material presented in this chapter has appeared in [90][101].

This technique comes with its own limitations, like the inadequate update of feature values stored in files, when compared with the database.

Yu et al. [118], in their recent work, used a statistical approach by employing a semi-Markov model to establish the browsing behaviour. From the outcome of the model, they showed that by using second-order statistical metrics, one can differentiate a mimicking attack from a flash crowd. This technique failed to convince that it can detect a sufficiently large number of mimicking attacks.

Hop-Count Filtering (HCF), which uses the Time-to-Live (TTL) value of the source packet header to detect spoofed DDoS attacks was proposed in [119]. TTL is an 8-bit field on the IP packet header, which decreases by one on every hop taken towards the destination [119]. Its major function is to specify the maximum lifetime of each transmitted packet on the Internet, however, due to its decrementing feature, it is being used to detect spoofed IP addresses. This is achieved by inferring the initial TTL value from the observed final value, in order to get the hop-count of a packet from its source to its destination.

According to RFC 1340, the recommended initial TTL value was fixed at 64, but this value is often ignored by OS developers. According to [120] that conducted research into the initial TTL values for different modern operating systems, six values were obtained: 30, 32, 60, 64,128, and 225. This set of values covers common OSs, such as Linux, Windows, UNIX, and BSD.

From the observed final value, the initial TTL value can be determined; and the difference between the final and the initial value gives the Hop Count. This hop count value will be compared with the value obtained from the constructed IP2HC mapping table. Packets will be accepted if the values match, while a mismatch would indicate that the IP address has been spoofed. The main drawback of this solution is that major OS developers do not have common initial TTL values. Therefore, a mismatch can easily occur when determining the initial TTL value from the observed TTL.

3

The BGP Anti-Spoofing Extension (BASE) is another spoofing mechanism proposed in [121]. BASE relies on BGP updates and it functions by sending update routes used in learning the direction of an incoming packet. It marks packets with a unique key, and uses the key as the incoming direction. BASE sends piggyback updates, which are made up of marking information and control information on top of BGP updates – as opposed to sending their own. The demerit of this technique is that BASE relies on BGP updates, which means it must travel the same path as a BGP update, which does

not travel in the same path as normal traffic. This leads to incorrect marking and misidentification of the legitimate packets as spoofed packets.

This chapter, hereby, proposes a host-based operating system (OS) fingerprint that uses both passive and active methods to match the OS of incoming packets from its database, in order to detect spoofed sources during a DDoS attack. This is achieved by using TCP/IP header classification of the incoming traffic of connecting hosts in the front-end of the cloud computing environment. This research has collected and analysed TCP SYN and SYN+ACK packets during connection establishment in both the passive and the active stages. To achieve this, TCP/IP header features, like Time to live (TTL) value, window size, IP Don't Fragment (DF) option, and the total length were used to classify the OS flavour. Furthermore, the final TTL value was used to cater for the possibility of false negatives during detection, where spoofed and true source runs similar OSs.

The rest of the chapter is structured as follows. Section 3.2 discusses the resource consumption in the cloud, while Section 3.3 lists the assumptions of this chapter. Section 3.4 describes the TCP/IP header features while Section 3.5 highlights the Methodology/implementation. The details of the findings and the evaluation were described in Sections 3.6 and 3.7 while Section 3.8 summarizes the chapter.

3.2 Resource Consumption in The Cloud

DDoS is a coordinated attack launched on cloud services, resources, and the infrastructure, in order to make it unavailable to legitimate users by flooding it with malicious packets. One of its key attributes is to spoof its IP address [89], in order to disguise and disallow easy traceback. The reasons for this attack can be traced to competition among the cloud providers, protests by certain groups, revenge, extortion, and the testing of proficiency. Notable among the attack tools is the Apache Killer.

A DDoS attack exploits the inherent weaknesses of cloud, which could include system weaknesses, computational system overheads, the misuse of protocol (HTTP, NEST), and infrastructural weaknesses. In launching the attack, DDoS carries out amplification, which can either be a direct or a reflexive attack. The source of the attack in the cloud could either be from external to internal, from internal to internal, or internal to external. External here refers to outside the cloud while internal means inside the cloud computing environment.

Other forms of resource and service consumption in the cloud are flash crowd (large amounts of legitimate users accessing services and resources at the same time), Economic Denial of Service (EDoS) and Fraudulent Resource Consumption (FRC).

SYN flooding

TCP is a transport layer connection-oriented protocol that ensures the reliable delivery of packets by sending an acknowledgment (ACK) message to every successfully delivered packet between source and destination, or negative acknowledgement (NACK), if otherwise. Recent attacks on the Internet and on cloud service providers, like the .cn domain of China, Amazon cloud, eBay, and Yahoo have shown its vulnerability to DDoS attacks. Recent research has shown that 90% of DDoS attacks use TCP [123]. As TCP is connection-oriented, when establishing connection to the front-end of the cloud, a SYN packet is sent by the attacker. Upon receiving it, it returns a SYN+ACK control packet to the sender, as an acknowledgment waiting for an ACK to complete the 3-way handshake. Attackers exploit the connection-oriented feature of TCP by initiating half-open connections – until there is a TCP connection timeout, which ties down the resources and chokes the network to prevent other incoming traffic. During a spoofed attack, the front-end of the cloud would never receive the final ACK to complete the 3-way handshake. This type of DDoS attack is called TCP SYN flooding; and it is always directed to the listening ports of the target [123].

It is the predominant type of DDoS attack, therefore, this chapter only considers TCP packets.

3.3 Assumptions

The scenario presented here is based on some key assumptions made. This work assumes that performance (PfR) routing, which intelligently delivers path control has been implemented in the network. Therefore, it is assumed that the path taken by the packet during passive and active stages is the same. Another assumption made was that no fingerprint concealment tool was used by the remote host to modify or prevent the probe packets sent towards it. This research has used a specially crafted probe packet during the active stage, consequently, this would have a minimal effect on the network congestion.

3.4 TCP/IP Header

TCP/IP header features, as shown in Figure 3.1, are sets of control information used to reliably send packets from the source to the destination. Some field values are constant when traversing networks while others change to describe the state of the packet. Analysis of TCP/IP headers has been used by previous researchers for different purposes. Among these is its use for OS fingerprinting that determines the source of the OS for a remote host. OS fingerprinting is often used by system administrators when managing lots of systems, in order to identify malicious clients, and to locate and patch vulnerable OSs.

In carrying out a forensic analysis on the incoming packets to the cloud environment, the unique combination of TCP/IP header field values that represent different OSs was harnessed by first determining and matching the OS during passive and active fingerprinting. Thereafter, using both SYN and SYN+ACK packets of TCP, and TCP/IP, the packet header analysis was carried out. The interesting fields considered are the TTL, window size, Don't Fragment (DF) bit, and total length.

The TTL field is an 8-bit field that indicates how long a packet can stay on the Internet as it decreases by one for each hop the router passes through to its destination. The recommended initial TTL was fixed at 64, but this value has been ignored by most OS developers. As stated earlier, with most modern OSs, the observed final TTL value can be used to infer its initial value. This is because most Internet devices are within 30 hops of each other [124]. Window size field, on the other hand, indicates the number of octets of data the sending computer is ready to receive without an ACK. The initial window size can vary from OS to OS. The total length determines the length of the entire packet, which is the header and the payload, while the DF bit is used in an IPv4 header to break down packets when traversing links with smaller maximum transfer units (MTU) than the packet size. It determines whether the OS sets the DF bits, or not.

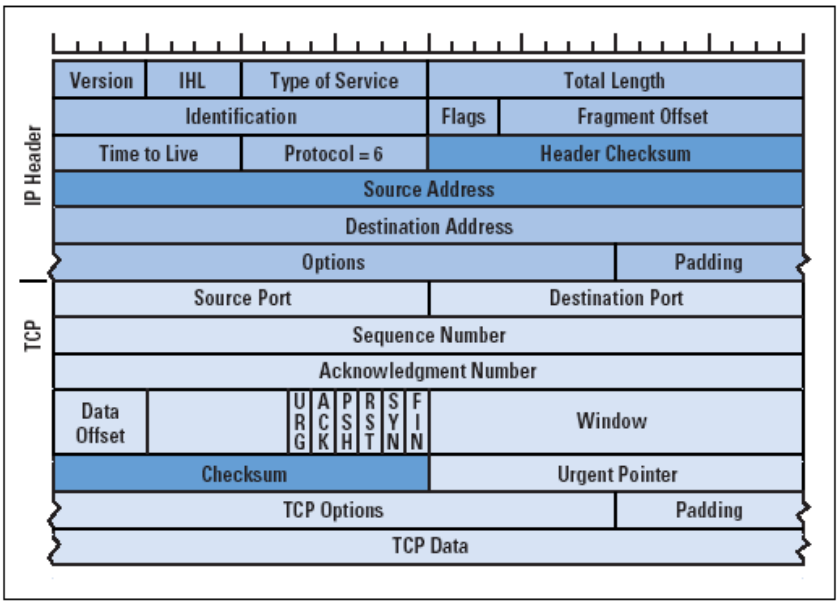

Figure 3.1 TCP/IP header format (adapted from [125]).

3.4.1 Operating System Fingerprinting

OS fingerprinting is used to determine the OS of a remote computer. This can either be determined passively or actively. Passive detection monitors the network traffic into a host, while active detection is performed by sending specially crafted probe packets towards a target to identify its OS. Detection is achieved by analysing the packet header of the response [126], and then matching it with the database of the known OS. It uses the unique features of each TCP/IP header response as a signature to identify any different hosts, according to their OS [126].

3.4.2 Operating System Distribution

The OS usage distribution is a key factor to be considered, when analysing a TCP/IP header. A recent survey carried out by [127] shows that Windows OS forms a large share of the current OS used over the Internet. Among the Windows flavours, windows 7 leads the pack, followed by windows XP. The reason behind this can be attributed to its ease to use over the years. Other Oss, like Mac and Linux are slowly gaining penetration globally among end- users, as shown in Figure 3.2 below. Each OS is implemented with a different kernel, however, each preceding version tends to be an improvement on the previous one.

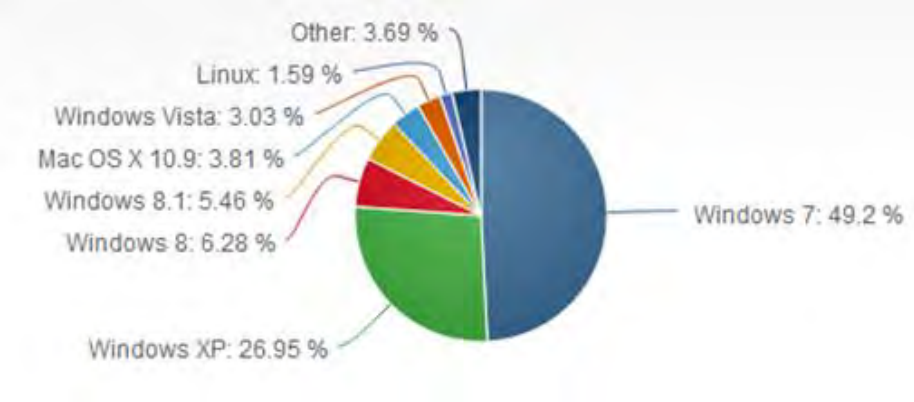

Figure 3.2 OS distribution (adapted from [127]).

3.4.3 Operating System Fingerprinting Method

As stated earlier, OS fingerprinting uses two main approaches to identify the remote OS of a system by a passive and/or active method.

During passive fingerprinting, connecting host packet headers are analysed and compared with the stored OS database to determine its OS. Examples of common passive fingerprinting tools are p0f, Ettercap and passive OS fingerprinting for IP tables (OSF). Active fingerprinting sends a specially crafted probe packet, which could be a combination of TCP, UDP, and ICMP to both open and closed ports of the target host. The header of the response packet is captured, analysed and matched with the known stored OS database to determine its OS. Common active tools include Nmap, SinFP, Xprobe, and Zion. In this chapter, P0f and Nmap were used to determine the OS of the connecting host.

3.5 Methodology and Implementation

In implementing OS fingerprinting in cloud computing to detect spoofed IPs during a SYN flooding DDoS attack, this research employs both active and passive methods of OS fingerprint by using Nmap and P0f (- see Figure 3.3). This was deployed at the front-end of the cloud computing environment to classify any malicious from legitimate packets before granting them access. The spoofed DDoS detector above operates in two different modes: the working mode and the idle mode. The detector would remain idle unless triggered by a large flow of packets big enough to deny cloud users access to the resources. During SYN flooding DDoS attacks, the incoming malicious packets often have their IP addresses spoofed before launching an attack –

in order to avoid easy traceback.

In the passive stage, the packet header features of incoming packets are recorded and analysed by p0f. Matching the analysed header with the p0f's database of known OS would determine the OS of the incoming packet. During the active stage, the probe packet is sent to the source IP of the received packet by using Nmap. If the spoofed IP address is active, Nmap captures its response and identifies its OS from its database. Our technique goes a step further, after classifying the OS of the incoming packets to analyse the TCP/IP header to negate any chances of common OS by both the true source and the spoofed source. There are chances of these happening as the minor kernel version upgrade may contain the same protocol implementation [26]. TTL, window size, total length, and DF were extracted from the packet header and analysed. The observed passive TCP/IP header and the probed active TCP/IP header during the active stage would be compared to find a match. If no match can be found, then the packet would be identified as spoofed and dropped, while a similar match might mean the IP address is its true source. Thereafter, this research proceeds to use the final TTL value to cater for any possible misclassification.

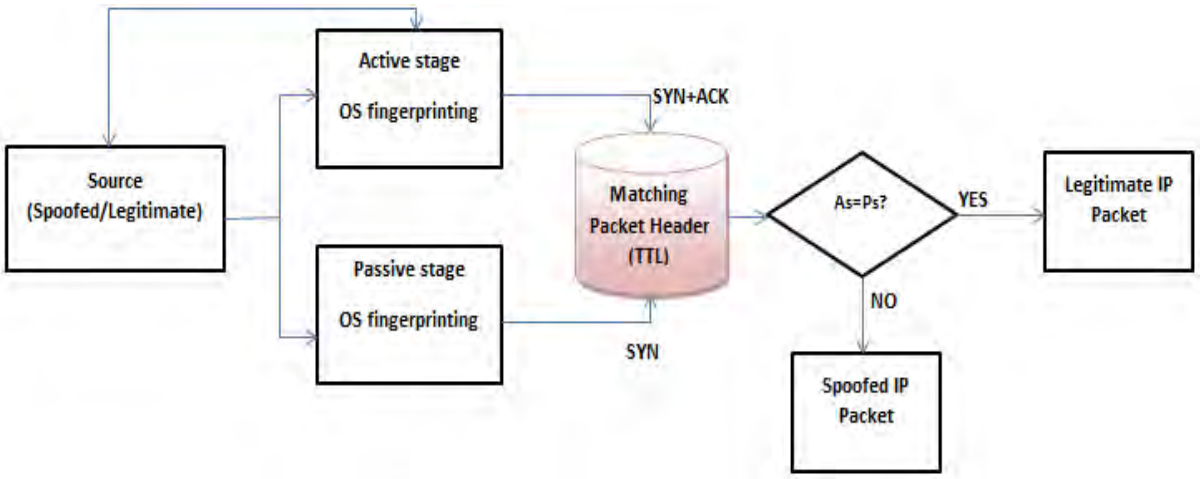

Figure 3.3 IP Spoofing detection method.

Testbed

Xen Cloud Platform

In implementing our proposed solution, this research deployed an open source Xen Cloud Platform (XCP). XCP uses a Xen Hypervisor to provide an abstraction layer between the OS and the cloud hardware [129]. Its basic architecture consists of Xen

hosts that house virtual machines and a shared storage that can be pooled by the XCP hosts [122]. One of the Xen hosts can be configured as a master host that serves as an administrative controller to other XCP hosts [122] – See Figure 3.4

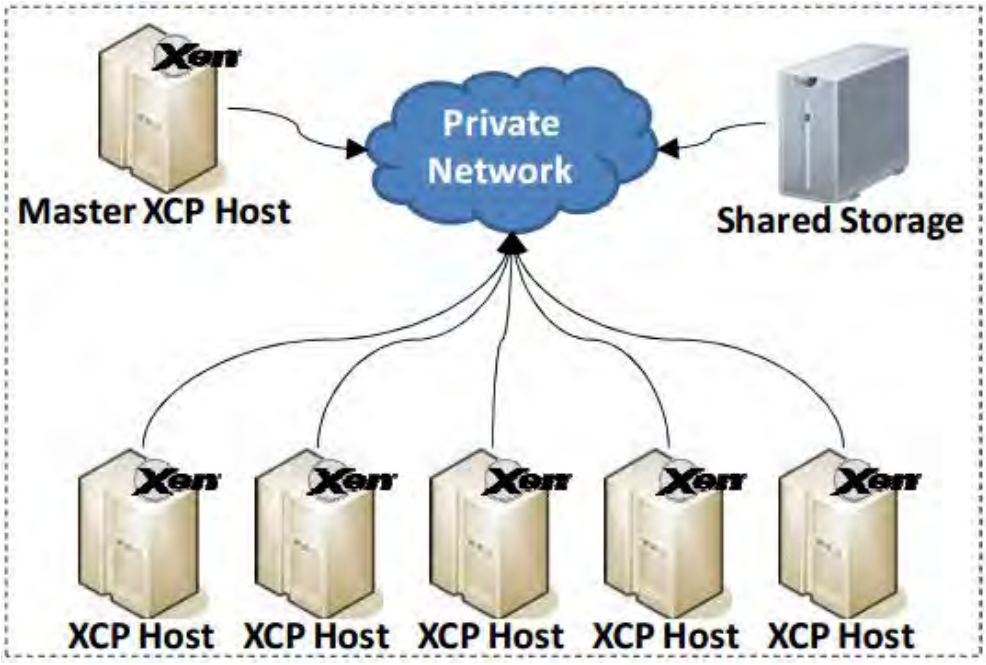

Figure 3.4 XCP architecture (adapted from [129]).

Xen Cloud Platform 1.6 has been installed on a 64bit, 8GB RAM Intel core i5 with 500GB hard disk machine. In this deployment, 4 virtual machines that run different cloud services for cloud users were deployed as the XCP host. These four hosts run Centos 6.5, Debian 7.6, Ubuntu 12.04LTS and Ubuntu 14.04 LTS, respectively (see [207] for Nmap OS signature database). Ubuntu 12.04 was made the master host, and serves as a front-end to the cloud. Nmap, P0f, and Wireshark were installed on the front end (master XCP). Algorithm 3.5.1 describes our matching algorithm for the passive and active stages during the working mode.

Algorithm 3.5.1 Passive and active matching algorithm.

For each packet during DDoS;

In Passive stage:

extract interesting TCP/IP header field attribute;

map values to OSp;

In Active stage:

send probe packet to source IP;

extract interesting TCP/IP header field attribute;

map values to OSa

If OSp ≠ OSa;

the packet is spoofed;

If OSp = OSa;

compare TTLf

If TTLf(p) ≠ TTLf(a);

packet is spoofed,

Else

the packet is legitimate

3.6 The Findings

Legitimate and malicious DDoS packets were generated from remote stations running different OSs directed towards the front end of our cloud computing environment. During passive and active OS fingerprinting, the TCP/IP header fields were extracted and analysed. The Table 3.1 below shows different remote stations establishing connection with the front-end of our cloud environment. Their TCP/IP header values were captured using Wireshark, as shown in Table 3.1.

Table 3.1 TCP/IP header format values of connecting hosts

OS release	Product name	Kernel version	TTL	DF	WSS	Total length
Centos 6.5	Final	2.6	64	1	14600	44
Ubuntu12.04	Precise	3.11	64	1	5840	56
Ubuntu14.04	Trusty	3.13	64	1	29200	44
Debian 7.6	Wheezy	3.2	64	1	14600	44
Windows 7	Win 7	6.1	128	1	8192	44
Windows 8	Win 8	6.2	128	1	8192	44

In analysing the interesting TCP/IP header features with the connecting SYN and SYN+ACK packets, it was deduced that some of the features observed in the SYN+ACK packet were a reflection of the SYN packet used for probing. This was taken into consideration, when reconciling the header features for different OSs. It was also observed that some OSs have similar feature values toeing the pattern of their OS flavours. Linux flavours have an initial TTL value of 64, as opposed to windows with 128. The window size also varies slightly along the line of OS flavours. The distribution of the window size and the initial TTL value against the kernel version is shown in Figures 3.5 and 3.6. These common features are the reasons why some OS fingerprinting tools give a range of guesses during OS identification.

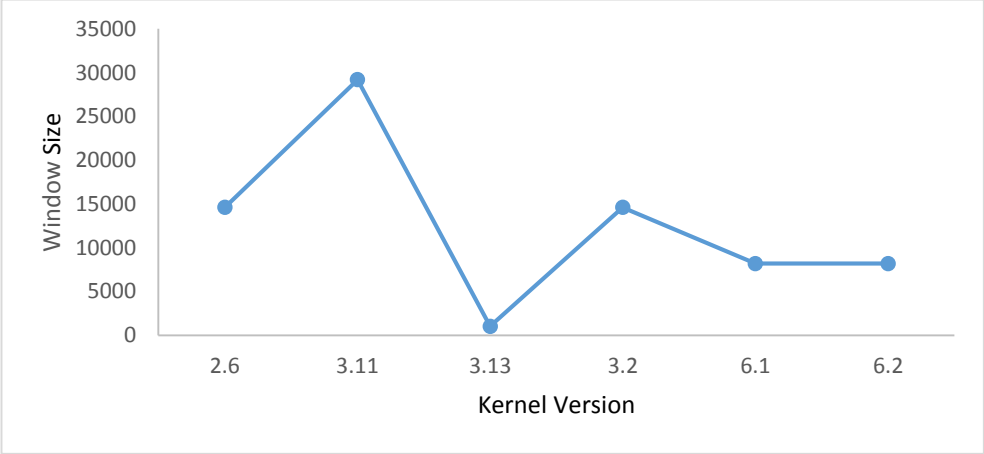

Figure 3.5 Kernel version distribution vs Window size.

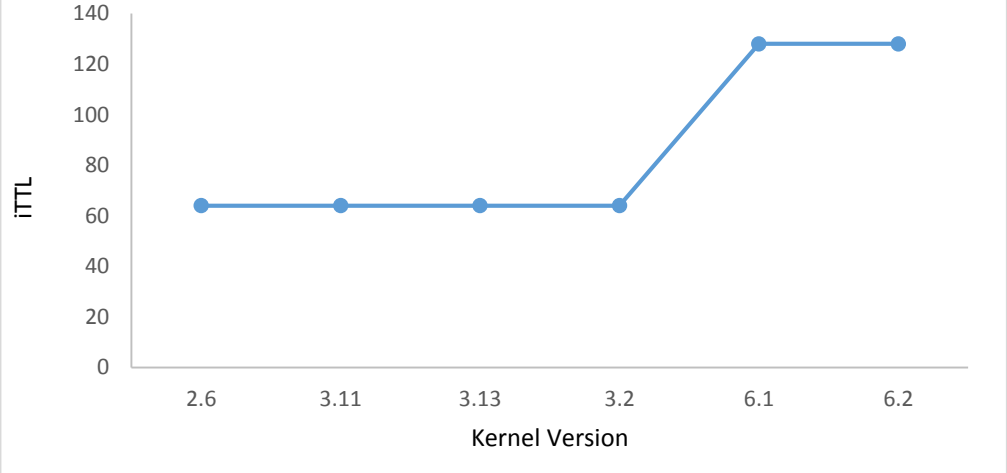

Figure 3.6 Kernel version distribution vs iTTL.

