PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By IBRAHIM M. WAZIRI, JR.

Entitled
PACKET FILTER PERFORMANCE MONITOR (ANTI-DDOS ALGORITHM FOR HYBRID TOPOLOGIES)

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

VICTOR RASKIN Co-chair	SAMUEL S. WAGSTAFF, JR.
JULIA M. TAYLOR Co-chair	
J. ERIC DIETZ	
BAIJIAN YANG	

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): VICTOR RASKIN

Approved by: EUGENE. H. SPAFFORD 7/19/2016
Head of the Departmental Graduate Program Date

PACKET FILTER PERFORMANCE MONITOR

(ANTI-DDOS ALGORITHM FOR HYBRID TOPOLOGIES)

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Ibrahim M. Waziri, Jr.

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2016

Purdue University

West Lafayette, Indiana

To my Dad

&

Younger Brother - Mohammed Waziri

(May his soul continue to Rest in Peace - Amen.)

ACKNOWLEDGMENTS

I would like to express my immense gratitude to my advisors: Prof. Victor Raskin and Prof. Julia Taylor for their time, constant guidance, moral support, and invaluable comments throughout my PhD. I am honored to have the opportunity to know and learn from such great and highly respected experts. I would also like to express my appreciation and gratitude to my committee members: Prof. J. Eric Dietz, Prof. Baijian Yang, and Prof. Samuel S. Wagstaff, Jr for their invaluable support, time, and help. Without their guidance, my PhD would never have been possible.

I am grateful to my parents for their tireless support and unwavering encouragement, especially my dad, who I look up to no matter how tall I grow. I owe him everything I am and have. Without my father, my dreams would never have become a reality. And for that I just want to say, thank you, dad!

My sincere appreciation goes out to Purdue CERIAS members, the Professors who took the time to teach me, the staff for their guidance, and my friends for taking me along on this incredible journey.

Lastly, I would like to thank the Purdue Graduate School, especially my supervisor Don Brier in the IMA office for providing me with an assistantship that funded my PhD studies.

Thank you all.

Ibrahim Waziri, Jr.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ACL	Access Control List
AES	Advanced Encryption Standard
AFM	Aggregation Flow Management
API	Application Program Interface
APT	Advanced Persistent Threat
AS	Autonomous Systems
CAPEX	Capital Expenditure
CBF	Confidence Based Filtering
CFU	Cloud Fusion Unit
CIA	Confidentiality Integrity Availability
DDoS	Distributed Denial of Service
DES	Data Encryption Standard
EC	Elastic Computing
FPGA	Field Programmable Gate Array
FTA	Fault Tree Analysis
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
ICMP	Internet Control Message Protocol
IPS	Intrusion Prevention System
IS	Information Security
LAN	Local Area Network
LISP	Locator ID Separation Protocol
MAC	Message Authentication Code
NAT	Network Address Translation

NFV	Network Function Virtualization
NIST	National Institutes of Standards and Technology
NTP	Network Time Protocol
OPEX	Operation Expenditure
OS	Operating System
PaaS	Platform as a Service
SaaS	Software as a Service
SDK	Software Development Kit
SDN	Software Defined Network
SLA	Service Level Agreement
SMLI	Stateful Multi-Large Inspection
SOS	Secure Overlay Services
SSDP	Simple Service Discovery Protocol
SSL	Secure Socket Layer
SVA	Security Virtual Appliance
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VM	Virtual Machine
VMM	Virtual Machine Monitor
VPN	Virtual Private Network
WWW	World Wide Web

ABSTRACT

Ibrahim M. Waziri, Jr. Ph.D., Purdue University, August 2016. Packet Filter Performance Monitor (Anti-DDoS Algorithm for Hybrid Topologies). Major Professors: Victor Raskin & Julia Taylor.

DDoS attacks are increasingly becoming a major problem. According to Arbor Networks, the largest DDoS attack reported by a respondent in 2015 was 500 Gbps. Hacker News stated that the largest DDoS attack as of March 2016 was over 600 Gbps, and the attack targeted the entire BBC website.

With this increasing frequency and threat, and the average DDoS attack duration at about 16 hours, we know for certain that DDoS attacks will not be going away anytime soon. Commercial companies are not effectively providing mitigation techniques against these attacks, considering that major corporations face the same challenges. Current security appliances are not strong enough to handle the overwhelming traffic that accompanies current DDoS attacks. There is also a limited research on solutions to mitigate DDoS attacks. Therefore, there is a need for a means of mitigating DDoS attacks in order to minimize downtime. One possible solution is for organizations to implement their own architectures that are meant to mitigate DDoS attacks.