To determine the true source, peradventure there is a similar match during spoofed DDoS detection; this research matches the final TTL value of the active and passive OS detections. Our assumption is that each packet traverses the same distance to its destination.

3.7 Evaluation

To evaluate our proposed methodology, this research established an ingress traffic towards the front-end of the cloud by considering two different scenarios. Scenario I: ingress traffic connection establishment involving only legitimate user access to the cloud. Scenario II: ingress traffic connection involving both the legitimate cloud users and the malicious users with spoofed IP address.

Scenario I

During connection establishment to the front-end from a remote source, the SYN packet of the TCP connection during the 3-way handshaking was analysed by the p0f during the passive stage. It identified the connecting machine to be running Windows 7. During the active stage, specially crafted probe packets was sent to the source IP and the response of the OS discovery returned Windows 7 as the OS of the probed system. This is depicted in Figure 3.7.

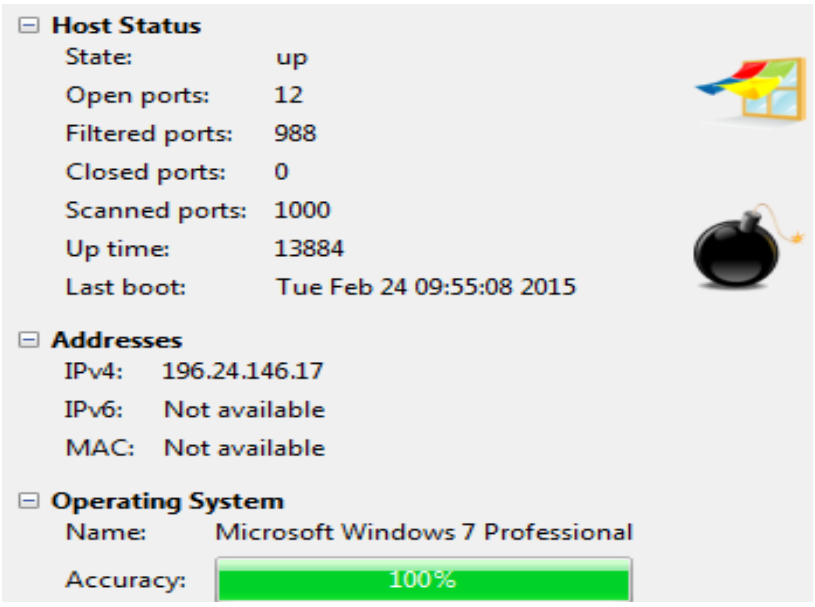

Figure 3.7 Active OS output using Nmap.

Scenario II

In the second scenario, both the legitimate and the DDoS attack with spoofed IP, using a traffic generator, was launched from a remote location. During the working mode, the source IP address of an incoming packet was used to verify the genuine source of the packet by identifying the remote host. A malicious connection with spoofed IP address was also identified with an OS mismatch during the passive and active stage. During the passive stage, OSp was determined to be Windows XP as shown in Figure 3.8 while OSa response of the OS discovery returned Windows 7. This was achieved by matching OSp to OSa (- see Algorithm 3.5.1)

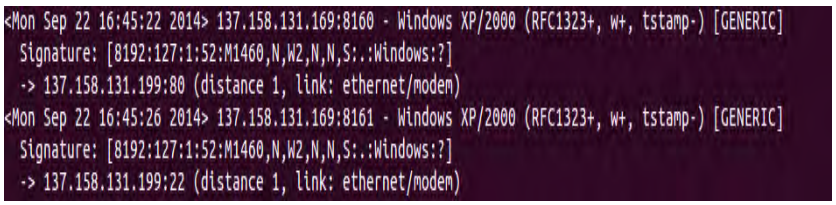

Figure 3.8 OSp P0f Windows output.

As stated earlier, after matching the OS during passive and active stages to detect spoofed DDoS by identifying the remote host's OS, there is a likelihood that both the spoofed and the true source might run similar OSs, as is evident in Figure 3.2, where windows 7 OS has a large share of the entire OS. Therefore, this research used the final value of the TTL value to determine whether the SYN and SYN+ACK packets originated from the same source by deploying the matching algorithm displayed in Algorithm 3.5.1

3.8 Summary

This chapter has reviewed the different methods employed to detect a spoofed DDoS attack. The interesting TCP/IP header features were analysed, thereafter both active and passive OS fingerprinting was proposed. This uses the SYN and SYN+ACK during connection, in order to verify the true source of an incoming packet in the front-end of the cloud computing environment. Due to the OS distribution, the final TTL

value was used to determine the spoofed DDoS, when the spoofed and true source run similar OSs. The key benefit of this technique is its ability to detect spoofed IP by determining the true source of an incoming packet during a spoofed DDoS attack. It presents little or no complexity when using the packet header field.

The next chapter presents a feature selection method that identifies the important features in a dataset, in order to optimally detect a DDoS attack in cloud computing.

Chapter 4: Ensemble-based Multi-Filter Feature Selection Method for DDoS Detection in Cloud Computing*

4.1 Introduction

In the previous chapter, the spoofing of IP addresses during DDoS attacks was addressed by proposing a host-based operating system (OS) fingerprinting that uses both passive and active methods to match the OS of incoming packets from its database. This serves as the first level of defence towards ensuring a high level of availability during DDoS attacks.

Due to the increase in the amount of data that needs to be processed [163-164] [169-170] during DDoS detection, feature selection methods have been proposed for pre-processing the data before classification, in order to identify the important features of a dataset, with the aim of improving the prediction accuracy and reducing the computational complexity. Therefore, this chapter presents an ensemble-based multi-filter feature selection (EMFFS) method that combines the output of information gain (IG), gain ratio, chi-squared and reliefF to select the important features.

The aim of the proposed method is to significantly reduce the feature set, while maintaining or improving the classification accuracy using a decision tree classifier. An intrusion detection benchmark dataset, NSL-KDD, consisting of 41 features is used to evaluate the efficiency of our proposed method in the Waikato environment for knowledge analysis (Weka).

The widespread adoption of cloud computing has increased the attractiveness of such services to cybercriminals. Despite the benefits offered by the use of cloud computing, it could nevertheless be exploited or targeted by cybercriminals, including State-sponsored actors (see [165]). This is not surprising, as popular consumer technologies, such as wireless sensor networks have also been reportedly targeted [166] [167]. One common form of attacks targeting cloud computing is the distributed denial of service (DDoS) attacks [130] [102].

*Material presented in this chapter has appeared in [195].

This thesis refers interested reader to [168] for other cloud-related security and privacy issues.

The existing defence methods that are capable of handling significant amounts of data generally contain redundant features, which result in excessive training and classification time [132].

Feature selection methods have been used in a wide range of applications, such as statistical pattern recognition, machine learning and data mining for data reduction, in order to achieve improved performance and detect outliers. Current feature selection methods can be broadly categorised into: filter, wrapper, and embedded approaches. In filter methods, the attributes are categorised, according to the intrinsic information of the data; and this is independent of the classification algorithm [133]. The features are assessed and ranked, according to their inherent properties – using simple measurements, such as distance, dependency and information [134].

Such methods are particularly efficient when dealing with large datasets, when compared with wrapper methods that provide a more precise result, but are more time-consuming [135]. Wrapper and embedded methods require a specific classification algorithm to determine the importance of a feature subset.

Recent studies have shown that combining feature selection methods can improve the performance of classifiers by identifying features that are weak as an individual but strong as a group [138], removing redundant features [135], and determining features that have a high correlation with the output class. Other methods have proposed a hybrid feature selection that combines both filter and wrapper. Filter feature selection represents a popular method that uses ranking and space search techniques.

The performance of a classification problem depends on the relevance of the selected attributes with regard to its class. Feature selection methods have been applied in classification problems to select a reduced feature subset from the original set, in order to achieve a faster and more accurate classification. Similar to many data mining and machine-learning techniques, two key factors are involved in building an optimum classifier: feature and model selection [136]. Selecting the right feature can be quite a challenging task; and several methods have been proposed to solve this and to discard any redundant, irrelevant and noisy features.

Wang and Gombault [137] proposed a filter selection method using IG and chi-squared to extract the nine most important features in the network traffic. Bayesian Network and C 4.5 (a decision tree classifier) were used to detect DDoS attacks in the network. The results obtained show that the detection accuracy remains the same while the overall efficiency improves. Bolon-Canedo et al. [138] combined discretizers, filters, and classifiers to improve the classification performance by significantly reducing the feature set. This was applied to both binary and multi-class classification problems, using the KDD Cup 99 benchmark dataset. A supervised inductive learning approach, a group method for data handling (GMDH) was proposed in [139], using monolithic and ensemble-based techniques. Filter feature selection methods using IG, gain ratio and GMDH were used to rank the features during the pre-processing phase. Lin et al. [140] proposed an anomaly intrusion detection that detects new attacks by using a support vector machine (SVM), a decision tree (DT), and simulated annealing (SA). The best features were selected from the KDD'99 dataset, using SVM and SA to improve the classification accuracy of DT and SVM, to detect new attacks.

Li et al. [143] proposed a gradual feature removal method that processes the dataset prior to combining the cluster method, an ant colony algorithm and SVM to classify the network traffic as either normal or anomalous. Sindhu et al. [141] proposed a wrapper method for feature selection to remove any irrelevant instances from a feature set to achieve higher detection accuracy using a neuro tree. A feature selection approach was proposed in [144] using the Bayesian Network; and NSL-KDD dataset was used to evaluate the selected features. The findings indicated that these features decreased the attack detection time and improved the classification accuracy, as well as the true positive rates.

Bhattacharya et al. [142] proposed a multi-measure multi-weight ranking approach that identifies important network features by combining wrapper, filter and clustering methods to assign multiple weights to each feature.

A rough set feature selection approach has proven to be an efficient mathematical tool, based on the upper and lower approximation. It presents equal classification capability with minimal subsets. Olusola et al. [145] proposed a rough set based feature selection method that selects the important features from the input data, using the KDD '99 dataset. Sengupta et al. [146] designed an online intrusion detection system (IDS) using both rough set theory and a Q-learning algorithm to achieve a maximum

classification algorithm that classifies the data as either normal or anomalous using the NSL-KDD network traffic data. A fast attribute reduction algorithm based on the rough set theory was proposed in [162]. The algorithm identifies the important features and discards any independent and redundant attributes to achieve an effective classification performance.

A review of the literature suggests that there are three general trends in feature selection, irrespective of the method used, namely: (1) proposed methods search and identify correlated features in the dataset, in order to remove the redundant features; (2) proposed methods identify the unique features that contain important information about different output classes in the data and discard those with little or no information; and (3) some features have been identified to be strong as a group, but weak individually. In the filter feature selection method, the features are ranked independently, according to their strength in predicting the output class.

Unlike previously proposed methods, this work presents an algorithm that supports data mining and security defence for cloud DDoS attacks, using a minimal feature set. Filter feature selection methods present different ranking algorithms; therefore, in this work, an EMFFS method that combines the output of IG, gain ratio, chi-squared and reliefF to find common features in the one-third split of the ranked features is proposed using a NSL-KDD benchmark dataset in the Weka environment. The dataset feature is reduced from 41 to 13 and J.48, a version of C4.5 decision tree classification algorithm is used to classify the data as either normal or anomalous. The rest of the chapter is organised as follows: Section 4.2 presents the proposed EMFFS method, while Section 4.3 describes the EMFFS execution process. The classification algorithm and the benchmark dataset are presented in Sections 4.4 and 4.5, while Section 4.6 presents the experimental findings. Finally, Section 4.7 summarizes the chapter.

4.2 Ensemble-based Multi-Filter Feature Selection Method

The filter feature selection method is a pre-processing phase for selecting important features from a dataset; and it is independent of the classification algorithm. The filter methods rely on the statistical intrinsic test in an original training dataset; and use a feature ranking scheme as the main criterion for feature selection by ordering. The

features are scored; and a pre-determined threshold is used to remove the features below the threshold. Due to its simplicity, it has been widely used for practical applications, including cloud computing, involving a huge amount of data. This section describes the proposed ensemble-based multi-filter feature selection method that combines the output of four filter selection methods – IG, gain ratio, chi-squared and reliefF – to harness their combined strength to select 13 common features among them.

Information Gain

One of the filter feature selection methods used in determining the relevant attributes from a set of features is IG. IG works by reducing the uncertainty associated with identifying the class attribute, when the value of the feature is unknown [148]. It is based on the information theory, which is used in ranking and selecting the top features to reduce the feature size before the start of the learning process.

The entropy value of the distribution is measured, in order to determine the uncertainty of each feature prior to ranking, according to their relevance in determining the different classes [147]. The uncertainty is determined by the entropy of the distribution, sample entropy, or the estimated model entropy of the dataset. The entropy of variable X [149] can, therefore, be defined as:

$$H(X) = -\sum_{i} P(x_i) log_2\big(P(x_i)\big) \quad (1)$$

Let $P(x_i)$ denotes the value of the previous probabilities of X. The entropy of X after observing the value of another variable Y is defined as:

$$H(X/Y) = -\sum_{j} P(y_j) \sum_{i} P(x_i|y_j) log_2 \Big(P\big(x_i|y_j\big)\Big) \quad (2)$$

In Equation 2, $P\big(x_i|y_j\big)$ is the posterior probability of X, given Y. The information gain is defined as the amount by which the entropy of X decreases, in order to reflect an additional information about X provided by Y; and it is defined as:

$$IG(X/Y) = H(X) - H(X|Y). \quad (3)$$

Based on this measure, it is clear that features Y and X are more correlated than features Y and Z, if $IG(X/Y) > IG(Z/Y)$. The feature ranking can, therefore, be

calculated by using Equation 3. This ranking will be used to select the most important features (see Appendix A.1 for details).

Gain Ratio

The gain ratio was introduced to improve the bias of IG towards features with a large diversity value [139]. When the data are evenly spread, the gain ratio exhibits a high value; while it gives a small value when all the data belong to only one branch of the attribute. It uses both the number and size of the branches to determine an attribute; and it corrects IG by considering the intrinsic information [150]. The intrinsic information of a given feature can be determined by the entropy distribution of the instance value. The gain ratio of a given feature x and a feature value y can be calculated [150] by using Equations 4 and 5.

$$Gain\ Ratio\ (y,x) = \frac{Information\ Gain(y,x)}{Intrinsic\ Value(x)} \quad (4)$$

Where,

$$\text{Intrinsic Value } (x) = -\sum \frac{|S_i|}{|S|} * Log_2 \frac{|S_i|}{|S|} \quad (5)$$

Note that $|S|$ is the number of possible values feature x can take, while $|S_i|$is the number of actual values of feature x. In this research, 14 features were selected, representing one-third of the ranked features, using the NSL-KDD benchmark dataset. These 14 features represent the highest ranked feature obtained by using the gain ratio (see Appendix A.2 for details).

Chi-squared

The chi-squared (χ^2) statistic is used to test the independence of two variables by computing a score to measure the extent of independence of these two variables. In feature selection, χ^2 measures the independence of the features with respect to the class. The initial assumption of χ^2 is that the feature and the class are independent before computing a score [151]. A score with a large value indicates the existence of a high dependent relationship. Chi-squared [152] can be defined as:

$$\chi^2(r, c_i) = \frac{N[P(r,c_i)P(\bar{r},\bar{c}_i) - P(r,\bar{c}_i)P(\bar{r},c_i)]^2}{P(r)P(\bar{r})P(c_i)P(\bar{c}_i)} \quad , \quad (6)$$

Where N denotes the entire dataset; and r indicates the presence of the feature ($\bar{r}$ its absence), c_i refers to the class. $P(r, c_i)$ is the probability that feature r occurs in class c_i and $P(\bar{r}, c_i)$ is the probability that the feature r does not occur in class c_i. Also, $P(r, \bar{c}_i)$ and $P(\bar{r}, \bar{c}_i)$ are the probabilities that the features does or does not occur in a class that is not labelled c_i and so on. $P(r)$ is the probability that the feature appears in the dataset; while $P(\bar{r})$ is the probability that the feature does not appear in the dataset. $P(c_i)\ and\ P(\bar{c}_i)$ represent the probability that a dataset is labelled as belonging to class c_i or not. In this research, 14 features were selected using chi-squared statistic, representing one-third of the ranked features, using the NSL-KDD benchmark dataset (see Appendix A.3 for details).

ReliefF

The reliefF feature selection method uses continuous sampling to evaluate the worth of a feature – to distinguish between the nearest hit and the nearest miss (nearest neighbour from the same class and from a different class) [153]. The attribute evaluator is used to append weight to each feature, according to its ability to distinguish the different classes. A user-defined threshold is determined and the weights of those features that exceed this threshold are selected as important features [151]. ReliefF evolved from the original Relief algorithm [154], and was developed to improve its limitations. Among the key attributes of ReliefF are its ability to deal with the multiclass problem, and its robust capability of dealing with noisy and incomplete data. A key advantage of ReliefF over other filter methods is that it has a low bias, and can be applied in all situations. This has been used to rank features in NSL-KDD benchmark dataset (see Appendix A.4 for details).

4.3 EMFFS Execution Process

Our proposed EMFFS method uses the output of the one-third split of ranked features of the filter methods described above. EMFFS is a pre-processing phase prior to learning, where individual filter methods are used for the initial selection process. The IG, the gain ratio, chi-square and the reliefF filter methods are used to rank the feature set of the original dataset to create a mutually exclusive subset before selecting a one-

third split of the ranked features (i.e., 14 features). These features are considered as the most important feature with respect to each filter method.

The resulting output of the EMFFS is determined by combining the output of each filter method and using a simple majority vote to determine the final selected feature. A threshold is determined to identify the frequently occurring features among the four filter methods and set to 3 (i.e., *T*=3). After combining all the selected feature sets, a counter is used to determine the common features with respect to the threshold set. Features that meet the threshold criteria are selected and used as the final feature set for classification. Figure 4.1, shows the proposed EMFFS method.

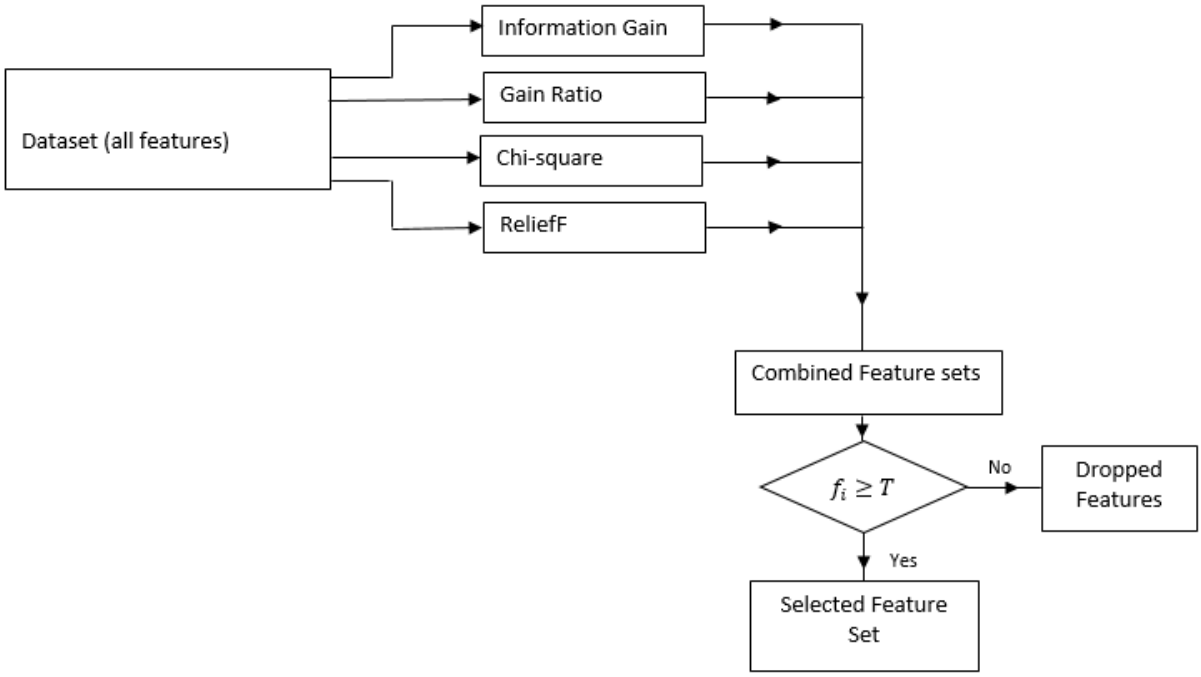

Figure 4.1 Ensemble-based Multi-Filter Feature Selection Method.

The EMFFS method is constructed through the algorithms presented below:

Algorithm 4.3.1 (Filter feature ranking methods)

Step 1: Let X_i be the feature set in the NSL-KDD dataset,

where $X_i = \{X_1, X_2, X_3 \ \ldots\ldots\ldots\ldots, X_{41}\ \}$ and

C_i represents the class (i.e., normal or anomalous), where $C_i = \{\ C_1, C_2\}$.

Step 2: For each filter method, rank and sort the features X_i according to their importance in determining the output class C_i.

Step 3: Select a one-third split of each filter selection method's output X_i'

Algorithm 4.3.2 (Combine output features)

Step 1: Combine the selected output features X_i' of each filter method.

Step 2: Determine the feature count threshold *T*.

Step 3: Compute the feature occurrence rate among the filter methods.

Algorithm 4.3.3 (Ensemble selection)

Step 1: Choose the intercepts of common features from 4.3.2

Step 2: If the feature count is less than the threshold, drop the feature;

otherwise select the feature.

Step 3: Repeat step 2 for all the features in the one-third split subset.

4.4 Classification Algorithm

The decision tree classification algorithm is a popular data mining classifier for prediction due to the ease of understating and the interaction between the variables. It is based on a greedy algorithm that uses a divide-and-conquer strategy to recursively construct a decision tree [155]. The tree is made up of a root node, internal nodes, branches and leaves, which represent a rule used in categorizing the data according to its attributes. The decision tree uses a supervised dataset with the root node being the first attribute with the test condition to split each input towards individual internal node, in line with the characteristics of the data record [154]. The node with the highest information gain is the root node; and the preceding node with the next highest information gain is selected as the test for the next node. This process continues until all the attributes have been compared; or until all the samples belong to the same class – with no remaining attribute from which the samples can be further partitioned [156].

A branch connects two nodes together; and it can also connect a node and a leaf. Each node is made up of branches labelled as the possible value of attributes in the parent

node [140]. The leaves are labelled, according to the decision value of the classification.

Consider a case selected at random from a set S of cases which belong to class C_i. The probability that an arbitrary sample belongs to class C_i can be determined, as follows [156]:

$$P_i = \frac{freq\,(C_i, S)}{|S|} \quad , \qquad (7)$$

Where $|S|$ is the number of samples in the set S. Therefore, the information it conveys can be represented by $-log_2 P_i$ bits. Now, suppose the probability distribution is given as $P = \{P_1, P_2, P_3 \ldots\ldots\ldots\ldots, P_n\}$, therefore, the information carried by the distribution, that has an entropy of P, can be expressed as:

$$Info\,(P) = \sum_{i=1}^{n} -P_i\, log_2 P_i \qquad (8)$$

Partitioning a set of K samples, based on the value of a non-categorical attribute X, into sets $K_1, K_2, K_3 \ldots\ldots\ldots\ldots, K_m$, the information required to determine the class of an element of K is the weighted average of the information needed to identify the class of an element K_i. The weighted average of Info (K_i) can be determined by:

$$Info\,(X, K) = \sum_{i=1}^{m} \frac{|K_i|}{K} \times Info(K_i) \qquad (9)$$

The information gain, $Gain\,(X, K)$, can therefore be calculated as follows:

$$Gain\,(X, K) = Info\,(K) - Info\,(X, K) \qquad (10)$$

Equation 10 represents the difference between the information needed to identify an element of K and the information needed to identify an element of K after the value of attribute X has been determined. Therefore, it follows that this is the information gain due to attribute X.

There are different algorithms for implementing the decision tree: C5.0 and its earlier version C4.5 have been described in [157], however, this thesis use J48, a version of C4.5 as the classifier.

4.5 Benchmark Datasets

The NSL-KDD dataset is an improved version of KDDCUP'99 widely deployed in the literature [144] [146] [161] for intrusion detection. This was used to validate our proposed algorithm. NSL-KDD is a labelled benchmark dataset developed from KDDCUP'99 to improve on its flaws. Researchers have identified several issues associated with the use of KDDCUP'99, such as the existence of large redundant records (which may result in the learning algorithm being biased towards frequently occurring records) and its high complexity [93]. NSL-KDD is used for evaluating the network intrusion systems; and it is made up of selected records from the initial KDDCUP'99.

This presents a reduced dataset size that makes the evaluation of different research works consistent and the validation of the learning algorithm complete, easy and affordable. NSL-KDD is made up of 41 features; and it is labelled as either attack or normal (see Table 4.1). These features are categorized into four groups, namely: basic, content, time-based traffic and connection-based traffic [142]. NSL-KDD comprises both the training and the testing datasets. The former is made up of 21 attack types while an additional 17 novel attack types are used for the test set [144].

The attacks are grouped into four categories: DoS, Probe, U2R, and R2L. While the distribution of the training dataset consists of 67343 normal (53.46%), 45927 DoS (36.46%), 11656 Probe (9.25%), 995 R2L (0.79%) and 52 (0.04%) U2R; the testing dataset on the other hand, contains 9711 normal (43.08%), 7456 DoS (33.08%), 2421 probe (10.74%), 2756 R2L (12.22%) and 200 U2R (0.89%).

From the attack distribution, it can be seen that DoS constitutes around 78.3% of the total attack. Therefore, this research uses 20% of the records in NSL-KDD train+ as the DoS training set that has been labelled as either attack or normal. A 10-fold cross-validation is used for both training and testing purposes. Table 4.1 describes the NSL-KDD feature dataset.

Table 4.1 NSL-KDD dataset features

#	Data features	#	Data features	#	Data features	#	Data features
1	Duration	12	Logged_in	23	Count	34	Dst_host_same_srv_rate
2	Protocol_type	13	Num_compromised	24	Srv_count	35	Dst_host_diff_srv_rate
3	Service	14	Root_shell	25	Serror_rate	36	Dst_host_same_src_port_rate
4	Flag	15	Su_attempted	26	Srv_serror_rate	37	Dst_host_srv_diff_host_rate
5	Src_bytes	16	Num_root	27	Rerror_rate	38	Dst_host_serror_rate
6	Dst_bytes	17	Num_file_creations	28	Srv_rerror_rate	39	Dst_host_srv_serror_rate
7	Land	18	Num_shells	29	Same_srv_rate	40	Dst_host_rerror_rate
8	Wrong_fragment	19	Num_access_files	30	Diff_srv_rate	41	Dst_host_srv_rerror_rate
9	Urgent	20	Num_outbound_cmds	31	Srv_diff_host_rate		
10	Hot	21	Is_host_login	32	Dst_host_count		
11	Num_failed_logins	22	Is_guest_login	33	Dst_host_srv_count		

4.6 Experimental Results

In this section, our proposed EMFFS method that pre-process the dataset to select the most important features for the decision tree classification algorithm, which classifies the data as either attack or normal in cloud computing, is described. Our analysis was carried out using Weka [158] that contains a collection of machine learning algorithms for data mining tasks. The parameters for classification in the experiments were set to the default values in Weka.

In this thesis, the NSL-KDD dataset has been used to evaluate the performance of our EMFFS method and the decision tree classifier using 10-fold cross-validation. In the 10-fold cross-validation, the data were divided into 10 folds of equal sizes before performing 10 iterations of training and validation. Within each iteration, a different fold of the data was used for validation; while the remaining nine folds are used for learning. All experiments were performed on a 64-bit Windows 8.1 operating system with 6 GB of RAM and Intel core i5-4210U CPU.

4.6.1 The Pre-processing Dataset

During the pre-processing phase, a feature selection was performed to determine the most important features of the NLS-KDD dataset, by ranking them, when using different filter methods. Fourteen (14) most important features of the filter methods

were determined by presenting the one-third split of the ranked features (see Table 4.2).

Table 4.2 Feature selection using filter methods

Filter method	Feature selected
Info Gain	**5**,**3**,**6**,**4**,**30**,**29**,**33**,**34**,35,**38**,**12**,**39**,**25**,**23**
Gain Ratio	**12**,26,**4**,**25**,**39**,**6**,**30**,**38**,**5**,**29**,**3**,37,**34**,**33**
Chi- Squared	**5**,**3**,**6**,**4**,**29**,**30**,**33**,**34**,35,**12**,**23**,**38**,**25**,**39**
ReliefF	**3**,**29**,**4**,32,38,**33**,**39**,**12**,36,**23**,26,**34**,40,31

After applying the algorithm 4.3.2 to the output of each of the four filter selection methods, this research searched for feature intercept and set the minimum threshold to 3. From Table 2, it can be seen that, even though each filter uses different ranking techniques, some features are common across all the different filter methods. Using a simple majority vote, features 4, 29, 34, 12, 39, 3, 5, 6,30,33,38, 25 and 23 (indicated in bold) appeared across more than three filter methods, this shows the level of importance these features are to the output class (see Table 4.3).

Table 4.3 Features selected by the Ensemble-based Multi-Filter Feature Selection (EMFFS) method

Filter method	Feature selected
EMFFS	3,4,29,33,34,12,39,5,30,38,25,23,6

Table 4.3 shows the 13 selected features out of the one-third split of the most important features of the NSL-KDD dataset, using the EMFFS method. This was used as the input features for training the decision tree classification algorithm, J48, in Weka.

4.6.2 Performance Measures

The performance of a classifier could be determined by using different metrics. Determining the accuracy usually involves the measure of True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). TP is the amount of attack classified correctly, while TN is the percentage of normal test samples classified correctly. FP is the number of attacks detected when it is indeed normal (false alarm);

and FN is the misclassification of a test sample as normal – when it is actually an attack.

The recently proposed mitigation strategies require a high detection rate and a low false alarm, therefore, in this work, a comparison of the accuracy, detection rate and the false alarm rate of our proposed EMFFS method with each filter method and the full dataset feature using the J48 classification algorithm is made. Furthermore, a comparison of the time required to build the classification model, which is the duration of the classifier's learning process after applying each feature selection method,is carried out.

Table 4.4 presents the results of the performance measure of the J48 classifier using the full dataset with 41 features, a one-third split of filter methods with 14 features, and our proposed EMFFS method with 13 features.

Table 4.4 Performance measure

Filter method	No of features	Accuracy	Detection rate	False alarm rate	Time
Full set	41	99.56%	99.49%	**0.38%**	2.75 Sec
Info Gain	14	99.66%	99.74%	0.41%	0.83 Sec
Gain Ratio	14	99.60%	99.68%	0.47%	1.12 Sec
Chi-squared	14	99.66%	99.74%	0.41%	0.92 Sec
ReliefF	14	99.08%	99.02%	0.87%	0.93 Sec
EMFFS	13	**99.67%**	**99.76%**	0.42%	**0.78 Sec**

Classification accuracy

Classification accuracy is the percentage of correctly defined data from the total set represented by the situation of TP and TN. The accuracy of the classifier can be determined by $ccuracy = \frac{TP+TN}{TP+TN+FP+FN} \times 100\%$. Figure 4.2 shows the classification accuracy across different filter feature selection methods and the EMFFS method. Our proposed method presents a slight increase in classification accuracy by an amount of 0.01%.

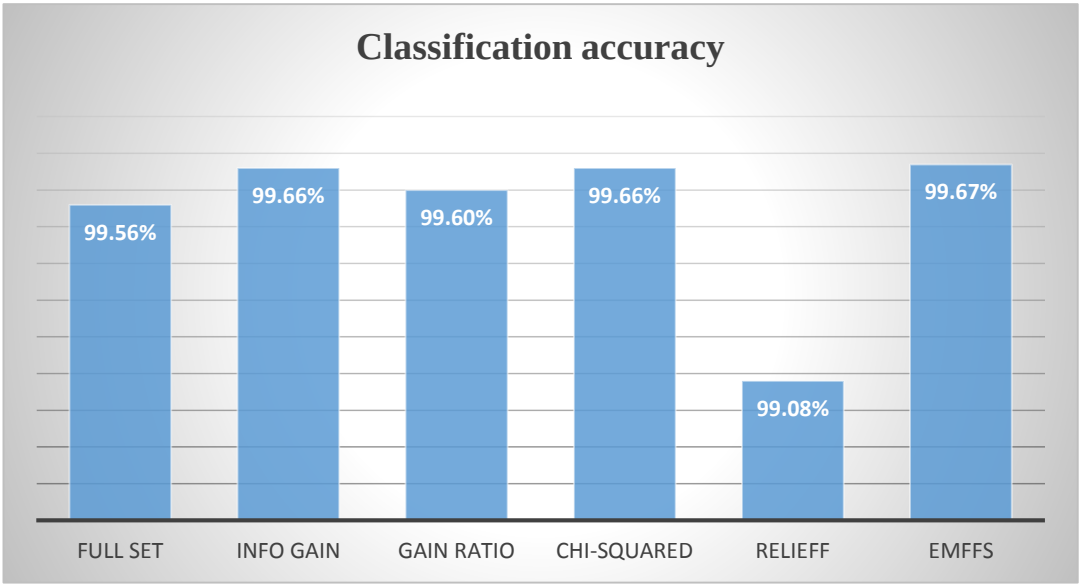

Figure 4.2 Classification accuracy for the filter methods.

Detection rate

The detection rate can be determined, based on the confusion matrix. It is calculated as: $detection\ rate = \frac{TP}{TP+FN} \times 100\%$. Figure 4.3 shows the performance of the EMFFS method in comparison with other filter feature selection methods. The findings demonstrated that our method, with 13 selected features, has a slight increase in detection rate by 0.02% when compared with the best filter feature selection method.

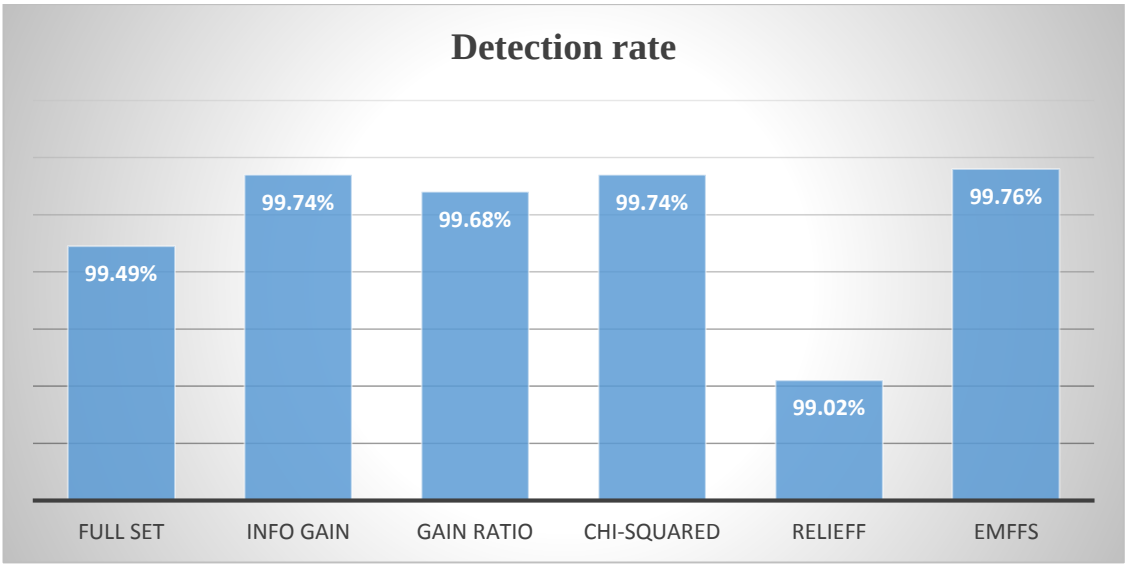

Figure 4.3 Detection rate for filter methods.

False alarm rate

A false alarm is the amount of normal data that have been falsely classified as an attack. This can be determined by $False\ alarm\ rate = \frac{FP}{FP+TN} \times 100\%$. Figure 4.4 shows the false alarm rate of the full feature set and different filter feature selection methods. ReliefF produces the highest false alarm rate; while the full feature set has the lowest rate with 0.38%. Our method presents a false alarm rate of 0.42%.

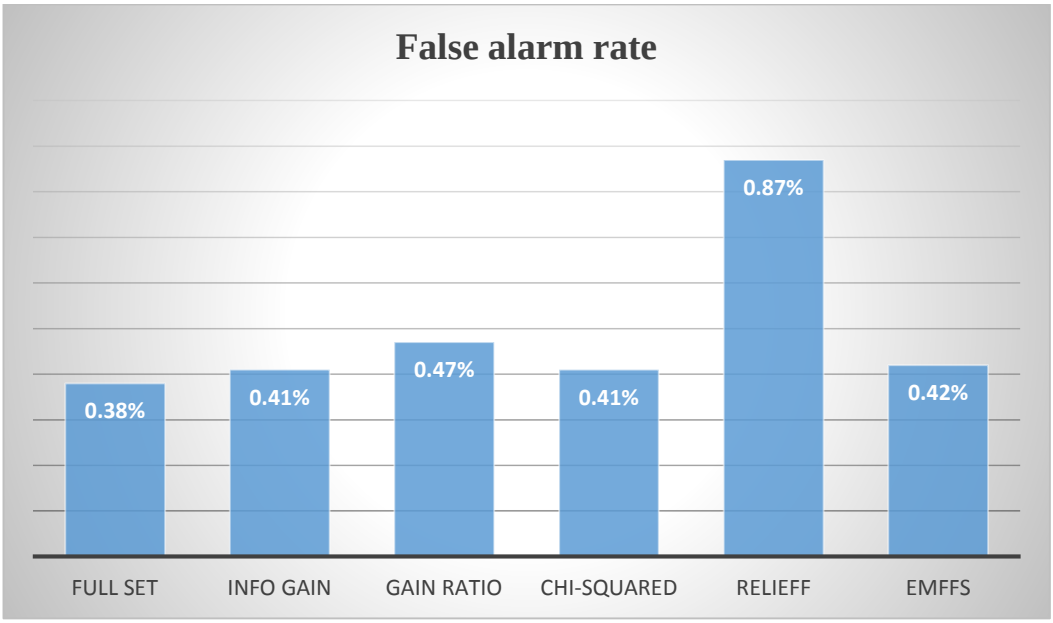

Figure 4.4 The false alarm rate for filter methods.

Time to build model

Figure 4.5 presents the time to build model in seconds across different filter selection methods and the full feature set. Our proposed method presents the best time with 0.78 seconds, when compared with other filter selection methods. The full feature set presents the worst learning time with 2.75 sec. This is due to the number of features the classifier has to process.

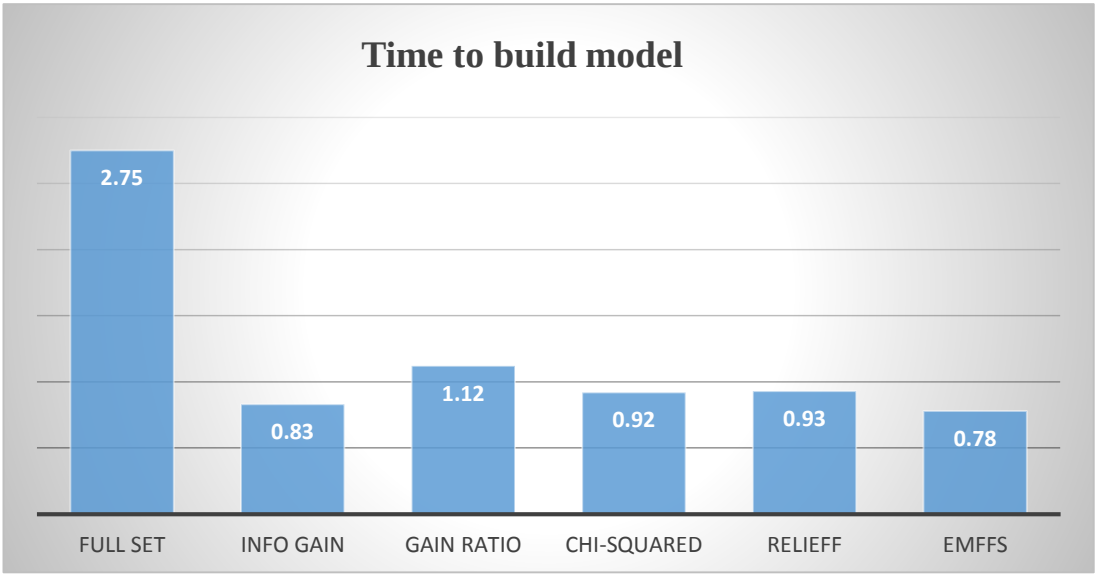

Figure 4.5 Time-to-build model for filter methods.

4.6.3 Discussion

The need for effective real-time classification of DDoS attacks in cloud computing increases the complexity of the detection techniques. The filter methods for feature selection have proven to be crucial when designing a lightweight detection system, which involves identifying the important features. In our proposed EMFFS method, 13 features were selected out of the available 41 features – by first presenting the output of the one-third split, when using four different filter methods. In this work, a threshold was determined; and a counter was used to select the important features by a simple majority voting. A comparison was made between our EMFFS method and the other filter methods with 14 features, and the full set consisting of 41 features when using the J48 decision tree classifier.

Our method with 13 features presents an improvement in classification accuracy and detection rate. This implies that the original dataset contains some levels of redundant features that have little or no contribution to make in identifying a particular class. For the time taken to build the model, our proposed method presents the best time. When compared with individual filter selection methods and the full feature set. This makes our ensemble-based multi-filter feature selection method more efficient with less complexity.

A comparison of the performance of our proposed method, EMFFS, with other methods in the literature was carried out by considering the number of features selected, the classification accuracy, and the time needed to build the model (see Table 4.5). This research observed that using 13 most important features with decision tree classifier, our method provides the best classification accuracy and a better learning time compared with the other schemes presented in Table 4.5.

Table 4.5 Performance comparison with other feature selection approaches

Approach	Classifier	No of features	Accuracy (%)	Time to build model (sec)
CFS [160]	C4.5	NA	99.13	NA[1]
CFS, CONS & INTERACT [136]	HNB_PKI_INT	7	93.72	NA[1]
Gradual feature removal [132]	Cluster methods, ant colony algorithm & SVM	19	98.62	NA[1]
CSE & CFS [159]	GA	32	78	NA[1]
Linear correlation-based [161]	C.45	17	99.1	12.02
Our method (EMFFS)	J48	13	**99.67**	**0.78**

NA[1]: Not available

4.7 Summary

One of the notable challenges faced by the current network intrusion systems in cloud computing is the handling of massive Internet traffic during DDoS attacks. Various feature selection methods have been used to pre-process the dataset prior to attack classification in cloud computing. This work has presented an ensemble-based multi-filter feature selection method that combines the output of one-third split of the ranked important features of information gain, gain ratio, chi-squared and reliefF. The resulting output of the EMFFS is determined by combining the output of each filter method. In this thesis, a set threshold is used to determine the final features by using a simple majority vote. The performance evaluation with the NSL-KDD dataset

demonstrated that the EMFFS method, with 13 features, achieves better performance than any of the individual filter feature selection methods using the J48 classifier and other proposed feature selection methods in the literature.

In the next chapter, this thesis describes a change-point cloud DDoS detection method, using packet inter-arrival time to classify the normal traffic packet from anomaly when using the CUSUM algorithm.

Chapter 5: Change-Point Cloud DDoS Detection using Packet Inter-Arrival Time*

5.1 Introduction

In previous chapters, this thesis first identified a common attribute of DDoS attack, which is the spoofing of source IP addresses; and it proposed a host-based OS fingerprinting that identifies the connecting hosts, according to their OS – to detect the true source of a packet during a spoofed DDoS attack. The EMFFS method has also been proposed as a pre-processing phase before classification, in order to compensate for the increase in the amount of data that need to be processed to achieve a higher classification accuracy.

It has been established that a DDoS attack pattern can take different forms: a constant rate attack, a pulsating rate attack, a gradual pulse attack, and an increasing rate attack [185]. To mitigate against these different forms of DDoS attack, this chapter harnesses the potential of change-point detection using the cumulative sum (CUSUM) algorithm. Change-point detection, a statistical anomaly detection technique, monitors and compares the features of the observed packet sequence against normal behavioural profile pattern obtained over a pre-determined period – with the aim of detecting any significant deviation.

More specifically, this work presents a novel traffic-flow pattern feature (hereafter referred to as packets IAT) to detect a DDoS attack and to evaluate the proposed approach.

High availability is an essential feature offered by cloud providers, particularly to paid-up cloud users, who can seek financial compensation if their resource availability falls below the level stated in a service level agreement.

Notwithstanding the various benefits offered by cloud computing, cloud services are subject to a range of cyber and physical threats [171, 172]. Although DDoS attacks are not new, they remain a threat to online businesses and cloud services [130] (e.g., DDoS attacks were identified as a key threat to cloud service availability [173]).

*Material presented in this chapter has appeared in [196].

DDoS attacks have grown: both in sophistication and size, since their "humble origin" in 1999 [53]. The availability and easy access to free and paid DDoS tools online has lowered the technical bar (e.g., no expertise required) in launching an attack [174, 175]. It is therefore not surprising that cloud DDoS mitigation remains a topic of interest, as evidenced in recently published studies [24] [176-177].

A number of proposed detection techniques use the bursty feature associated with DDoS attacks to differentiate DDoS traffic from normal traffic. However, there exists a seemingly similar behavioural pattern exhibited by a flash crowd (i.e., large groups of legitimate users attempting to access a resource concurrently). Differentiating DDoS attacks from flash crowds has also been studied in the literature [178, 179].

Change-point detection uses the statistical features of packets to detect anomalous behaviour; and this has been deployed in DDoS detection. MacDermott et al. [180] proposed the CUSUM algorithm to analyse and detect inter-cloud traffic for early DDoS detection. Fragkiadakis et al. [181] studied the performance of simple thresholds and the use of the CUSUM algorithm to detect network anomaly in the physical layer. The result shows that CUSUM performed better than the simple threshold in all types of attack intensities. Zhou et al. [174] proposed a distributed detection method, which employs adaptive CUSUM at the source end, using both the mean and variance values to detect attacks in outgoing traffic.