In this dissertation, we presented and implemented an architecture that utilizes an activity monitor to change the states of firewalls based on their performance in a hybrid network[1]. Both firewalls are connected inline. The monitor is mirrored to monitor the firewall states. The monitor reroutes traffic when one of the firewalls becomes overwhelmed due to a HTTP DDoS flooding attack. The monitor connects to the API of both firewalls. The communication between the firewalls and monitor is encrypted using AES, based on PyCrypto python implementation.

[1]A hybrid network is a network comprised of both hardware and virtual firewalls

This dissertation is structured in three parts. The first part found the weakness of the hardware firewall and determined its threshold based on spike and endurance tests[2]. This was achieved by flooding the hardware firewall with HTTP packets until the firewall became overwhelmed and unresponsive. The second part implements the same test as the first, but targeted towards the virtual firewall. The same parameters, test factors, and determinants were used; however a different load tester was utilized. The final part was the implementation and design of the firewall performance monitor.

The main goal of the dissertation is to minimize downtime when network firewalls are overwhelmed as a result of a DDoS attack.

[2]Memory & CPU Utilization were used as determinants

1. INTRODUCTION

Individuals and organizations store data either in their individual computers or organization servers. Networking allow users to share data with just a click. The internet has revolutionized the communication world; it is a medium for data dissemination and a means for collaboration and interaction between individuals via their computers, irrespective of geographical location. To ensure connectivity and ease of access to information, two ways of storing information are in place: 1) using a physically - accessible storage device or; 2) using a cloud service that connects remotely by using the network.

One feature of networking is its capability to give access to the users connected to that network. Without security measures in place, there would be no control over access to information on a network. Now that the internet connects every electronic device to a single global network, ensuring data security has become a tremendous challenge. Distributed Denial of Service (DDoS), Advanced Persistent Threats (APTs), and other forms of attack are continuously increasing, according to *Fortune Magazine* (Gandel, 2015) Cyber attacks cost companies $400 billion annually. Another report from *CNN Money* estimates that the average US firm spends at least $15 million a year on cyber crime (Griffiths, 2015). Current protection mechanisms are inadequate to address the evolving cyber threats (Griffiths, 2015). Advancements in our defense mechanisms cannot manage the current rate at which new cyber threats arise (Griffiths, 2015).

Different appliances and defense methods are used to protect and ensure data security within a network. However, as technology advances, so does the mechanisms used to protect that particular technology. In networking and cloud computing we incorporate traditional and virtual systems, which results in a hybrid network topology (Buyya, Broberg, & Goscinski, 2010). For corporations and individuals to better secure data and information in a network, they need to implement different security measures. One of the most integral aspect of network security implementation is the deployment of a firewall (S. Ioannidis, Keromytis, Bellovin, & Smith, 2000).

Given the traditional and virtual network environments utilized by organizations, the need arises for firewalls that protect both traditional and virtual environments. It is a common practice to have a hardware firewall dedicated to the traditional network, and a virtual firewall dedicated to the virtual network. The problem with this implementation is the fact that different types of attacks can be implemented within different network environments. If an attack is targeted towards a traditional network, the virtual network gets affected, and vice-versa. For example, if a DDoS attack targeted towards a dedicated traditional network firewall (hardware) is successful, the firewall becomes non-responsive, therefore making the virtual network non-responsive.

1.1 Motivation

In this section, we discuss why both hardware and virtual firewalls fail under DDoS attacks in hybrid architectures. This is the reason that motivates us to build a performance monitor that monitors the state of the firewalls.

1.1.1 Challenges with Traditional Packet Filter Architectures

Traditional (Hardware) firewall appliances are closed boxes that connect to the external network on one interface(s), and the internal network on the other interface(s). Hardware firewalls have minimal operating systems, which makes them fast in terms of processing capabilities. Optimal performance of hardware firewalls is dependent upon the firewall's resources, which include the CPU and Memory availability (Kenney, 1996; Zalenski, 2002). Because of these factors, for a hardware firewall to function as expected, it is required to have enough resources to process instructions (Panko, 2010).

Denial of service attacks clogs up available resources (CPU, Memory, etc.) on a target system, thereby overwhelming the system. Overwhelming a system causes it to crash, reboot, or generally refuse any assistance to legitimate clients. DoS attack are exceptionally basic; surely, pretty much every server will undoubtedly experience such an attack sooner or later (Douligeris & Mitrokotsa, 2004). In a situation where the attack is facilitated crosswise over numerous hijacked systems (zombies) by an attacker (master), the attack is referred to as DDoS (Carl, Kesidis, Brooks, & Rai, 2006; Douligeris & Mitrokotsa, 2004). DDoS threats come in many varieties, some of which target the underlying server infrastructure. Others exploit vulnerabilities in applications and communication protocols. Unlike other kinds of cyber attacks which are typically launched to establish a long-term foothold and hijack sensitive information, denial of service assaults do not attempt to breach the security perimeter. Rather, they attempt to make services, websites, and servers unavailable to legitimate users. In some cases, however, DoS is also used as a smokescreen for other malicious activities, and to dismantle security appliances (e.g., web application firewalls).