Detection at the intermediate network uses an adaptive CUSUM algorithm, based on the exponential weighted moving average (EWMA), in order to ascertain a change of traffic caused by abnormal aggregation. Tartakovsky et al. [182] proposed a score based multi-cyclic detection algorithm as a sequential change-point detection – to detect online anomalous traffic in computer networks.

Recently, DDoS attacks are becoming more sophisticated – by forging packet header information (e.g., TCP flags) to cause a high false negative during detection. For a DDoS attack to be deemed successful, it must consume substantial bandwidth and resources; both of which are a function of the packet IAT (the focus in this chapter).

Considering the shareability and elastic nature of the cloud to dynamically provide resources, DDoS attacks can result in catastrophic consequences to cloud providers, users, and other stakeholders. Therefore, this chapter proposes a change-point detection, based on the CUSUM algorithm (similar to the approach undertaken in [174, 13, 16]),

in addition to using the packet IAT attribute. Using IAT feature allows us to detect different forms of DDoS attacks; and to the best of our knowledge, this work is the first to combine the CUSUM algorithm and the packet IAT in cloud DDoS detection.

In a typical Internet communication, packets leaving a host traverse different links and nodes in the Internet before reaching their destination. During this process, the packets are likely to be subjected to random delays – due to link or nodal congestion. As a result, the observed packet IAT at the front-end may be affected by random noise. Even though this problem exists, this research believes that the IAT between packets of the same flow exhibits strong regularity (i.e., patterns). Therefore, this thesis assumes that the network condition for both normal and attack traffic is constant; as the measured IAT for both traffics is affected by the same network condition.

The rest of the chapter is organised as follows. Section 5.2 describes the proposed cloud DDoS change-point detection framework, while Section 5.3 presents the experimental simulation. Section 5.4 discusses the results obtained, and finally, Section 5.5 summarises the chapter.

5.2 Conceptual Framework

Figure 5.1 depicts a typical scenario of an incoming traffic accessing a cloud service – by either a legitimate user or an adversary. The number and frequency of hits (i.e., patterns) would probably vary across individual users. Consequently, the observed packets in a flow will vary with respect to their IAT; and they are non-stationary. In our context, this would be the IAT of the TCP packet attribute related to the HTTP traffic. The monitored period would be broken down into smaller windows; and the average IAT of the incoming packets would be determined. On the other hand, when analysing different forms of DDoS attack events, the observed traffic patterns are likely to have a constant rate attack (attack traffic rate hits a maximum and maintains it throughout the attack period), pulsating attack (attack traffic rate oscillates between zero and maximum), gradual pulse attack (attack traffic attains a maximum rate gradually, and maintains the rate before decreasing gradually), and an increasing rate attack (attack traffic gradually attains a maximum rate and stays constant until the attack terminates) [184]. These different forms of DDoS attack pattern follow a seemingly similar and

predictable pattern with regard to packet IAT due to their automation (e.g., use of bot malware). Therefore, this research will use this feature to detect DDoS in cloud.

Figure 5.1 presents a conceptual cloud DDoS change-point detection framework, the Flow-Based Classifier (FBC). FBC is responsible for classifying the incoming traffic accessing the cloud environment into different flows. This is achieved by inspecting the packets' header content. Similar to the approach undertaken in [185], a flow is defined as one that has successive packets with the same 5-tuple of protocol type, source IP address:port, destination IP address:port. From the traffic flow, the IAT is computed and used to detect any changes in the packet sequence. To achieve this, a measure of normal IAT pattern is determined *a priori*. This is used to detect the presence of a pattern change during a DDoS attack on a per packet basis, using the CUSUM algorithm. If an attack traffic is detected, an alarm is triggered; and the packets in the flow are dropped.

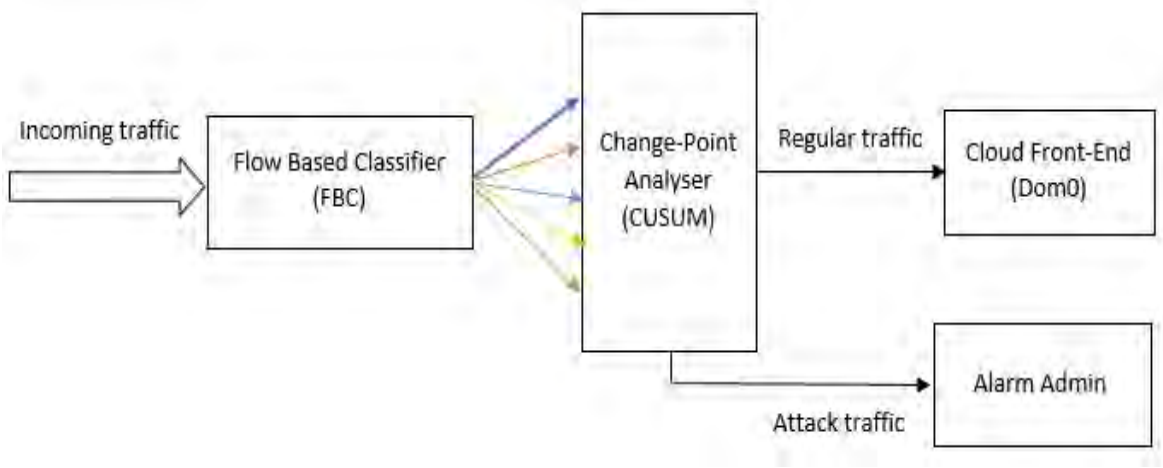

Figure 5.1 Framework of the proposed DDoS Change-point detection in cloud.

5.2.1 Choice of Detection Feature

The key aspect of our detection method is the choice of an improved feature for detecting DDoS in cloud computing compared with the previous work. From the literature, the packet IAT has been applied in areas of network performance monitoring [186] and the identification of applications over the Internet [187]. Arshadi et al. [188] used the IAT distribution of a TCP flow to describe a normal traffic, which conforms to a Weibull distribution against an anomaly that deviates from this. This work monitors the packet IAT of a flow by determining the amount of time that elapses between the receipt of a packet and subsequent packets. The inter-arrival distribution

is then used to determine the probability of occurrence of a DDoS attack in a traffic flow.

Let the observed packet *P* be defined by its packet inter-arrival time P_t in a traffic flow. The packet inter-arrival time P_t can, therefore, be expressed as: $P_t = A_{t+1} - A_t$ which is the time separation between two consecutive packet-arrival times A_{t+1} and A_t of a flow, for *t=0, 1, 2, 3, 4……N.*

This research leverages the attribute of the DDoS attack that exhibits a seemingly predictable inter-arrival pattern – due to its automation, which is exponentially distributed.

5.2.2 Change-Point Detection

Statistical methods used in detecting an anomaly involve the use of a sequential change-point algorithm that records an observation and stores it as an input [189]. The approach flags an alert on detecting an unusual trend in the monitored event, thereby signifying that a change has occurred. The emergence of change-point detection can be traced back to the 1920-1930s; and it was motivated by quality control requirements [190]. The Shewhart chart [191] was the early dominant method before sequential statistics was developed by Wald [192]. The works of Shewhart and Wald have been the bedrock of most literature in the theory and practice of sequential change detection [190].

A DDoS attack affects the statistical features (e.g., mean and variance) of a packet flow with temporal fluctuations. Change-point detection can therefore be used to determine the point in time a change occurs in an observed series of statistically homogeneous events.

According to [183], this can be broadly categorized into: (a) Fixed-size batch detection and (b) sequential change-point detection. The former monitors the changes in the mean value of every fixed length period consisting of *n* observations from the random variable while the latter monitors the successive variables, in order to make an on-the-spot detection of a change in the pattern. Sequential change-point detection does not require a lot of memory or computational overheads, therefore, in this work, our task is modelled by using sequential change-point detection.

5.2.3 CUSUM Algorithm

The CUSUM algorithm is a sequential detection technique, which can be used to detect irregular traffic patterns that cause changes in the normal traffic flow. Using this algorithm, one can observe the packet IAT of a traffic flow at the front-end of the cloud (i.e., dom0) and decide whether to grant or deny access to the cloud service or the resources.

Let P_m(λ ,t) represent the distribution of the monitored packets in a traffic flow during a sample period with packet inter-arrival time X_t at time t with mean λ. Suppose the series of packets in a traffic flow contains a DDoS attack, there would be an abrupt increase in the IAT value X_t when $t > \tau$, where τ is the point of change in the packet IAT in the traffic flow. $P_m(\lambda, t)$ can, therefore, be characterized into distribution of the pre-attack (normal) period $P_m(\lambda_0, t)$ and the attack period $P_m(\lambda_1, t)$, as expressed in Eqn. (1):

$$P_m(\lambda, t) = \begin{cases} P_m(\lambda_0, t); t \leq \tau \\ P_m(\lambda_1, t); t > \tau \end{cases} \quad (1)$$

The average IAT denoted by λ of X_t will take the value λ_0 when $t \leq \tau$ during a normal period, and λ_1 during an attack period, when $t > \tau$. Where:

$\lambda_0 = \frac{1}{n}\sum X_t$, for $t \leq \tau$ and $\lambda_1 = \frac{1}{n}\sum X_t$ for $t > \tau$

Due to the bursty nature of the DDoS attack, the mean IAT $\lambda_1 > \lambda_0$. Therefore, to detect the change due to this increase, a change-point detection mechanism is deployed using the CUSUM algorithm.

As mentioned earlier, a DDoS attack comes in different forms; and modelling a complex Internet system, using a parametric description, is not trivial. The CUSUM deployed is a statistical process control algorithm that dynamically detects the point of change in mean value from the observed series and responds appropriately.

To compute the CUSUM statistic, this work adds the difference between the current value and the average of the previous sum, as expressed in Eqn. (2).

$$C_i = C_{i-1} + (X_i - \lambda) \; for \; i = 1,2,3, \ldots N \quad (2)$$

In Eqn. (2), X_i represents the IAT between packets in a traffic flow; and λ is the mean IAT of the sampled packets.

The general assumption of the CUSUM algorithm is that the mean of the random sequence during the normal condition is negative; and it only becomes positive when a change occurs [178]. Using the IAT as a feature for detecting change within a monitored traffic packet in a flow follows a similar pattern. The CUSUM value during a normal traffic condition swings from (approximately) zero to a negative value, and then abruptly moves towards a positive value on detecting a change [193].

5.3 Simulation

In this section, SYN flooding attacks were simulated, since these are the most common type of flooding attacks [194], whose sole aim is to consume the available resources and cause a denial of service. This type of attack is a volume-based anomalous traffic flow. For our simulation, the CAIDA DDoS dataset [23] was used. This is made up of an hour of anonymized traced from DDoS attacks on August 4, 2007, between 20:50:80 and 21:56:16 UTC, split into 5 minutes pcap files. Non-attack packets had been removed from the dataset (i.e., only attack traffic and responses remained in the dataset). The dataset was first pre-processed by separating it into different protocols (i.e., ICMP and TCP), before splitting the TCP packets into different flows by using FBC. This is because the TCP is our protocol of interest. The attack IP was anonymized to 192.95.27.190 and the victim 71.126.222.64. Figures 5.2 and 5.3 show a plot of IAT against packet numbers in a flow for attack and normal packets.

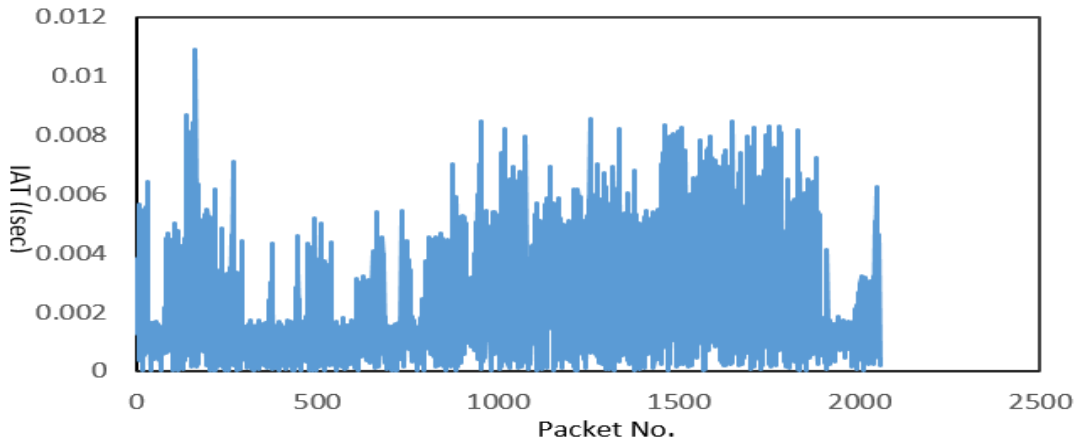

Figure 5.2 Graph of IAT against packet numbers during a SYN flooding attack.

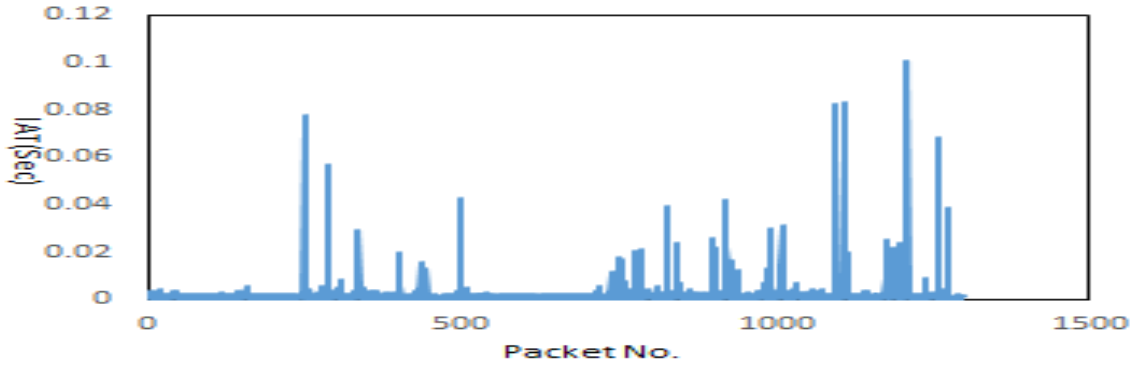

Figure 5.3 Graph of IAT against packet numbers during a normal traffic flow.

The pattern and behaviour of the DDoS attack flow can be determined by studying the IAT attribute between the packets in a flow. As observed in Figure 5.2, the IAT of the traffic flow exhibits a pulsating DDoS attack, while 5.3 exhibits a subtle distribution with some noticeable peaks in the IAT pattern.

For the normal traffic flow, a publicly available benchmark traffic model from Auckland-VIII dataset, Waikato Internet Traffic Storage (WITS) [34] was used. From the repository of a continuous 13 days packet header trace taken in December 2003 at the University of Auckland, the dataset from Dec 2 05:00:00 2003 to Dec, 2 05:59:59 2003 was selected. The capture point is between the University's network infrastructure and the Internet; and all the IP addresses were anonymized. Similar to the DDoS attack dataset, the dataset was pre-processed to extract TCP packets, before separating the traffic into different flows, using FBC. Figure 5.3 shows the graph of IAT against the packets of a reference normal flow.

Two separate experiments was carried out using 50 and 100 packets from both DDoS and a legitimate traffic benchmark dataset to evaluate our proposed method. In the first experiment, the first 50 packets in the flow consisted of legitimate packets while the subsequent 50 were attack packets. The same applies to the second experiment (100 packets scenario). The normal packets collected from the front-end of the cloud are used to train the detector *a priori* before deployment, to detect any subsequent deviation from the pre-determined pattern

The observed IAT, X_i , of these packets was used during the network connection to calculate C_i as described in Eq. (2). The average and standard deviation were also computed for both normal and attack traffic flows, as shown in Table 5.1. For normal traffic, the first simulation involving 50 packets has an average IAT, λ_0= 0.00129 sec. (i.e., $\lambda_{0(50)}$ = 0.00129); while $\lambda_{0(100)}$ present a value of 0.00108. The standard deviations are $\sigma_{0(50)}$ = 0.00131 and $\sigma_{0(100)}$ = 0.00117, respectively. For the attack traffic, the values are as follows: $\lambda_{1(50)}$ = 0.00208, $\lambda_{1(100)}$ = 0.00187, $\sigma_{1(50)}$ = 0.00167, $\sigma_{1(100)}$ = 0.00144.

From our findings, it is observed that there is a clear distinction between the average and standard deviations of the packet IAT of the attack traffic flow and the legitimate traffic flow. To analyze our work, this research used a Change-Point analyzer [96] (-see Appendix B)

Table 5.1 IAT distribution in traffic flow

Packet No.	λ_0	λ_1	σ_0	σ_1
50	0.00129	0.00208	0.00131	0.00167
100	0.00108	0.00187	0.00117	0.00144

Normal Traffic behaviour

For the trace-driven experiments for 50 and 100 packets IAT in a traffic flow, a change-point detection using the CUSUM algorithm for normal traffic flow was simulated, and is thus presented in Figures 5.4 and 5.5.

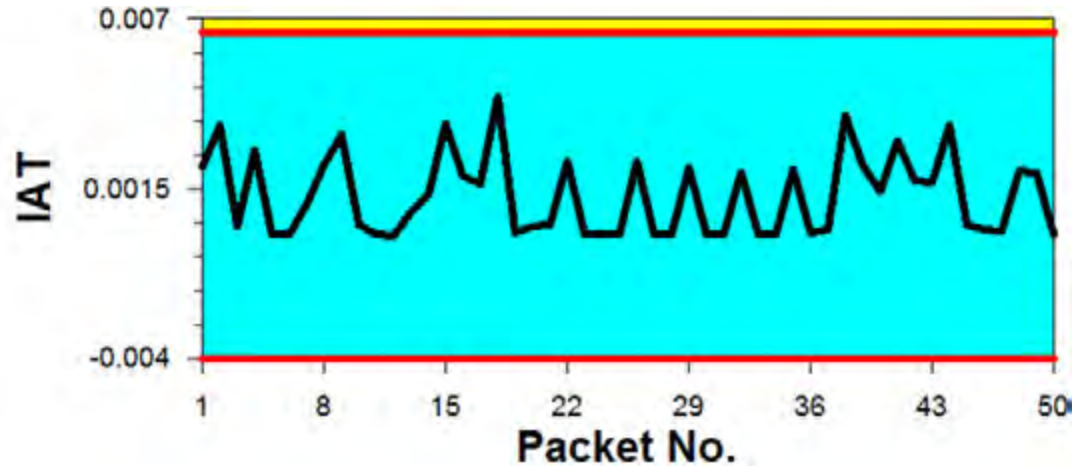

Figure 5.4 Plot of IAT against packet number.

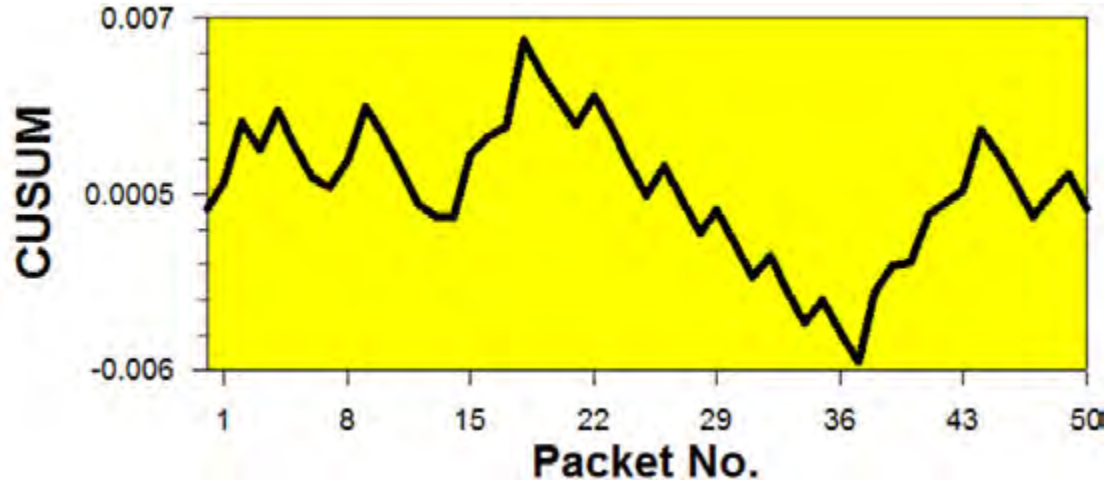

Figure 5.5 CUSUM statistics.

Figure 5.4 shows the plot of IAT in seconds against the packet number in a flow. Visual inspections of the IAT values show no major change. This is confirmed by the CUSUM plot shown in Figure 5.5. The CUSUM plot starts and ends on zero while the maximum to minimum CUSUM determines the limit values for the CUSUM plot, which ranges between 0.007 and -0.006.

Figures 5.6 and 5.7 show the graph for a 100-packet experiment in a traffic flow and the packet IAT, X_i presents a similar result (as shown in Figure 5.4 and 5.5). The CUSUM algorithm is determined by adding the difference between the current value and the average of the previous sum. Therefore, in this normal traffic flow, the IAT pattern follows a sequence and there was no significant change observed using CUSUM.

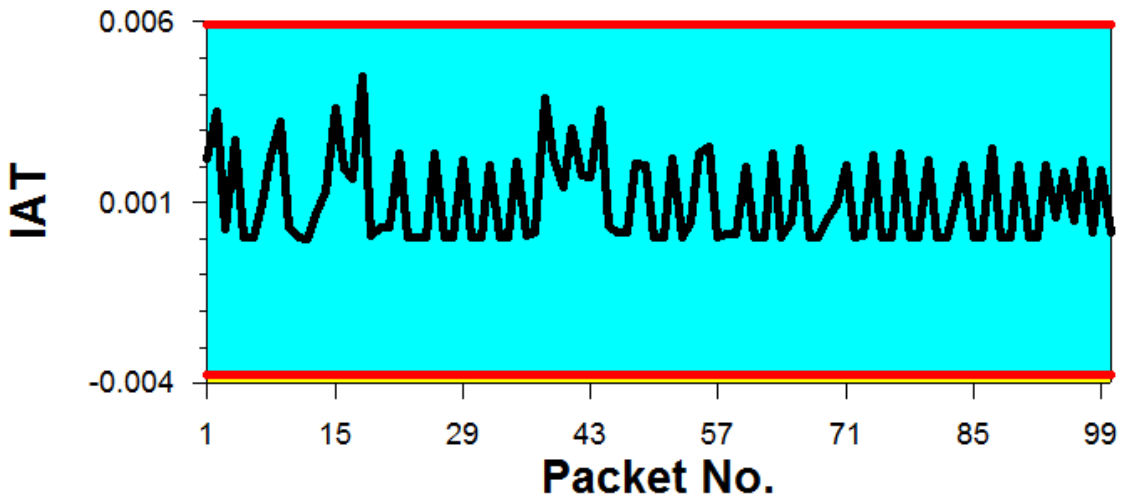

Figure 5.6 Graph of IAT against packet numbers during a normal traffic flow.

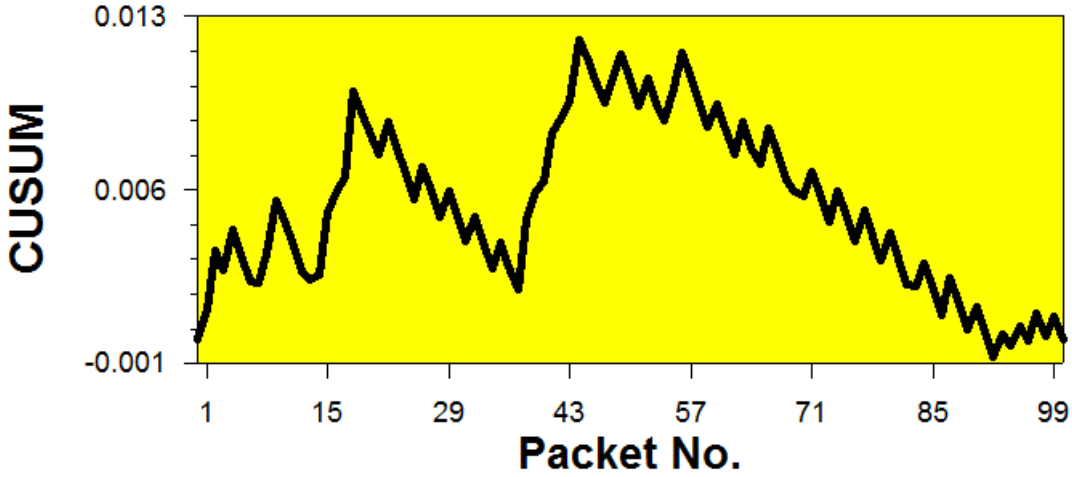

Figure 5.7 CUSUM statistics.

SYN Flooding Detection

For the SYN flooding detection experiments, both Auckland-VIII [34] and CAIDA DDoS datasets [23] were used. To measure the efficiency of our proposed method, this research appends attack traffic to a normal traffic flow to determine the point of change, using the CUSUM algorithm.

For 50 normal and 50 SYN flooding packets, Figures 5.8 and 5.9 show the plot of IAT against the packet number and the CUSUM test statistic used in detecting a change-point within the traffic flow.

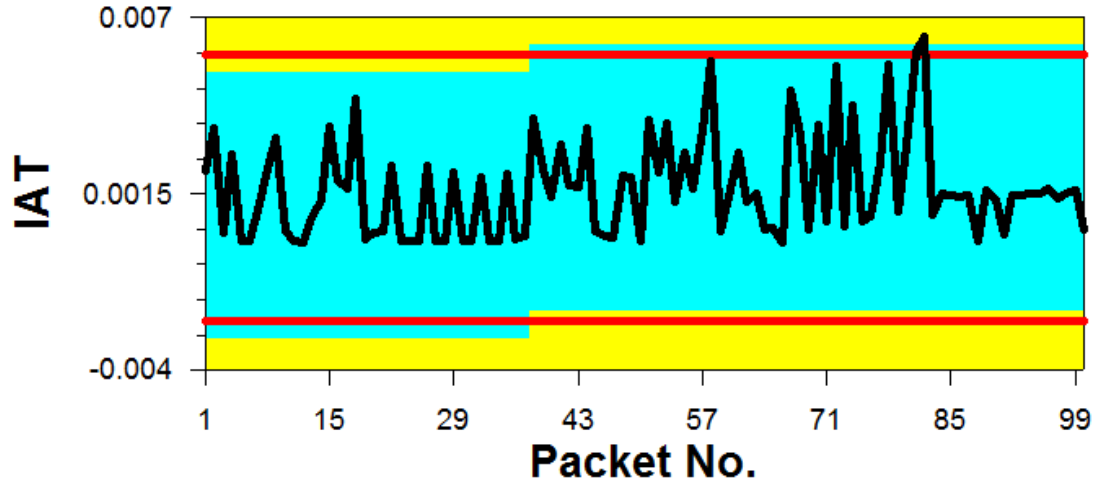

Figure 5.8 Auckland-VIII/CAIDA DDoS Plot of IAT against packet number.

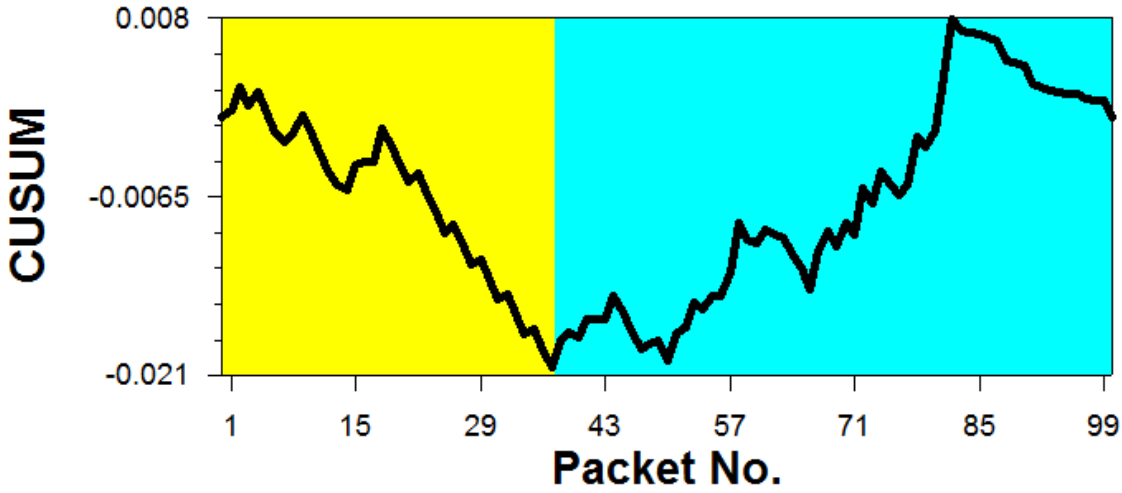

Figure 5.9 CUSUM statistics

From Figure 5.9 , it can be observed that our method detects an anomaly in the IAT on the 38th packet. This is deemed a false alarm as the DDoS packet starts from the 51st packet. This research also conducted a similar experiment for 100 normal and 100 SYN-flooding packets in a single flow (see Figures 5.10 and 5.11).

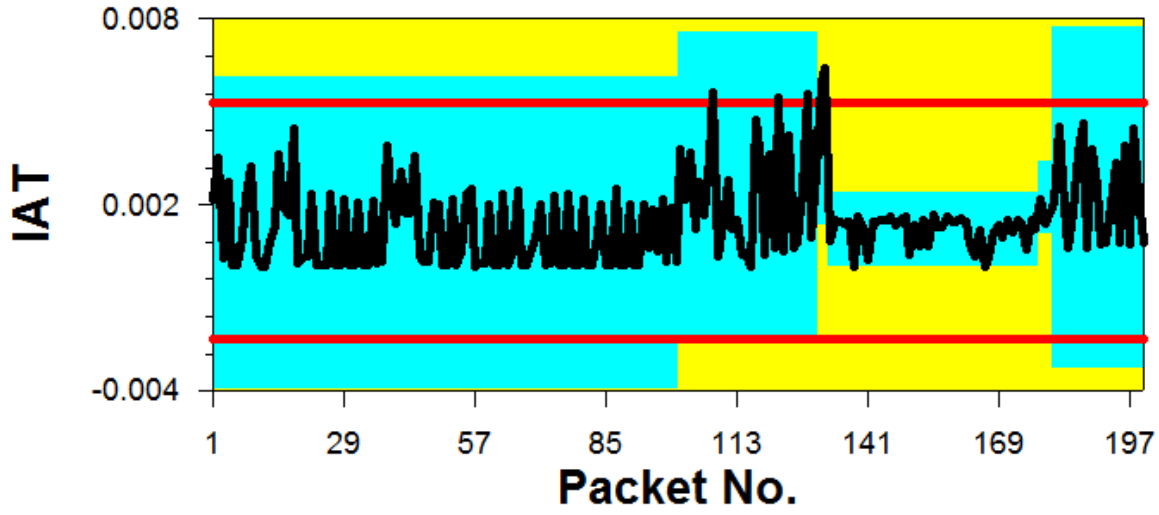

Figure 5.10 Auckland-VIII/CAIDA DDoS Plot of IAT against packet number.

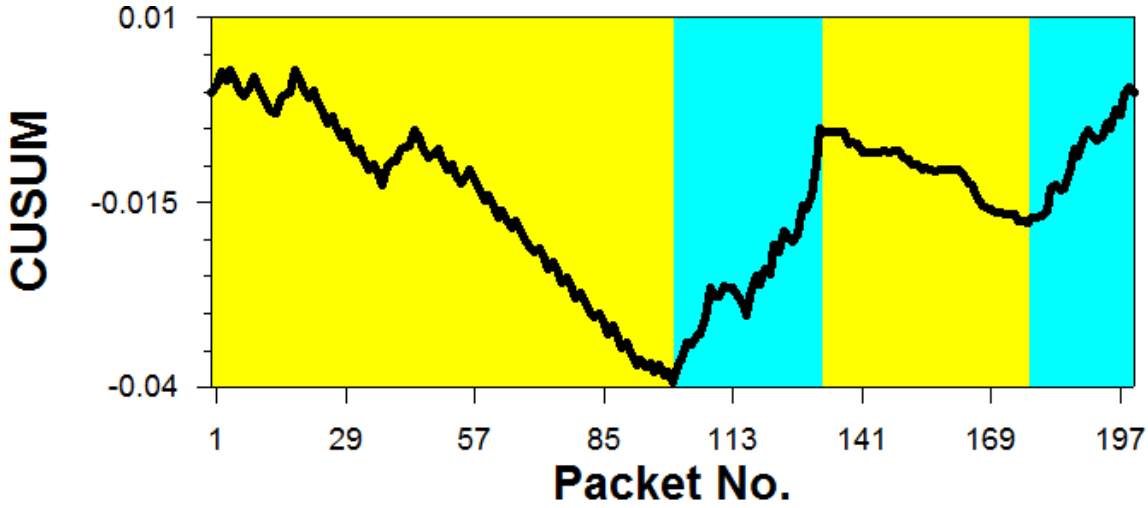

Figure 5.11 CUSUM statistics.

A change was detected on packet number 101, which is the starting point of our attack traffic. This is an improvement on the 50 packet experiment as it achieves a 100 % detection rate. This suggests that using the change-point detection attribute helps to achieves a better performance on a longer period data, by observing the pattern to detect a shift. This, therefore, shows that the CUSUM change-point detection method requires more packet samples to detect the occurrence of a true change during a DDoS attack in the network.

5.4 Discussion

To detect a DDoS flooding attack, it is important to use the minimum number of traffic features, in order to reduce the complexity and the associated overheads. Therefore, in this work, only packet IAT features were used.

To evaluate the performance of our proposed change-point detection, using the CUSUM algorithm, a publicly available datasets was used. When evaluating and testing the proposed method, the benchmark dataset used must be representative. The CAIDA DDoS attack dataset [23] presents one of the most recent datasets used in the literature for analysis. Our evaluation shows that 150 packets also present an optimum packet count for normal flow, in order to detect a change during a DDoS attack with perfect detection accuracy. 100 packets normal flow however have been adopted by our flow-based classifier (FSB), as depicted in Figure 5.1. When compared with the other methods proposed in the literature, it is difficult for an attacker to manipulate our selected detection feature and still circumvent detection. Table 5.2 presents our findings.

Table 5.2 Detection accuracy for different traffic flows

Packet No.	$P_m(\lambda_0, t)$	$P_m(\lambda_1, t)$	τ	Accuracy
100	50	50	38	76 %
200	100	100	101	100 %
300	150	150	151	100 %
400	200	200	200	99.5 %
500	250	250	249	99.2%

5.5 Summary

In this chapter, our proposed method uses both the flow-based classifier and the change-point detection, using the CUSUM algorithm to detect cloud DDoS flooding attacks, based on the packet IAT rate.

To demonstrate the utility of our proposed method, this thesis conducted a trace-driven experiment by deploying our approach to detect changes within the flow consisting of both normal and attack packets. Thereafter, this work evaluated our method using flooding attacks from the CAIDA benchmark dataset and normal traffic trace from Auckland-VIII. The findings suggested that optimal performance was achieved by CUSUM during 100- and 150-packet counts for normal flow to detect an equal amount of attack packets.

Chapter 6: Conclusion and Future work

As the final and concluding chapter of this thesis, this section presents the overall contributions of this study against the set objectives by presenting a chapter summary indicating the exact contribution. Thereafter, recommendations and suggestions for future work are discussed.

6.1 Key Contributions

This thesis entitled: "DDoS Defence for Service Availability in Cloud Computing" expounds the overall goal of this study. The availability of cloud services and resources ensures that cloud providers meet the recommended "five 9's" i.e., 99.999% up-time requirement (synonymous with only 5 minutes of downtime in a year), which can be hindered by a DDoS attack.

This thesis, therefore, focused on service availability in cloud computing, by presenting the foremost security challenge, DDoS attack, which affects the availability of services and resources offered by the cloud provider to the cloud user. This thesis has presented three methods in relation to the set objectives as the key contributions to mitigate against DDoS attacks in cloud computing:

- Contribution 1: This thesis proposes an IP spoofing detection technique that uses a host-based OS fingerprinting in passive and active stages to analyse and match the OS of incoming packets in the front-end of the cloud environment. Packet header features in both the active and passive stages are used to determine the true source of incoming packets by identifying their OS to detect a spoofed source IP. Additionally, the observed final time to live (TTL) value during active and passive stages of the OS fingerprinting is used to cater for false negatives during detection.
- Contribution 2: Due to the magnitude of traffic that needs to be processed during a DDoS attack defence in cloud computing, this thesis has presented a feature selection method called Ensemble-based Multi-filter Feature Selection (EMFFS) that pre-processes the data before classification. EMFFS combines the outputs for four filter feature selection methods: information gain (IG), gain ratio, chi-squared and reliefF, to select important features. This significantly reduces the feature set, thus, reducing the computational complexity, while maintaining or improving the classification accuracy using a decision tree classifier.

- Contribution 3: This thesis presents a change-point monitoring algorithm to detect the DDoS flooding attack against cloud services, by examining the packet inter-arrival time (IAT). Our method leverages on the fact that most DDoS attacks are automated and exhibit similar patterns. Therefore, when closely examined, they can be distinguished from normal traffic and tracked by using a cumulative sum algorithm (CUSUM).

In summary, this thesis has presented 6 chapters. Chapter 1 is the introductory chapter that gave a general overview of cloud computing, its services and a deployment model. Additionally, this chapter presented cloud security challenges with the emphasis on DDoS attacks, which affect the availability of cloud, and its defences. Chapter 1 also presents the research motivation, the objectives and the contributions. The literature review and the contributions of this thesis were presented in Chapters 2-5.

Chapter 2, the literature review chapter, presented a detailed review of the academic literature on DDoS attacks against cloud services, and the mitigation strategies published between January 2010 and December 2015. A taxonomy of the different types of cloud DDoS and their defences was also presented. From the review, it was observed that anomaly-based detection and access point deployment are the most popular defence techniques and deployment locations in the literature. Furthermore, Chapter 2 identified the research gaps in the current proposed DDoS defences in cloud computing.

In Chapter 3, the solution to a common DDoS attack attribute, the spoofing of IP addresses that frustrates easy traceback, was proposed. This Chapter proposed an IP spoofing detection technique that uses OS fingerprinting in both active and passive stages to analyse the interesting features on the TCP/IP header features to determine the connecting host's OS. This was achieved by monitoring the SYN and SYN +ACK control packets, during TCP connection, to verify the true source of an incoming packet at the front-end of the cloud environment during a DDoS attack. The study in Chapter 3 thereafter used the final TTL value to further determine a spoofed source in the event where the spoofed and the true source run similar OSs.

Chapter 4 presented an extended defence to the first layer of defence, OS fingerprint based IP Spoofing detection, as presented in Chapter 3. An ensemble-based multi-filter feature selection method was proposed in this Chapter, to cater for the challenges

of increase in the amount of data that need to be processed, experienced by the current intrusion detection system. EMFFS combines the output of one-third split of ranked important features by using information gain, gain ratio, chi-squared and reliefF. Our method selects 13 out of the 41 features of NSL-KDD benchmark dataset, and uses a decision tree classifier, J48, for classification. The evaluation of our proposed method shows a better performance in terms of classification accuracy and time to build the model, when compared with other proposed methods in the literature.

Chapter 5 proposed a change-point cloud DDoS detection method, using the CUSUM algorithm to mitigate against different forms of DDoS attack. This technique used both a flow based classifier and the CUSUM algorithm to detect a cloud DDoS flooding attack, based on the packet inter-arrival time. The demonstration of the efficiency of our proposed method was conducted by using a trace-driven experiment to detect traffic flow consisting of both normal and attack packets. The evaluation of our method using normal traffic trace from the Auckland-VIII benchmark dataset and the CAIDA DDoS dataset shows that optimal performance is achieved by CUSUM during 100- and 150-packet counts for normal traffic flow, in order to detect an equal amount of attack packets.

6.2 Future Work

When assessing the availability of cloud services and resources, apart from security, which has been considered in this thesis, application and infrastructural failure are the other factors that can be considered. Application and infrastructural failure of cloud components can either be physical, human, and/or operational, which can be caused by system failure, network failure, power outages, design error, or a software bug. To compensate for these failures, in this thesis, the following recommendations are considered as useful guidelines for future research possibilities.

- Research could be carried out as future work to extend our solution beyond security to cater for both hardware and software failures.
- As discussed in this thesis, the high availability of cloud resources is essential, as attack or failure of the infrastructure would be catastrophic to both the providers and the end-users. One mitigation strategy is virtual machine (VM) migration, where VMs are moved from one physical host to another, in order to improve the

performance and reliability. There are different approaches to VM migration, namely: cold migration, hot migration, and live migration. Cold migration involves shutting down the guest OS – before moving the VM to a predetermined host and restarting the system. Hot migration, on the other hand, only suspends the running guest OS – rather than shutting it down before it is transmitted and resumed at the predetermined target host. The latter has an advantage over the former; as the running applications in the guest OS are not restarted from scratch. Live migration guarantees a continuous service of the hosted applications, while allowing a VM and its running OS to be moved from one physical host to another. During live migration, a VM and its environment – comprising running task, OS, memory, vCPU and sometimes the disk – are moved seamlessly between two physical hosts. Other notable benefits of VM migration include: improved load balancing, transparent mobility, pro-active fault tolerance, and green computing.

- VM live migration can, however, be resource-intensive as it consumes a large amount of CPU cycles and network bandwidths. Therefore, recent implementations have introduced a shared storage (i.e., network attached storage) between the source and the target hosts. With a shared storage, disk storage does not need to be migrated, therefore, only the content of the memory pages that is not available in the shared storage device is transferred. This immensely reduces the transmission time and the downtime of applications running on the moving VM. During live migration, downtime is the amount of time the migrating VM halts to move from source to target host, while the total migration time refers to the total time from the commencement of the migration to the time when the VM is up and running on the target host. Downtime, migration time, and the number of dirty pages (data) migrated during VM live migration are some of the key performance metrics researchers must optimize, in an attempt to achieve high availability, load balancing, and resilience in a virtualized environment.

  As future work, the migration time and the downtime could be optimized, by introducing a smart pre-copy live migration approach that estimates the downtime after each iteration, to determine whether to proceed to the stop-and-copy stage, or not.

References

[1] W.Voorsluys, J. Broberg, and R. Buyya, "Introduction to cloud computing," *Cloud computing: Principles and paradigms*, pp.1-44, Feb. 2011.

[2] R. Buyya, C. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 4, pp. 599-616, Jun 2009.

[3] J. Huang, and D. Nicol, "Trust mechanisms for cloud computing," *Journal of Cloud Computing*, vol. 2, no. 1, pp. 1-4, Dec 2013.

[4] X. Xu, "From cloud computing to cloud manufacturing," *Robotics and Computer-integrated Manufacturing*, vol. 28, no. 1, pp. 75-86, Feb 2012.

[5] M. Almorsy, J. Grundy, and I. Müller, "An analysis of the cloud computing security problem," *in Proceedings of APSEC 2010 Cloud Workshop*, Nov 2010, pp. 1-6.

[6] W Tsai, X. Sun, and J. Balasooriya, "Service-Oriented Cloud Computing Architecture," *In IEEE Seventh International Conference on Information Technology: New Generations (ITNG),* April 2010*,* pp. 684-689.

[7] M. Alvarado, R. Agrawal, and Y. Baker, "Security mechanisms utilized in a secured cloud infrastructure," *in Proceedings of IEEE Southeastcon*, Apr 2013, pp. 1-5.

[8] V. Medina, and J. García, "A survey of migration mechanisms of virtual machines," *ACM Computing Survey (CSUR)*, vol. 46, no. 3, pp.1-33, Jan 2014.

[9] M. Mahjoub, A. Mdhaffar, R. Halima, and M. Jmaiel, "A comparative study of the current Cloud computing technologies and offers," *in First IEEE International Symposium on Network Cloud Computing and Applications (NCCA)*, Nov 2011, pp. 131-134.

[10] F. Machida, D. Kim and K. Trivedi, "Modelling and analysis of software rejuvenation in a server virtualized system with live VM migration," *Performance Evaluation*, vol. 70, no 3, pp.212-230, March 2013.

[11] J. Varia, “Best practices in architecting cloud applications in the AWS cloud,” *Cloud Computing: Principles and Paradigms*, pp. 459-490, 2011.

[12] K. Clark, M. Warnier, M Frances and T, Brazier, "The Future of Cloud-based Botnets?."
URL:https://www.researchgate.net/profile/Frances_Brazier/publication/220865587_Botclouds_-_The_Future_of_Cloud-based_Botnets/links/00463528e5a2f8da3a000000.pdf.

[13] J. Mirkovic, M. Robinson, P. Reiher and G. Oikonomou, “Distributed defence against DDOS attacks,” *University of Delaware CIS Department technical report CIS-TR-2005-02*, pp.1-12, 2005.

[14] J. Mirkovic and P. Reiher, “A taxonomy of DDoS attack and DDoS defense mechanisms,” *ACM SIGCOMM Computer Communication Review*, vol. 34, no.2, pp. 39-53, April 2004.

[15] L. Yang, T. Zhang, J. Song, J. Wang and P. Chen, “Defense of DDoS attack for cloud computing,” *in IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, vol. 2, May 2012, pp. 626-629 Vol.2.

[16] B. Gupta, R. Joshi, and M. Misra, “Distributed Denial of Service prevention techniques,” *International Journal of Computer and Electrical Engineering (IJCEE)*, vol. 2, no. 2, pp. 268-276, 2010.

[17] P. Mell and T. Grance, “The NIST definition of cloud computing,” *NIST Special Publication 800-145*, pp. 20-23, 2011.

[18] M. Khorshed, A. Ali and W. Wasimi, “A survey on gaps, threat-remediation challenges and some thoughts for proactive attack detection in cloud computing,” *Future Generation Computer Systems*, vol. 28, no. 6, pp.833-851, June 2012.

[19] W. Christof, D. Anandasivam, B. Blau, D. Borissov, D. menin, D. Michalk and J. Stober, “Cloud computing – a classification, business models, and research directions,” *Business & Information Systems Engineering*, vol. 1, no. 5, pp. 391-399, Oct 2009.

[20] N. Gonzalez, C. Miers, F. Redigolo, M. Naslund and M. Pourzandi, “A quantitative analysis of current security concerns and solutions for cloud computing,” *Journal of Cloud Computing*, vol. 1, no. 1, pp. 1-18, Dec 2012.

[21] J. Wayne, "Cloud hooks: Security and privacy issues in cloud computing," *in proceedings of IEEE 44th Hawaii International Conference on System Sciences (HICSS)*, Jan 2011, pp. 1-10.

[22] [Online] www.cloudcontrols.org/cloud-standard-information/cloud-definations/

[23] The CAIDA UCSD "DDoS Attack 2007" Dataset. [Online]: http://www.caida.org/data/passive/ ddos-20070804_dataset.xml.

[24] B. Wang, Y. Zheng, W. Lou and Y. Hou, "DDoS attack protection in the era of cloud computing and Software-Defined Networking," *Computer Networks*, vol. 81, 308-319, April 2015.

[25] F. Wong and C. Tan, "A survey of trends in massive DDoS attacks and cloud-based mitigations," *International Journal of Network Security & Its Applications (IJNSA)*, vol. 6, no. 3, pp. 57-71, May 2014.

[26] Y. Gu., Y.Fu., A. Prakash., Z.Lin and H.Yin, "OS-Sommelier: memory-only operating system fingerprinting in the cloud, "in *Proceedings of the Third ACM Symposium on Cloud Computing*, *ACM*, Oct 2012, pp. 1-13.

[27] M. Darwish, A. Ouda and L. Capretz, "Cloud-based DDoS attacks and defences," in IEEE International Conference on Information Society (I-Society), June 2013, pp. 67-71.

[28] J. Liebeskind, "Keeping organizational secrets: Protective institutional mechanisms and their costs," *Industrial and Corporate Change*, vol. 6, no. 3, pp. 623-663, Sept 1997.

[29] J. Choi, C. Choi, B. Ko, D. Choi and P. Kim, "Detecting web based DDoS attack using MapReduce operations in cloud computing environment," *Journal of Internet services and information security*, vol. 3, no. 3/4, pp. 28-37, Nov 2013.

[30] S. Zargar, J. Joshi and T. David, "A survey of defence mechanisms against distributed denial of service (DDoS) flooding attacks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 2046-2069, Jan 2013.

[31] The Truth about DDoS Attacks: Part 1 <http://www.carbon60.com/the-truth-about-ddos-attacks-part-1/>; 2013.

[32] R. Deshmukh and K. Devadkar, "Understanding DDoS Attack & its Effect in Cloud Environment," *Procedia Computer Science*, vol. 49, pp. 202-210, Dec 2015.

[33] A. Mishra, B. Gupta and R. Joshi, "A comparative study of distributed denial of service attacks, intrusion tolerance and mitigation techniques," *in Proceedings of IEEE European Intelligence and Security Informatics Conference (EISIC)*, Sept 2011, pp. 286-289.

[34] Waikato Internet Traffic Storage (WITS): Auckland VIII trace. [online]: <http://www.wand.net.nz/wits/auck/8/auckland_viii.php>

[35] B. Cha and J. Kim, "Study of multistage anomaly detection for secured cloud computing resources in future Internet," in *IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC)*, Dec 2011, pp. 1046-1050.

[36] M. Bhuyan, D. Bhattacharyya and J. Kalita, "An empirical evaluation of information metrics for low-rate and high-rate DDoS attack detection," *Pattern Recognition Letters*, vol. 51, pp. 1-7, Jan 2015.

[37] H. Beitollahi, and G. Deconinck, "Analyzing well-known countermeasures against distributed denial of service attacks," *Computer Communications*, vol. 35, no. 11, pp. 1312- 1332, Jan 2012.

[38] Y. Dantas, V. Nigam and I. Fonseca, "A Selective Defense for Application Layer DDoS Attacks," *in: Proceedings of IEEE Joint Intelligence and Security Informatics Conference (JISIC),* Sept 2014, pp. 75-82.

[39] B. Cha and J. Kim, "Study of multistage anomaly detection for secured cloud computing resources in future Internet," in: *Proceedings of IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC),* Dec 2011, pp. 1046-1050.

[40] X. Rui, M. Wen-Li and Z. Wen-Ling, "Defending against UDP flooding by negative selection algorithm based on eigenvalue sets," in: *Proceedings of IEEE fifth International Conference on Information Assurance and Security (IAS'09)*, Aug 2009, pp. 342-345, Vol. 2.

[41] J. Choi, C. Choi, B. Ko and P. Kim, "A method of DDoS attack detection using HTTP packet pattern and rule engine in cloud computing environment," *Soft Computing*, vol. 18, no. 9, pp. 1697-1703, Sept 2014.

[42] T. Karnwal, T. Sivakumar and G. Aghila, "A comber approach to protect cloud computing against XML DDoS and HTTP DDoS attack," in: *Proceedings of IEEE Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, March 2012, pp. 1-5.

[43] N. Gruschka and L. Iacono, "Vulnerable cloud: Soap message security validation revisited," in: *Proceedings of IEEE International Conference on Web Services (ICWS 2009)*, July 2009, pp. 625-631.

[44] S. Arukonda and S. Sinha, "The Innocent Perpetrators: Reflectors and Reflection Attacks," *Advances in Computer Science*, vol. 4, no. 1, pp. 94-98, 2015.

[45] A. Ali-Eldin, T. Johan and E. Erik, "An adaptive hybrid elasticity controller for cloud infrastructures," in: *IEEE Network Operations and Management Symposium (NOMS)*, April 2012, pp. 204-212.

[46] M. Sqalli, F. Al-Haidari and K. Salah, "EDoS-shield-a two-steps mitigation technique against EDoS attacks in cloud computing, in: *Proceedings of Fourth IEEE International Conference on Utility and Cloud Computing (UCC)*, Dec 2011, pp. 49-56.

[47] J. Idziorek, M. Tannian and D. Jacobson," Attribution of fraudulent resource consumption in the cloud," in *Proceedings of 5th IEEE International Conference on Cloud Computing (CLOUD)*, June 2012, pp. 99-106.

[48] [online] http://www.troyhunt.com/2013/01/what-is-loic-and-can-i-be-arrested-for.html

[49] [online] http://resources.infosecinstitute.com/dos-attacks-free-dos-attacking-tools/

[50][online]https://stormsecurity.wordpress.com/2009/03/03/application-layer-ddos-simulator/

[51][online]http://www.hackerschronicle.com/2014/03/davoset-most-powerfull-ddos-tool.html

[52] [online] http://security.radware.com/knowledge-center/DDoSPedia/Pyloris/

[53] M. Bhuyan, H. Kashyap, D. Bhattacharyya, J. Kalita, "Detecting distributed denial of service attacks: methods, tools and future directions," *The Computer Journal*, pp. 6-20, March 2013.

[54] N. Iyengar, G. Ganapathy, P. Kumar and A. Abraham, "A multilevel thrust-filtration defending mechanism against DDoS attacks in cloud computing environment," *International Journal of Grid and Utility Computing*, vol. 5, no. 4, pp. 236-248, Jan 2014.

[55] A. Bakshi and B. Yogesh, "Securing cloud from DDoS attacks using intrusion-detection system in virtual machine," in *Proceedings of Second IEEE International*

Conference on Communication Software and Networks (ICCSN'10), Feb 2010, pp. 260-264.

[56] A. Lonea, D. Popescu, O. Prostean and H. Tianfield, "Evaluation of Experiments on Detecting Distributed Denial of Service (DDoS) Attacks in Eucalyptus Private Cloud," *Soft Computing Applications*, pp. 367-379, 2013.

[57] T. Karnwal, S. Thandapanii and A. Gnanasekaran, "A filter tree approach to protect cloud computing against XML DDoS and HTTP DDoS attack," *In Intelligent Informatics*, pp. 459-469, 2013.

[58] I. Gul and M. Hussain, "Distributed cloud intrusion detection model," *International Journal of Advanced Science and Technology*, vol. 34, pp. 71-82, Sept 2011.

[59] C. Lo, C. Huang and J. Ku, "A cooperative intrusion detection system framework for cloud computing networks," in *Proceedings of 39th IEEE international conference on Parallel processing workshops (ICPPW)*, Sept 2010, pp. 280-284.

[60] S. Gupta and P. Kumar, "VM profile based optimized network attack pattern detection scheme for DDoS attacks in cloud," in *Proceedings of International Symposium of Security in Computing and Communications (SSCC 2013)*, 2013, pp. 255-261.

[61] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel and M. Rajarajan, "A survey of intrusion detection techniques in cloud," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 42-57, Jan 2013.

[62] P. Shamsolmoali and M. Zareapoor, "Statistical-based filtering system against DDOS attacks in cloud computing," in *Proceedings of International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Sept 2014, pp. 1234-1239.

[63] M. Zakarya, "DDoS Verification and Attack Packet Dropping Algorithm in Cloud Computing," *World Applied Sciences Journal*, vol. 23, no.11, 1418-1424, 2013.

[64] T. Vissers, T. Somasundaram, L. Pieters, K. Govindarajan, P. Hellinckx, "DDoS defense system for web services in a cloud environment," *Future Generation Computer Systems*, vol. 37, pp.37-45, July 2014.

[65] S. Alqahtani, R. Gamble, "DDoS Attacks in Service Clouds," in *Proceedings of 48th IEEE International Conference on System Sciences (HICSS)*, Jan 2015, pp. 5331-5340.

[66] A. Girma, M. Garuba, J. Li and C, Liu, "Analysis of DDoS Attacks and an Introduction of a Hybrid Statistical Model to Detect DDoS Attacks on Cloud Computing Environment," in *Proceedings of 12th IEEE International Conference on Information Technology-New Generations (ITNG)*, April 2015, pp. 212-217.

[67] V. Huang, R. Huang and M. Chiang, "A DDoS Mitigation System with Multistage Detection and Text-Based Turing Testing in Cloud Computing," in *Proceedings of 27th IEEE 27th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, March 2013, pp. 655-662.

[68] A. Lonea, D. Popescu and H. Tianfield, "*Detecting DDoS attacks in cloud computing environment*," International Journal of Computers Communications & Control, vol. 8, no 1, pp. 70-78, 2013.

[69] H. Kwon, T. Kim, S. Yu and H. Kim, "Self-similarity based lightweight intrusion detection method for cloud computing," in *Proceedings of 3rd International Conference on Intelligent Information and Database Systems (ACIIDS)*, Nov 2011, pp. 353-362.

[70] M. Ismail, A. Aborujilah, S. Musa and A. Shahzad, "Detecting flooding based DoS attack in cloud computing environment using covariance matrix approach," in *Proceedings of the 7th ACM International Conference on Ubiquitous Information Management and Communication (ICUIMC '13)*, Jan 2013, pp. 36.

[71] W. Dou, Q. Chen and J. Chen, "A confidence-based filtering method for DDoS attack defense in cloud environment," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1838-1850, Sept 2013.

[72] P. Negi, A. Mishra and B. Gupta, "Enhanced CBF Packet Filtering Method to Detect DDoS Attack in Cloud Computing Environment," *International Journal of Computer Science*, vol. 2, no. 4, pp. 142-146, April 2013.

[73] R. Michelin, A. Zorzo and C. De Rose, "Mitigating DoS to authenticate cloud REST APIs," in *Proceedings of 9th IEEE International Conference for Internet Technology and Secured Transactions (ICITST)*, Dec 2014, pp. 106-111.

[74] H. Bedi, and S. Shiva, "Securing cloud infrastructure against co-resident dos attacks using game theoretic defense mechanisms," in *Proceedings of ACM International Conference on Advances in Computing, Communications and Informatics (ICACCI '12)*, Aug 2012, pp. 463-469.

[75] N. Jeyanthi and N. Iyengar, "Escape-on-Sight: An Efficient and Scalable Mechanism for Escaping DDoS Attacks in Cloud Computing

Environment," *Cybernetics and Information Technologies*, vol. 13, no 1, pp. 46-60, March 2013.

[76] D. Krishnan and M. Chatterjee, "An adaptive distributed intrusion detection system for cloud computing framework," in *Proceedings of International Conference of Recent Trends in Computer Networks and Distributed Systems Security (SNDS)*, Oct 2012, pp. 466-473.

[77] S. Gupta, P. Kumar and A. Abraham, "A profile-based network intrusion detection and prevention system for securing cloud environment," *International Journal of Distributed Sensor Networks*, pp.1-12, March 2013.

[78] A. Chonka, J. Abawajy, "Detecting and mitigating HX-DoS attacks against cloud web services," in *Proceedings of 15th IEEE International Conference on Network-Based Information Systems (NBiS)*, Sept 2012, pp. 429-434.

[79] C.Modi, D. Patel, A. Patel and R. Muttukrishnan, "Bayesian Classifier and Snort-based network intrusion detection system in cloud computing," in *Proceedings of 3rd International IEEE Conference on Computing Communication & Networking Technologies (ICCCNT)*, July 2012, pp. 1-7.

[80] S. Teng, C. Zheng, H. Zhu, D. Liu, W. Zhang, "A Cooperative Intrusion Detection Model for Cloud Computing Networks," *International Journal of Security and Its Applications*, vol. 8, no. 3, pp.107-118, 2014.

[81] S. Yu, Y. Tian, S. Guo and D. Wu, "Can we beat DDoS attacks in clouds?" *IEEE Transaction on Parallel and Distributed Systems,* vol. 25, no. 9, pp. 2245-2254, Sept 2014.

[82] F. Guenane, M. Nogueira and G. Pujolle, "Reducing DDoS attacks impact using a hybrid cloud-based firewalling architecture," in *Proceedings of IEEE Global Information Infrastructure and Networking Symposium (GIIS)*, Sept 2014, pp. 1-6.

[83] N. Iyengar and G. Ganapathy, "Trilateral Trust-Based Defense Mechanism against DDoS Attacks in Cloud Computing Environment," *Cybernetics and Information Technologies*, vol. 15, no. 2, pp.119-140, Sep 2014.

[84] N. Jeyanthi, N. Iyengar, P. Kumar and A. Kannammal, "An enhanced-entropy approach to detect and prevent DDoS in cloud environment*," International Journal of Communication Networks and Information Security (IJCNIS)*, vol. 5, no 2, pp.119-140, Aug 2013.

[85] N. Jeyanthi, U. Barde, M. Sravani, V. Tiwari and N. Iyengar, "Detection of distributed denial of service attacks in cloud computing by identifying spoofed IP,"

International Journal of Communication Networks and Distributed Systems, vol. 11, no. 3, pp.262-279, Jan 2013.

[86] S. Chapade, K. Pandey and D. Bhade, "Securing cloud servers against flooding-based DDoS attacks," in *Proceedings of IEEE International Conference on Communication Systems and Network Technologies (CSNT)*, April 2013, pp. 524-528.

[87] H. Liu, "A new form of DOS attack in a cloud and its avoidance mechanism," in: *Proceedings of the 2010 ACM workshop on Cloud computing security workshop*, Oct 2010, pp. 65-76.

[88] R. Aishwarya and S. Malliga, "Intrusion detection system – An efficient way to thwart against Dos/DDos attack in the cloud environment," in *Proceedings of IEEE International Conference on Recent Trends in Information Technology (ICRTIT)*, April 2014, pp. 1-6

[89] B. Joshi and B.K Joshi, "Securing cloud computing environment against DDoS attacks," in *Proceedings of IEEE International Conference on Computer Communication and Informatics (ICCCI)*, Jan 2012, pp. 1-5.

[90] O.Osanaiye, "IP spoofing detection for preventing DDoS attack in Cloud Computing," in *Proceedings of 18th IEEE International Conference on Intelligence in Next Generation Networks (ICIN)*, Feb 2015, pp. 139-141.

[91] D. Zissis and D. Lekkas, "Addressing cloud computing security issues," *Future Generation Computer Systems*, vol. 28, no. 3, pp. 583-592, March 2012.

[92] V. Chandola, B. Arindam and K. Vipin, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, pp. 1-58, July 2009.

[93] M. Tavallaee, E. Bagheri, W. Lu and A. Ghorbani, "A detailed analysis of the KDD CUP 99 dataset," in *Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA 2009)*, 2009, pp.1-6.

[94] M. Bhuyan, D. Bhattacharyya and J. Kalita, "Network anomaly detection: methods, systems and tools," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp.303-336, Jan 2014.

[95] R.M. Swathi, "DDoS Attacks on the Rise: Increased 132% in Q2 2015, HTTPS Most Targeted," online: Available from: http://dazeinfo.com/2015/08/28/Internet-security-ddos-attacks-china-australia-us-uk-akamai/

[96] W. Taylor, Change-Point Analyzer 2.0 shareware program, Taylor Enterprises, Libertyville, Illinois. 2000, [online]: http://www.variation.com/cpa.

[97] T. Ehrenkranz and J. Li "On the state of IP spoofing defense," *ACM Transactions on Internet Technology (TOIT)*, vol. 9, no 2, pp. 6:1-29, May 2009.

[98] G. Yao, B, Jun and P. Xiao, "VASE: Filtering IP spoofing traffic with agility." *Computer Networks*, vol. 57, no. 1, pp. 243-257, Jan 2013.

[99] A. Chonka, Y. Xiang, W. Zhou and A. Bonti, "Cloud security defence to protect cloud computing against HTTP-DoS and XML-DoS attacks," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp.1097-1107, July 2011.

[100] F. Palmieri, U. Fiore and A. Castiglione, "A distributed approach to network anomaly detection based on independent component analysis," *Concurrency and Computation: Practice and Experience*, vol. 26, no. 5, pp.1113-1129, April 2014.

[101] O.Osanaiye and M. Dlodlo, "TCP/IP header classification for detecting spoofed DDoS attacks in Cloud environment," *in Proceedings of 16th IEEE International Conference on Computer as a Tool (EUROCON 2015)*, Sept 2015, pp. 1-6.

[102] M. Ficco and M. Rak, "Stealthy denial of service strategy in cloud computing," *IEEE Transactions on Cloud Computing*, vol. 3, no. 1, pp. 80-94, Jan 2015.

[103] J. Idziorek, M Tannian and D. Jacobson, "The Insecurity of Cloud-Utility Models," *IEEE IT Professional*, vol, 15 no. 2, pp.22-27, March 2013.

[104] F. Palmieri, S. Ricciardi, U. Fiore and M. Ficco and A. Castiglione, "Energy-oriented denial of service attacks: an emerging menace for large cloud infrastructures," *Journal of Supercomputing*, vol. 71, no. 5, pp. 1620-1641, May 2015.

[105] V. Prokhorenko, K Choo and K. Ashman, "Web-Application Protection Techniques: A Taxonomy," *Journal of Network and Computer Applications*, vol. 31, no. 60, pp. 95-112, Jan 2016.

[106] A. Merlo, M. Migliardi, N. Gobbo, F. Palmieri and A. Castiglione, "A denial of service attack to UMTS networks using SIM-less devices," *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 3, pp. 280-291, May 2014.

[107] M. Ficco, S. Venticinque and B. Di Martino, "An advanced intrusion detection framework for cloud computing," *Computer Systems Science and Engineering*, vol. 28, no. 6, pp. 401-411, 2013.

[108] M. Ficco, "Security event-correlation approach for cloud computing," *International Journal of High Performance Computing and Networking*, vol. 7, no. 3, pp.173-185. Jan 2013.

[109] F. Palmieri, M. Ficco and A. Castiglione, "Adaptive Stealth Energy-related DoS Attacks Against Cloud Data Centers," in *Proceedings of 8th IEEE International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, July 2014, pp. 265-272.

[110] M. Ficco and F. Palmieri, "Introducing Fraudulent Energy Consumption in Cloud Infrastructures: A New Generation of Denial-of-Service Attacks," *IEEE System Journal*, no. 99, pp.1-11, April 2015.

[111] Q. Jia, H. Wang, D. Fleck, F. Li, A. Stavrou and W. Powell, "Catch Me if You Can: A Cloud-Enabled DDoS Defense," in *Proceedings of 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, June 2014, pp. 264-275.

[112] A. Marnerides, P. Spachos, P. Chatzimisios and A. Mauthe, "Malware detection in the cloud under Ensemble Empirical Mode Decomposition," in: *Proceedings of IEEE International Conference on Computing, Networking and Communications (ICNC) and Information Security Symposium*, Feb 2015, pp. 82-88.

[113] Q. Yan, R. Yu, 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 Communications Survey & Tutorials*, vol. 18, no. 1, pp. 602-622, Jan 2016.

[114] Z. Anwar and A. Malik, "Can a DDoS Attack Meltdown My Data Center? A Simulation Study and Defense Strategies, *IEEE Communications Letters*, vol. 18, no. 7, pp. 1175-1178, July 2016.

[115] B. Wang, Y. Zheng, W. Lou and Y. Hou, "DDoS attack protection in the era of cloud computing and Software-Defined Networking," in *Proceedings of 22nd IEEE International Conference on Network Protocols (ICNP)*, April 2014, pp. 624-629.

[116] Z. Chen, G. Xu, V. Mahalingam, L. Ge, J. Nguyen, W. Yu and C. Lu, "A Cloud Computing Based Network Monitoring and Threat Detection System for Critical

Infrastructures," Big *Data Research*, Nov 2015, http://dx.doi.org/10.1016/j.bdr.2015.11.002.

[117] V. Varadharajan and U. Tupakula, "Counteracting security attacks in virtual machines in the cloud using property-based attestation," *Journal of Network and Computer Applications*, vol. 40, pp. 31-45, April 2014.

[118] S.Yu, Guo, S. and I. Stojmenovic, "Fool Me If You Can: Mimicking Attacks and Anti-Attacks in Cyberspace." *Computers IEEE Transaction*, vol. 64, no. 1, pp. 139-151, Jan 2015.

[119] J. Cheng, H. Wang and K.G. Shin, "Hop-count filtering: an effective defense against spoofed DDoS traffic," In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, Oct 2003, pp.30-41.

[120] The Swiss Education Research Network. Default TTL values in TCP/IP. Online: http://secfr.nerim.net/docs/fingerprint/en/ttl_default.html (Date accessed: March 3, 2014)

[121] H. Lee, M. Kwon, G. Hasker and A. Perrig, "BASE: An incrementally deployable mechanism for viable IP spoofing prevention", In *Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security,* March 2007, pp. 20-31.

[122] T. Cordeiro, D. Damalio, N. Pereira, P Endo, A. Palhares, G. Gonçalves, D. Sadok, J. Kelner, B. Melander, V. Souza and J, Mångs, "Open source cloud computing platforms*."* in IEEE *9th International Conference on Grid and Cooperative Computing (GCC)* Nov*,* 2010*,* pp. 366-371.

[123] H. Wang, D. Zhang and K. G. Shin, "Detecting SYN flooding attacks," in *Proceedings of INFOCOM 21st Annual Joint Conference of the IEEE Computer and Communications Societies,* vol. 3, June 2002, pp. 1530-1539, Vol 2.

[124] B. KrishnaKumar, P. K. Kumar and R. Sukanesh, "Hop count based packet processing approach to counter DDoS attacks," in *IEEE International Conference on Recent Trends in Information, Telecommunication and Computing (ITC),* March 2010, pp. 271-27.

[125] Cisco.com "The Internet Protocol Journal", *A quarterly Technical Publication for Internet and Intranet Professionals,* 7(4), pp. 1-40.

[126] B. Zhang, T. Zou, Y. Wang and B. Zhang, "Remote Operation System Detection Base on Machine Learning," in *IEEE International Conference on Frontier of Computer Science and Technology, FCST'09*, Dec 2009, pp. 539-542.

[127] Desktop Operating System Market Share. URL: http://www.netmarketshare.com/operating-system-market-share.aspx (accessed 14/11/2014)

[128] S. Shiaeles and M. Papadaki, "FHSD: An Improved IP Spoof Detection Method for Web DDoS Attack," *The Computer Journal*, vol. 58, no. 4, pp. 892-903, April 2015.

[129] P. Endo, G. Gonçalves, J. Kelner and D. Sadok, "A survey on open source cloud computing solutions," in *Brazilian Symposium on Computer Networks and Distributed Systems*, May 2010, pp.3-16.

[130] O. Osanaiye, K. Choo and M. Dlodlo, "Distributed Denial of Service (DDoS) Resilience in Cloud: Review and Conceptual Cloud DDoS Mitigation Framework," *Journal of Network and Computer Applications*, vol. 67, pp. 147-165, May 2016.

[131] L. Krämer, J. Krupp, D. Makita, T Nishizoe, T. Koide, K. Yoshioka and C. Rossow, "AmpPot: Monitoring and Defending Against Amplification DDoS Attacks," in *Proceedings of 18th International Symposium on Research in Attacks Intrusion and Defenses (RAID)*, Nov 2015, pp. 615-636.

[132] Y. Li, J. Xia, S. Zhang, J. Yan, X. Ai and K. Dai, "An efficient intrusion detection system based on support vector machines and gradually feature removal method," *Expert Systems with Applications*, vol. 39, no. 1, pp. 424-430, Jan 2012.

[133] P. Bermejo, L. de la Ossa, J. Gámez and J. Puerta, "Fast wrapper feature subset selection in high-dimensional datasets by means of filter re-ranking," Knowledge-*based Systems*, vol. 25, no. 1, pp. 35-44, Feb 2012.

[134] Y. Chen, Y. Li, X. Cheng, L. Guo, "Survey and taxonomy of feature selection algorithms in intrusion detection systems," in *Proceedings of the 2nd SKLOIS Conference Information Security and Cryptology (INSCRYPT)*, Nov 2006, pp. 153-167,

[135] W. Wang, Y. He, J. Liu and S. Gombault, "Constructing important features from massive network traffic for lightweight intrusion detection," *IET Information Security*, vol. 9, no. 6, pp. 374-379, April 2015.

[136] L. Koc, T. Mazzuchi and S. Sarkani, "A network intrusion detection system based on a Hidden Naïve Bayes multiclass classifier," *Expert Systems Applications*, vol. 39, no. 18, pp. 13492-13500, Dec 2012.

[137] W. Wang and S. Gombault, "Efficient detection of DDoS attacks with important attributes," in *Proceedings of the 3rd International conference on Risks and Security of Internet and Systems (CRiSIS'08)*, Oct 2008, pp. 61-67.

[138] V. Bolon-Canedo, N. Sanchez-Marono and A. Alonso-Betanzos, "Feature selection and classification in multiple class datasets: An application to KDD Cup 99 dataset," *Expert Systems Applications*, vol. 38, no 5, pp. 5947-5957, May 2011.

[139] Z. Baig, S. Sait and A. Shaheen, "GMDH-based networks for intelligent intrusion detection," *Engineering Application of Artificial Intelligence*, vol. 26, no 7, pp. 1731-1740, Aug 2013.

[140] S. Lin, K. Ying, C. Lee and Z. Lee, "An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection," *Applied Soft Computing*, vol. 12, no 10, pp. 3285-3290, Oct 2012.

[141] S. Sindhu, S. Geetha and A. Kannan, "Decision tree based lightweight intrusion detection using a wrapper approach," *Expert Systems Applications*, vol. 39, no. 1, pp. 129-141, Jan 2012.

[142] S. Bhattacharya and S. Selvakumar, "Multi-Measure Multi-Weight Ranking Approach for the Identification of the Network Features for the Detection of DoS and Probe Attacks," *The Computer Journal*, pp. 1-21, Oct 2015.

[143] Y. Li, J. Xia, S. Zhang, J. Yan, X. Ai, K. Dai, "An efficient intrusion detection system based on support vector machines and gradually feature removal method," *Expert System Applications*, vol. 39, no 1, pp. 424-430, Jan 2012.

[144] F. Zhang and D. Wang, "An effective feature selection approach for network intrusion detection," in *Proceedings of the 8th IEEE International Conference on Networking, Architecture and Storage (NAS)*, July 2013, pp. 307-311.

[145] A. Olusola, A. Oladele and D. Abosede, "Analysis of KDD'99 Intrusion-detection dataset for selection of relevance features," in *Proceedings of the World Congress on Engineering and Computer Science*, Oct 2010, pp.1-7.

[146] N. Sengupta, J. Sen, J. Sil and M. Saha, "Designing of online intrusion detection system using rough set theory and Q-learning algorithm," *Neurocomputing*, vol. 111, pp. 161-168, July 2013.

[147] A. Tesfahun and D. Bhaskari, "Intrusion Detection using Random Forests Classifier with SMOTE and Feature Reduction," in *Proceedings of the International Conference on Cloud & Ubiquitous Computing & Emerging Technologies (CUBE)*, Nov 2013, pp. 127-132

[148] B. Agarwal, N. Mittal, "Optimal feature selection for sentiment analysis," in *Proceedings of 14th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing)*, March 2013, pp. 13-24.

[149] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, Aug 2003, pp. 856-863.

[150] H.Ibrahim, S. Badr and M. Shaheen, "Adaptive Layered Approach using Machine Learning Techniques with Gain Ratio for Intrusion Detection Systems," *International Journal of Computer Application*, vol. 56, no. 7, pp. 10-16, Oct 2012.

[151] L. Devi, P. Subathra, P. Kumar, "Tweet Sentiment Classification Using an Ensemble of Machine Learning Supervised Classifiers Employing Statistical Feature Selection Methods," in *Proceedings of the Fifth International Conference on Fuzzy and Neuro Computing (FANCCO-2015)*, 2015, pp. 1-13.

[152] N. Nissim, R. Moskovitch, L. Rokach and Y Elovici, "Detecting unknown computer worm activity via support vector machines and active learning," *Pattern Analysis and Application*, vol.15, no. 4, pp. 459-475, Nov. 2012.

[153] M. Moradkhani, A. Amiri, M. Javaherian and H Safari, "A hybrid algorithm for feature subset selection in high-dimensional datasets using FICA and IWSSr algorithm," *Applied Soft Computing*, vol. 35, pp. 123-135, Oct 2015.

[154] N. Sánchez-Maroño, A. Alonso-Betanzos and M. Tombilla-Sanromán, "Filter methods for feature selection – a comparative study," in *Proceedings of the Eighth International Conference on Intelligent Data Engineering and Automated Learning (IDEAL 2007)*, Dec 2007, pp. 178-187.

[155] J. Gehrke, V. Ganti, R. Ramakrishnan and W. Loh, "BOAT—optimistic decision tree construction," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, June 1999, pp. 169-180.

[156] C. Xiang, P. Yong and L. Meng, "Design of multiple-level hybrid classifier for intrusion detection system using Bayesian clustering and decision trees," *Pattern Recognition Letters*, vol. 29, no. 7, pp. 918-924, May 2008.

[157] T. Bujlow, T. Riaz and J. Pedersen, "A method for classification of network traffic based on C5. 0 Machine Learning Algorithm," in *Proceedings of the IEEE International Conference on Computing, Networking and Communications (ICNC)*, Jan 2012, pp. 237-241.

[158] Data mining Software in Java. http://www.cs.waikato.ac.nz/ml/weka/ (Last accessed 7 February 2016)

[159] S. Rastegari, P. Hingston and C. Lam, "Evolving statistical rulesets for network intrusion detection," *Applied Soft Computing*, vol. 33, pp. 348-359, Aug 2015.

[160] J. Yu, H. Kang, D. Park, H. Bang and D Kang, "An in-depth analysis on traffic flooding attacks detection and system using data mining techniques," *Journal of System Architecture*, vol. 59, no. 10, pp. 1005-1012, Nov 2013.

[161] H. Eid, A. Hassanien, T. Kim and S. Banerjee, "Linear correlation-based feature selection for network intrusion detection model", in *Proceedings of the 1st International Conference on Advances in Security of Information and Communication Networks (SecNet)*, 2013, pp. 240-248.

[162] G. Geng, N. Li, S. Gong, "Feature selection Method for Network Intrusion Based on Fast Attribute Reduction of Rough Set", in *the Proceedings of International Conference on Industrial Control and Electronics Engineering (ICICEE)*, Aug 2012, pp. 530-534.

[163] L. Zhao, L. Chen, R. Ranjan, KKR. Choo and J. He, "Geographical information system parallelization for spatial big data processing: a review," *Cluster Computing*, vol. 19, no. 1, Jan 2016.

[164] D. Quick and K.K.R. Choo, "Impacts of Increasing Volume of Digital Forensic Data: A Survey and Future Research Challenges," *Digital Investigation*, vol. 11, no. 4, pp. 273–294, Dec 2014.

[165] C. Esposito, M. Ficco and F. Palmieri and A. Castiglione, "Interconnecting federated clouds by using publish-subscribe service," *Cluster Computing*, vol. 16, no. 4, pp. 887-903, Dec 2013.

[166] S. Shamshirband, N. Anuar, M. Kiah, V. Rohani, D. Petković, S. Misra and A. Khan, "Co-FAIS: Cooperative fuzzy artificial immune system for detecting intrusion in wireless sensor networks," *Journal for Network and Computer Applications*, vol. 42, pp. 102-117, June 2014.

[167] S. Shamshirband, A. Patel, NB. Anuar, M.L.M. Kiah and A. Abraham, "Cooperative game theoretic approach using fuzzy Q-learning for detecting and preventing intrusions in wireless sensor networks," *Engineering Applications of Artificial Intelligence*, vol. 32, pp. 228-241, June 2014.

[168] A. Khan, M. Kiah, S. Madani, M. Ali and S. Shamshirband, "Incremental proxy re-encryption scheme for mobile cloud computing environment," *The Journal of Supercomputing*, vol. 68, no. 2, pp. 624-651, May 2014.

[169] Z. Xu, Y. Liu, N. Yen, L. Mei, X. Luo, X. Wei and C. Hu, "Crowd-sourcing based description of urban emergency events using social media big data," IEEE Transactions on Cloud Computing, 2016, DOI: 0.1109/TCC.2016.2517638.

[170] Z. Xu, H. Zhang, V. Sugumaran, KKR. Choo, L. Mei, Y. Zhu, "Participatory sensing-based semantic and spatial analysis of urban emergency events using mobile social media," *EURASIP Journal on Wireless Communication and Networking*, vol. 1, pp.1-9, Dec 2016.

[171] R. Choo, "Cloud computing: challenges and future directions," *Trends and Issues in Crime and Justice*, vol. 400, pp. 1-6, Oct 2010.

[172] B. Martini and R. Choo, "Cloud storage forensics: owncloud as a case study," *Digital Investigation*, vol. 10, no. 4, pp. 287-99, Dec 2013.

[173] J. Arshad, P. Townend and J. Xu, "A novel intrusion severity analysis approach for Clouds," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 416-428, Jan 2013.

[174] Z. Zhou, X. Chen, J. Wang and X. Li, "A Distributed Detection Scheme Based on Adaptive CUSUM and Weighted CAT Against DDoS Attacks*,"* in*: Proceedings of the 3rd International Conference on Multimedia Technology (ICMT 2013)*, 2013, pp. 97-105.

[175] R. Choo, "The cyber threat landscape: Challenges and future research directions," *Computers and Security*, vol. 30, no. 8, pp. 719-731, Dec 2011.

[176] V. Gulisano, M. Callau-Zori, Z. Fu, R. Jiménez-Peris, M. Papatriantafilou and M. Patiño-Martínez, "STONE: A streaming DDoS defense framework," *Expert Systems with Applications*, vol. 42, no. 24, pp. 9620-9633, Dec 2015.

[177] T. Vissers, T. Van Goethem, W. Joosen and N. Nikiforakis, "Manoeuvring Around Clouds: Bypassing Cloud-based Security Providers," in *the proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, Oct 2015, pp. 1530-154.

[178] T. Thapngam, S. Yu, W. Zhou and G. Beliakov, "Discriminating DDoS attack traffic from flash crowd through packet arrival patterns*,"* in*: proceedings of IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2011, pp. 952-957.

[179] M. Sachdeva, K. Krishan and S. Gurvinder, "A comprehensive approach to discriminate DDoS attacks from flash events," *Journal of Information Security and Applications*, vol. 26, pp. 8-22, Feb 2015.

[180] Á. MacDermott, Q. Shi, M. Merabti and K. Kifayat, "Security as a Service for a Cloud Federation*. In: the proceedings of 15th Post-Graduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting (PGNet2014)*, 2014, pp. 77-82.

[181] A. Fragkiadakis, V. Siris, N. Petroulakis and A. Traganitis, "Anomaly-based intrusion detection of jamming attacks, local versus collaborative detection," *Wireless Communications and Mobile Computing*, vol. 15, no. 2, pp. 276-94, Feb 2015.

[182] A. Tartakovsky, S. Polunchenko and G.Sokolov, "Efficient computer network anomaly detection by change-point detection methods," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no.1, pp. 4-11, Feb 2013.

[183] T. Peng, C. Leckie and K. Ramamohanarao, "Proactively detecting distributed denial of service attacks using source IP address monitoring," in*: proceeding of 3rd International Networking Conference (IFIO-TC6)*, May 2004, pp. 771-782.

[184] H. Takada and U. Hofmann, "Application and analyses of cumulative sum to detect highly distributed denial of service attacks using different attack traffic patterns," 2004, http://www.ist-intermon. org/dissemination/newsletter7.pdf.

[185] T. Nguyen and G. Armitage, "A survey of techniques for Internet traffic classification using machine learning," *IEEE Communications Surveys and Tutorials*, vol.10, no. 4, pp. 56-76, Oct 2008.

[186] E. Garsva, N. Paulauskas, G. Grazulevicius and L. Gulbinovic, "Packet inter-arrival time distribution in academic computer network," *ElektronikairElektrotechnika*, vol. 20, no. 3, pp. 87-90, Dec 2014.

[187] M. Jaber, R. Cascella and C. Barakat, "Can we trust the inter-packet time for traffic classification?" in *IEEE International Conference on Communications (ICC)*, June 2011, pp. 1-5.

[188] L. Arshadi and A. Jahangir, "On the TCP Flow Inter-arrival Times Distribution," in *Fifth IEEE UKSim European Symposium on Computer Modelling and Simulation (EMS)*, Nov 2011, pp. 360-365.

[189] V. De Oca, D. Jeske, Q. Zhang, C. Rendon and M. Mazda, "A cusum change-point detection algorithm for non-stationary sequences with application to data network surveillance," *Journal of Systems and Software*, vol. 83, no. 7, pp. 1288-1297, July 2010.

[190] A. Polunchenko and A. Tartakovsky, "State-of-the-art in sequential change-point detection," *Methodology and Computing in Applied Probability*, vol. 14, no. 3, pp. 649-684, Sept 2012.

[191] W. Schmid, "On-the-run length of a Shewhart chart for correlated data," *Statistical Papers*, vol. 36, no. 1, pp. 111-130, Dec 1995.

[192] A. Wald, "Sequential tests of statistical hypotheses," *The Annals of Mathematical Statistics*, vol. 16, no. 2, pp. 117-186, June 1945.

[193] D. Martínez-Rego, D. Fernández-Francos, O. Fontenla-Romero and A. Alonso-Betanzos, "Stream-change detection via passive-aggressive classification and Bernoulli CUSUM," *Information Sciences*, vol. 305, pp. 130-145, June 2015.

[194] S. Gavaskar, R. Surendiran and D. Ramaraj, "Three Counter-Defense Mechanisms for TCP SYN Flooding Attacks," *International Journal of Computer Applications*, vol. 6, no. 6, pp. 12-15, Sept 2010.

[195] O. Osanaiye**,** H. Cai, K-KR. Choo, A. Dehghantanha, Z. Xu and M. Dlodlo, "Ensemble-based Multi-Filter Feature Selection Method for DDoS Detection in Cloud Computing," *EURASIP Journal for Wireless and Communications Network*, no.1, pp. 1-10, May 2016.

[196] O. Osanaiye**,** K-KR Choo and M. Dlodlo, "Change-Point Cloud DDoS Detection using Packet Inter-Arrival time," *8th Computer Science & Electronic Engineering Conference (CEEC'16).*

[197] M. Aazam and E.-N. Huh, "Fog Computing and smart gateway based communication for Cloud of Things," in: *proceedings of IEEE International Conference on Future Internet of Things and Cloud (FiCloud)*, August 2014, pp. 464–470.

[198] M. Díaz, C. Martín and Rubio B, "State-of-the-art, challenges, and open issues in the integration of Internet of Things and Cloud Computing," *Journal of Network and Computer Applications*, vol. 67, pp. 99-177, May 2016.

[199] F. Bonomi, R. Milito, P. Natarajan and J. Zhu, "Fog Computing: A platform for Internet of Things and analytics," *The Series Studies in Computational Intelligence, Big Data and Internet of Things: A Roadmap for Smart Environments*, vol. 546, pp. 168-186, March 2014.

[200] Cisco, "Fog Computing and Internet of Things: Extend the Cloud to Where the Things Are, A white paper," pp. 1-6, April 2015.

[201] V. Sehgal, A. Patrick, A. Soni and L. Rajput, "Smart Human Security Framework Using Internet of Things, Cloud and Fog Computing," in: *Series of Intelligent Distributed Computing*, vol. 321, pp. 251-263, 2015.

[202] M. Yannuzzi, R. Milito, R. Serral-Gracià, D. Montero and M. Nemirovsky, "Key ingredients in an IoT recipe: Fog Computing, Cloud computing, and more Fog Computing," in: *proceedings of 19th IEEE International Workshop on Computer Aided Modelling and Design of Communication Links and Networks (CAMAD)*, Dec 2014, pp. 325-329.

[203] G. Suciu, V. Suciu, A. Martian, R. Craciunescu, A. Vulpe, I. Marcu, S. Halunga, and O. Fratu, "Big Data, Internet of Things and Cloud Convergence–An Architecture for Secure E-Health Applications," *Journal of Medical Systems*, vol. 11 issue 39, pp. 1-8, Nov 2015.

[204] S. Cirani, G. Ferrari, N. Iotti and M. Picone, "The IoT hub: a Fog node for seamless management of heterogeneous connected smart objects," in: *proceedings of 12th Annual IEEE International Conference on Sensing, Communication, and Networking-Workshops (SECON Workshops)*, June 2015, pp. 1-6.

[205] J. Varia, “Best practices in architecting cloud applications in the AWS cloud,” *Cloud Computing. Principles and Paradigms*, John Wiley & Sons, Inc. Jan 2011, pp. 459-490.

[206] F. Longo, R. Ghosh, V. Naik and K.Trivedi, “A scalable availability model for infrastructure-as-a-service cloud,” in: *proceedings of the 41st IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*, June 2011, pp. 335-346.

[207] [online] https://svn.nmap.org/nmap/nmap-os-db [Accessed 14/06/2016]

Appendix

Appendix A

Summary of Filter Feature Ranking Methods

A.1 Information Gain (IG)

Entropy is the root of IG ranking method which is often used in information theory measure to determine the purity of an arbitrary collection of examples. The entropy measure determines the uncertainty of each feature prior to ranking, according to their relevance in determining the output class.

The entropy of variable X can therefore be defined as

$$H(X) = -\sum_{i} P(x_i) log_2\big(P(x_i)\big) \quad (1)$$

Let $P(x_i)$ denotes the value of prior probabilities of X. The entropy of X after observing value of another variable Y is defined as

$$H(X/Y) = -\sum_{j} P(y_j) \sum_{i} P\big(x_i|y_j\big) log_2 \Big(P\big(x_i|y_j\big)\Big) \quad (2)$$

In Equation 2, $P\big(x_i|y_j\big)$ is the posterior probability of X given Y. The information gain is defined as the amount by which the entropy of X decreases to reflect an additional information about X provided by Y and is defined as

$$IG(X/Y) = H(X) - H(X|Y). \quad (3)$$

Based on this measure, it is clear that features Y and X are more correlated than features Y and Z, if $IG(X/Y) > IG(Z/Y)$. The feature ranking can, therefore, be calculated using Equation 3. This ranking will be used to select the most important features form the NSL-KDD dataset as shown in table B1 below.

Table A1: Ranked features of NLS-KDD dataset using IG

Rank	Value	Feature	Rank	Value	Feature
1	0.807	src_bytes	22	0.064	protocol_type
2	0.672	service	23	0.057	rerror_rate
3	0.632	dst_bytes	24	0.054	dst_host_rerror_rate
4	0.519	Flag	25	0.053	srv_rerror_rate
5	0.516	diff_srv_rate	26	0.034	duration
6	0.508	same_srv_rate	27	0.011	Hot
7	0.473	dst_host_srv_count	28	0.010	wrong_fragment
8	0.439	dst_host_same_srv_rate	29	0.006	num_compromised
9	0.413	dst_host_diff_srv_rate	30	0.004	num_root
10	0.404	dst_host_serror_rate	31	0.002	num_access_files
11	0.402	logged_in	32	0.001	is_guest_login
12	0.396	dst_host_srv_serror_rate	33	0.001	num_file_creations
13	0.391	serror_rate	34	0.001	su_attempted
14	0.382	Count	35	0.000	root_shell
15	0.377	srv_serror_rate	36	0.000	num_shells
16	0.269	dst_host_srv_diff_host_rate	37	0.000	Land
17	0.195	dst_host_count	38	0.000	num_failed_logins
18	0.193	dst_host_same_src_port_rate	39	0.000	Urgent
19	0.144	srv_diff_host_rate	40	0.000	num_outbound_cmds
20	0.094	srv_count	41	0.000	is_host_login
21	0.089	dst_host_srv_rerror_rate			

Figure A1 below presents the ranked features of NSL-KDD dataset in relation with its output class

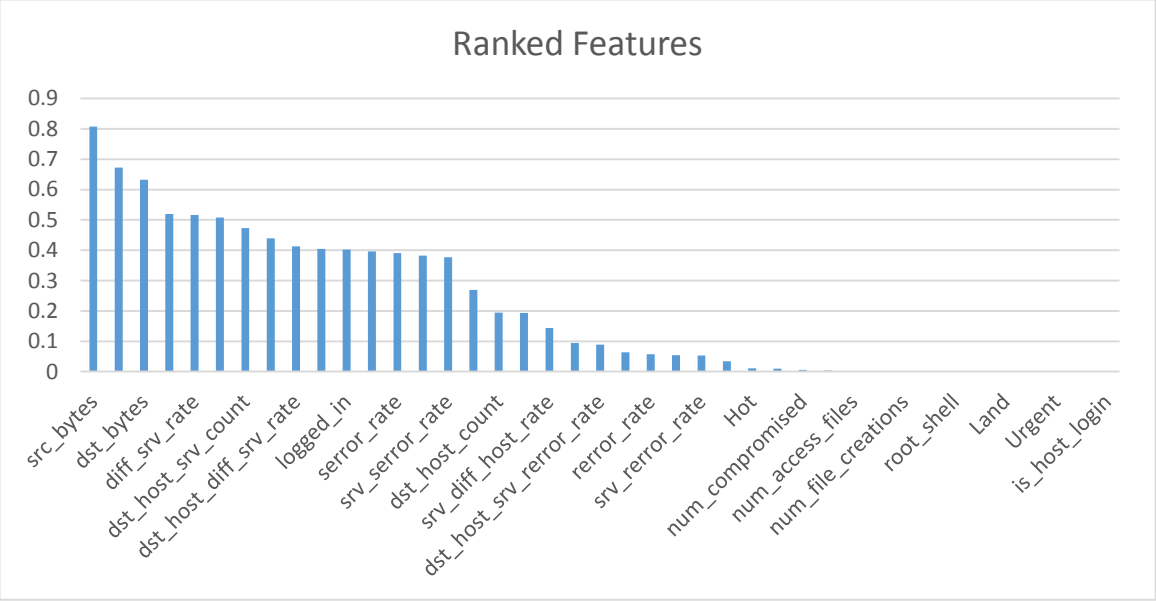

Fig A1: Figure of ranked features of NSL-KDD dataset using IG

A.2 Gain Ratio

Gain ratio was introduced to improve the bias of IG with respect to features with large diversity value. The intrinsic information of a given distribution can be determined by the entropy distribution of the instance value.

Gain ratio of a given feature x and a feature value y can be calculated using Equations 4 and 5.

$$Gain\ Ratio\ (y,x) = \frac{Information\ Gain(y,x)}{Intrinsic\ Value(x)} \quad , \qquad (4)$$

$$\text{Where Intrinsic Value } (x) = -\sum \frac{|S_i|}{|S|} * Log_2 \frac{|S_i|}{|S|} \qquad (5)$$

Note that $|S|$ is the number of possible values feature x can take, while $|S_i|$is the number of actual values of feature x. Gain ratio has been used to rank the features of NSL-KDD benchmark dataset and is presented in table A2

Table A2: Ranked features of NLS-KDD dataset using Gain ratio

Rank	Value	Feature	Rank	Value	Feature
1	0.415	logged_in	22	0.083	rerror_rate
2	0.371	srv_serror_rate	23	0.082	dst_host_same_src_port_rate
3	0.340	flag	24	0.080	num_root
4	0.337	serror_rate	25	0.078	su_attemped
5	0.331	dst_host_srv_serror_rate	26	0.074	protocol_type
6	0.324	dst_bytes	27	0.070	Hot
7	0.280	diff_srv_rate	28	0.066	num_compromised
8	0.277	dst_host_serror_rate	29	0.064	num_access_files
9	0.261	src_bytes	30	0.061	Duration
10	0.261	same_srv_rate	31	0.055	dst_host_rerror_rate
11	0.172	service	32	0.033	num_shells
12	0.169	dst_host_srv_diff_host_rate	33	0.032	num_file_creations
13	0.157	dst_host_same_srv_rate	34	0.029	srv_count
14	0.154	dst_host_srv_count	35	0.016	root_shell
15	0.134	wrong_fragment	36	0.015	is_guest_login
16	0.131	dst_host_diff_srv_rate	37	0.000	Land
17	0.122	count	38	0.000	num_failed_logins
18	0.121	srv_diff_host_rate	39	0.000	num_outbound_cmds
19	0.102	dst_host_srv_rerror_rate	40	0.000	Urgent
20	0.100	dst_host_count	41	0.000	is_host_login
21	0.084	srv_rerror_rate			

Figure A2 below presents the ranked features of NSL-KDD dataset in relation with its output class using gain ratio filter feature selection method

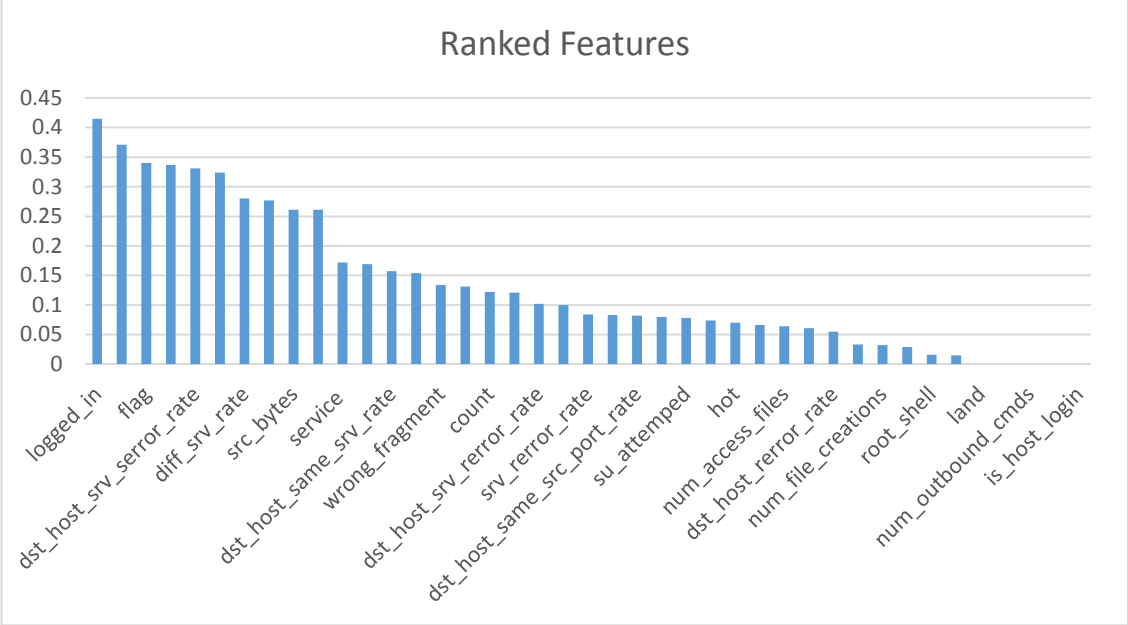

Fig A2: Figure of ranked features of NSL-KDD dataset using gain ratio

A.3 Chi-squared

Chi-squared statistic can be used to test the independence of two variables by computing a score to determine the extent of independence between the two variables. Chi-squared can be determined using equation 6.

$$\chi^2(r, c_i) = \frac{N[P(r,c_i)P(\bar{r},\bar{c}_i) - P(r,\bar{c}_i)P(\bar{r},c_i)]^2}{P(r)P(\bar{r})P(c_i)P(\bar{c}_i)} \quad , \qquad (6)$$

Where N denotes the entire dataset and r indicates the presence of the feature ($\bar{r}$ its absence), c_i refers to the class. $P(r, c_i)$ is the probability that feature r occurs in class c_i and $P(\bar{r}, c_i)$ is the probability that the feature r does not occur in class c_i. Also, $P(r, \bar{c}_i)$ and $P(\bar{r}, \bar{c}_i)$ are the probabilities that the features does or does not occur in a class that is not labelled c_i and so on. $P(r)$ is the probability that the feature appears in the dataset while $P(\bar{r})$ is the probability that the feature does not appear in the dataset. $P(c_i)$ and $P(\bar{c}_i)$ is the probability that a dataset is labelled to class c_i or not.

Table A3 presents the ranked features of NSL-KDD benchmark using chi-squared.

Table A3: Ranked features of NLS-KDD dataset using Gain ratio

Rank	Value	Feature	Rank	Value	Feature
1	21786.7	src_bytes	22	02035.9	protocol_type
2	18639.1	service	23	01798.1	rerror_rate
3	17581.0	dst_bytes	24	01793.5	dst_host_rerror_rate
4	15153.7	Flag	25	01728.2	srv_rerror_rate
5	14723.0	same_srv_rate	26	01005.9	duration
6	14715.6	diff_srv_rate	27	00319.3	Hot
7	13897.3	dst_host_srv_count	28	00258.8	wrong_fragment
8	13402.9	dst_host_same_srv_rate	29	00193.0	num_compromised
9	12617.5	dst_host_diff_srv_rate	30	00103.0	num_root
10	11927.4	logged_in	31	00055.0	num_access_files
11	11742.6	Count	32	00037.7	is_guest_login
12	11419.5	dst_host_serror_rate	33	00026.3	num_file_creations
13	11309.4	serror_rate	34	00018.4	su_attempted
14	11023.2	dst_host_srv_serror_rate	35	00008.7	root_shell
15	10810.4	srv_serror_rate	36	00004.6	num_shells
16	07461.2	dst_host_srv_diff_host_rate	37	00000.0	Land
17	06183.5	dst_host_same_src_port_rate	38	00000.0	num_failed_logins
18	06171.9	dst_host_count	39	00000.0	Urgent
19	03928.3	srv_diff_host_rate	40	00000.0	num_outbound_cmds
20	02947.4	srv_count	41	00000.0	is_host_login
21	02712.4	dst_host_srv_rerror_rate			

Figure A3 below presents the ranked features of NSL-KDD dataset in relation with its output class using chi-squared filter selection method

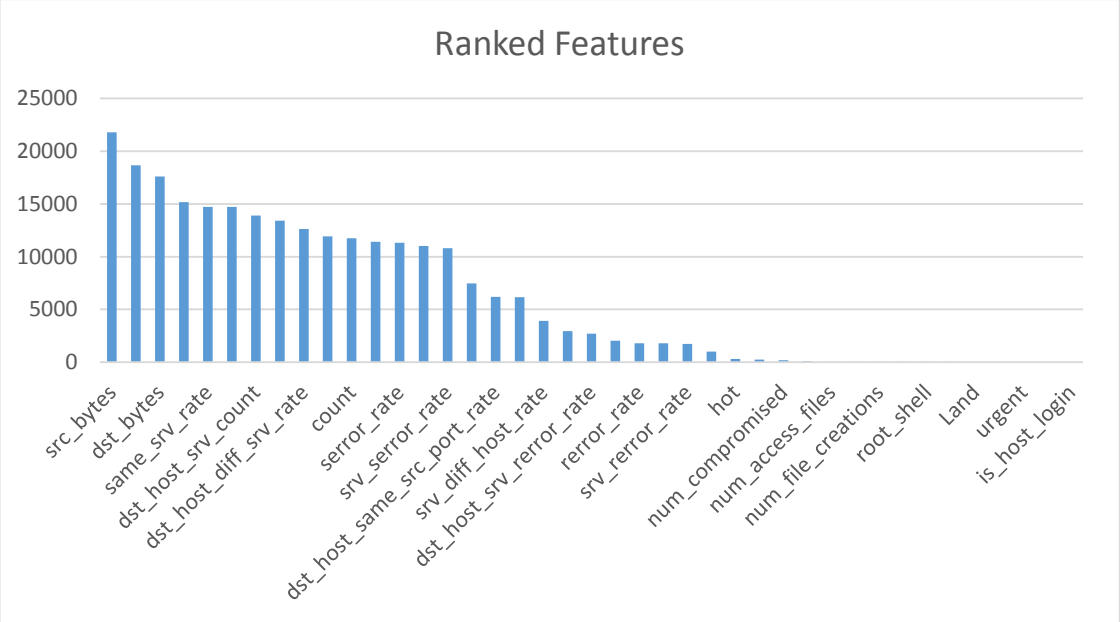

Fig A3: Figure of ranked features of NSL-KDD dataset using chi-squared

A.4 ReliefF

ReliefF feature selection evaluates the worth of a feature by using continuous sampling to distinguish between the nearest hit and the nearest miss. The attribute evaluator is used to append attach weight to each feature according to their strength to distinguish the different classes.

ReliefF attempts to approximate the difference of probabilities for the weights of a feature X:

$$W_X = P\,(different\ value\ of\ X | nearest\ instance\ of\ different\ class) - P(different\ value\ of\ X | nearest\ instance\ of\ same\ class\ ) \quad (7)$$

After removing the context sensitivity provided by the "nearest instance" condition, the attributes are treated as independent of one another. This is expressed in equation 8

$$Relief_x = P(different\ value\ of\ X | different\ class) - P(different\ value\ of\ X | same\ class) \quad (8)$$

Table A4 presents the ranked features of NSL-KDD benchmark using ReliefF

Table A4: Ranked features of NLS-KDD dataset using ReliefF

Rank	Value	Feature	Rank	Value	Feature
1	0.517	service	22	0.012	srv_rerror_rate
2	0.242	same_srv_rate	23	0.010	Hot
3	0.237	Flag	24	0.008	wrong_fragment
4	0.205	dst_host_count	25	0.008	rerror_rate
5	0.169	dst_host_serror_rate	26	0.007	Duration
6	0.140	dst_host_srv_count	27	0.003	dst_bytes
7	0.138	dst_host_srv_serror_rate	28	0.003	num_failed_logins
8	0.126	logged_in	29	0.002	is_guest_login
9	0.125	dst_host_same_src_port_rate	30	0.000	root_shell
10	0.117	count	31	0.000	num_compromised
11	0.094	srv_seeror_rate	32	0.000	su_attempted
12	0.080	dst_host_same_srv_rate	33	0.000	urgent
13	0.076	dst_host_rerror_rate	34	0.000	num_shells
14	0.064	srv_diff_host_rate	35	0.000	src_bytes
15	0.055	dst_host_diff_srv_rate	36	0.000	num_file_creations
16	0.043	diff_srv_rate	37	0.000	num_access_files
17	0.034	srv_count	38	0.000	num_root
18	0.018	dst_host_srv_rerror_rate	39	0.000	is_host_login
19	0.017	protocol_ type	40	0.000	num_outband_cmds
20	0.015	dst_host_srv_diff_host_rate	41	0.000	Land
21	0.015	serror_rate			

Figure A4 below presents the ranked features of NSL-KDD dataset in relation with its output class using RefliefF filter selection method

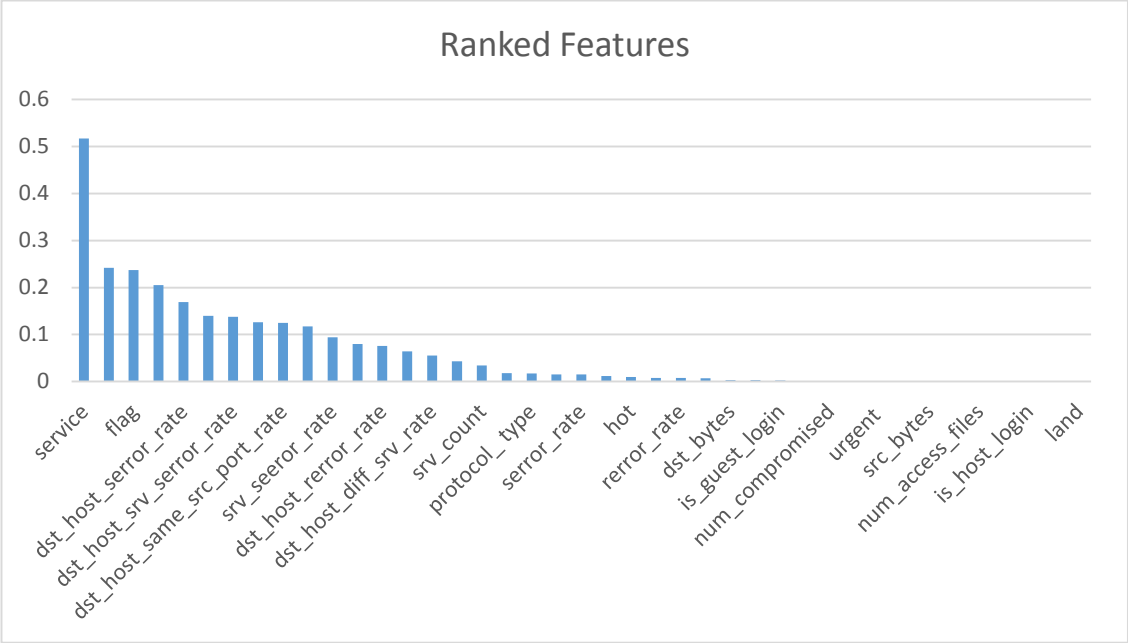

Fig A4: Figure of ranked features of NSL-KDD dataset using ReliefF

Appendix B

B.1 Change-point using CUSUM algorithm to detect Cloud DDoS flooding attack

Analysing a change-point involves determining whether a change has taken place within series of random variables. The change-point analysis [96] can be used when dealing with large dataset capable of detecting the occurrence of subtle changes that occur, that are often missed by control charts. Traditionally, control charts were often used to detect changes and differs from change-point analysis, because it can be updated during the collection of each data point. The latter, on the other hand can only be carried out once all the data is collected. One of the main benefits of change-point analysis is that it controls the change-wise error rate, therefore, each detected change is likely to be real.

In this thesis, two datasets have been deployed, namely; CAIDA DDoS dataset (one of the most current datasets), which is made up of an hour of anonymized trace from DDoS attacks and Auckland-VIII dataset from Waikato Internet Traffic Storage (WITS) for non-attack packets.

B.2 Cumulative Sum (CUSUM) algorithm

Taylor [96] suggested both CUSUM and bootstrapping. Let $x_1, x_2, x_3 \ldots\ldots\ldots\ldots, x_t$ represent the packet inter-arrival time of the monitored packets. The cumulative sum C_i can be calculated as follows:

For the non-attack:

$$\lambda_0 = \frac{1}{n}\sum x_1 + x_2 + x_3 + \cdots . x_t \qquad (1)$$

Where λ_0 is the average IAT during the pre-attack period.

For the attack period:

$$\lambda_1 = \frac{1}{n}\sum x_1 + x_2 + x_3 + \cdots . x_t \qquad (2)$$

Therefore, the average inter-arrival time is denoted by λ, that is $\lambda = \lambda_0 + \lambda_1$

During the monitored period, let P_m (λ,t) represent the distribution of the monitored packets in a traffic flow during the sampled period with packet inter-arrival time x_t at time t with mean λ.

Now, suppose the series of packets in a traffic flow contains a DDoS attack; there would be an abrupt increase in the IAT value x_t , when $t > \tau$, where τ is the point of change in the packet IAT in the traffic flow. The distribution of monitored packet during the sampled period P_m (λ,t) that consist of the pre-attack (normal) period P_m (λ_0, t) and the attack period P_m (λ_1, t), as expressed in equation 3.

$$P_m(\lambda, t) = \begin{cases} P_m(\lambda_0, t); t \leq \tau \\ P_m(\lambda_1, t); t > \tau \end{cases} \quad (3)$$

The average IAT denoted by λ of x_t will take the value λ_0 when $t \leq \tau$ during a normal period, and λ_1 during an attack period, where $> \tau$.

Due to the bursty nature of the DDoS attack, the mean IAT $\lambda_1 > \lambda_0$. Therefore, to detect the change due to this increase, we deploy a change-point detection using CUSUM algorithm. The CUSUM deployed is a statistical process control algorithm that dynamically detects the point of change in mean value from the observed series and responds appropriately.

To compute the CUSUM statistic:

Let C_0 be equal to zero,

Therefore, C_i are computed recursively by adding the difference between the current value and the previous sum, as expressed in equation 4.

$$C_i = C_{i-1} + (X_i - \lambda) \; for \; i = 1,2,3, \ldots N \quad (4)$$

From the equation 4 above, it must be noted that the cumulative sums are not the cumulative sums of the values, rather, they are the cumulative sums of the difference between the individual values and the average. These CUSUM value always starts from zero and moves to a negative value during normal condition. On detecting a change, it abruptly moves to a positive value [193] and ends at zero.

Interpreting CUSUM requires some skills. Suppose that during the period of monitor, the average IAT value tends to be above the overall average. This means the value that is added to the cumulative sum will be positive, thereby abruptly increasing the sum. Therefore, in interpreting the CUSUM chart, the change in the slope upwards indicate a period where the values tend to be above the overall average. A downward slope, also indicate a monitoring period where the values tend to be lower than the overall average. The changes observed in these directions will mean a sudden shift in the

average, while values that follow a relative straight part during the monitored period indicates no major change in the average value.

To determine the apparent change, a confidence level can be conducted by performing a bootstrap analysis [96]. Before performing the bootstrap analysis, it is often required to estimate the magnitude of change. To achieve this, the preferred choice that works well notwithstanding the distribution and the occurrence of multiple changes is C_{diff}, defined as:

$$C_{diff} = C_{max} - C_{min} \quad (5)$$

Where, $C_{max} = {}_{i=0,\ldots\ldots n}^{max} C_i$

$C_{min} = {}_{i=0,\ldots\ldots n}^{min} C_i$

As soon as the estimator of the magnitude of change has been selected, the bootstrap analysis can thereafter be performed. A single bootstrap can be performed through the following process:

(i) Generate a bootstrap sample of n data points, denoted as $x_1^0, x_2^0, x_3^0 \ldots\ldots x_n^0$ by randomly reordering the original values. This is termed sampling without replacement.

(ii) Based on the bootstrap sample in (i), determine the bootstrap CUSUM, denoted as $C_1^0, C_2^0, C_3^0 \ldots\ldots C_n^0$

(iii) Determine the maximum, minimum and the difference of the bootstrap CUSUM, denoted as $C_{max}^0, C_{min}^0, and\ C_{diff}^0$

(iv) Determine whether the bootstrap difference C_{diff}^0 is less than the original difference C_{diff}

The bootstrap analysis involves the process of performing a large number of bootstrap and also counting the number of bootstraps for which $C_{diff}^0 < C_{diff}$. Let N be the number of bootstrap samples carried out and Y be then number of bootstraps for which C_{diff}^0 is less than C_{diff}. The confidence level signifying that a change has occurred can therefore be calculated as a percentage as follows:

Confidence Level = $100\,\frac{Y}{N}$ % (6)

Bootstrapping results in a distribution-free approach with just a single assumption, that of an independent error structure.

Immediately a change is detected, an estimate is made to determine when the change occurred. One of such estimator is the CUSUM estimator. Let's define m such that:

$$|S_m| = \underset{i=0,1,2.....,n}{max}|S_i| \quad (7)$$

S_m in equation (7) is the point that is furthest from zero in the CUSUM chart. The point m estimates the last point before the change occurred. The point m+1 is therefore the first point after the detected change. The mean square error (MSE) is the second estimator of when the change occurred.

The MSE (m) can therefore be defined as:

$$\text{MSE (m)} = \sum_{i=1}^{m}(x_i - \lambda_0)^2 + \sum_{i=m+1}^{n}(x_i - \lambda_1)^2 \quad (8)$$

Where, $\lambda_0 = \frac{\sum_{i=1}^{m} x_i}{m}$ and $\lambda_1 = \frac{\sum_{i=m+1}^{n} x_i}{n-m}$

During MSE error estimation, the data is split into two different segments, 1 to m and m+1 to n, before observing how well the data fits into the two estimated averages. The value of m, which minimizes MSE (m) is regarded as the best estimator of the last point before the change occurred, while the point m+1 signify the first point after the detected change. In a similar fashion, data of each segment can be passed through the method mentioned above to determine the level 2 change-points that divides corresponding segments into sub segments. Repeating the procedure above ensures the detection of significant change-points at subsequent levels for which each of the associated confident limits and levels can be determined using bootstrapping. In a similar manner, multiple change-point can be detected by introducing additional change-points at each successive pass that will continue to split the segments into two. Backward elimination is used to remove those points that does not qualify test of significance once the change-point and the associated confidence level has been detected. In other to limit the rate of false detection when a point is eliminated, the surrounding change-points are re-estimated together with their significance level.