A successful DDoS attack is a highly noticeable event that impacts the entire online user base (Mirkovic & Reiher, 2004). This makes a DDoS attack a popular weapon of choice for hacktivists, cyber vandals, extortionists, and anyone else looking to make a point or champion a cause (Douligeris & Mitrokotsa, 2004). DoS assaults often last for days, weeks, or even months at a time, which makes them extremely destructive to any network. DDoS can cause loss of revenues, erode consumer trust, force businesses to spend fortunes in compensations, and cause users to suffer long-term reputation damage (Kenney, 1996; Mirkovic & Reiher, 2004; Thomas & Stoddard, 2011).

Since the optimal operation of hardware firewalls is dependent on the firewall's available resources (Kenney, 1996). And DDoS are known to target and exhaust resources. This makes hardware firewalls, among other devices, an attack target (Byers, Rubin, & Kormann, 2004).

1.1.2 Challenges with Virtual Packet Filter Architectures

Network function virtualization (NFV) is a strategy to virtualize the network functions carried out by proprietary dedicated hardware. NFV decreases the quantity of proprietary hardware required to execute and run network services[1]. NFV allows network operators to integrate middle-boxes in virtual machines (VM) and put those VMs at subjective areas in the network (Anwer, Benson, Feamster, & Levin, 2015; ESTI, n.d.).

Network Function Virtualization (NFV) has drawn noteworthy consideration from both industry and the scholarly world with a vital movement within telecommunication service sector. By decoupling network functions (NFs) from the physical devices from which they run, NFV can possibly result in significant reduction in operating costs (OPEX) and capital costs (CAPEX), and to encourage the deployment of new services with expanded agility and faster time-to-value (Mijumbi et al., 2015). NFV is still in its earliest stages and there is a chance and opportunity for research groups to create new models, frameworks, and applications, and to assess choices and trade-offs in creating advance technologies for its optimized deployment.

NFV's have their own security issues, which include hyperthreats and hypercalls, as explained in (Shropshire, 2015). However, in this dissertation, we only focus on security issues that arise from DDoS attacks. Considering that middle-boxes are hardware appliances, they are expensive, hard to oversee, and their usefulness is hard or difficult to change. NFV has alleviated all these problems, with its flexibility (Martins et al., 2014). Because virtualized network functions are deployed on dedicated servers, which are hardware appliances, there is a need to secure the server appliances used to virtualize NFs. One tool that is being used to secure such server appliance perimeters is a hardware firewall.

[1]TechTarget - NFV defined

1.1.3 Challenges with Hybrid Packet Filters Architectures

A hybrid packet filter topology is required for a particular network architecture. Traditional (hardware) firewall secures the NFV server's perimeter, and the virtualized firewall secures other virtualized network functions. One packet filter inherits the problem of another packet filter (Buyya et al., 2010; Cheswick, Bellovin, & Rubin, 2003; Cisco, 2014).

Attacks that exhaust resources (such as DDoS) which are targeted towards the hardware firewalls used to secure the virtualized network functions server results in network downtime, and inaccessibility of anything connected inline beyond the firewall (Kenney, 1996; Martins et al., 2014). If the hardware firewall is down, the server that NFs runs on automatically becomes unavailable, and hence the unavailability of data and network downtime.

1.2 Problem Statement

This dissertation aims to enhance packet filter performance in hybrid networks and to minimize network downtime caused by DDoS attacks in hybrid packet filter topologies. This is achieved by monitoring the CPU and memory utilization of different firewalls. To ensure that, we claim the following: *it is possible to implement an architecture that mitigates DDoS attacks, minimizes network downtime, and transfers packet filtering service between firewalls in a hybrid network topology.* We formulate and validate the following hypotheses.

- It is possible to mitigate DDoS attacks on packet filters without dropping legitimate packets.
- It is possible to migrate packet filtering services interchangeably between firewalls in a hybrid packet filtering network when one firewall becomes overwhelmed.

- It is possible to develop a monitoring tool that monitors the CPU and Memory states of firewalls and minimizes network downtime during a DDoS attack on a network.

1.3 Contribution and Overview

The main contribution of this dissertation is to avoid network downtime as a result of firewall failures during a DDoS attack in a hybrid network. To ensure that, we implement a number of tests on the Traditional, Virtual, and Hybrid architectures and also develop an algorithm used to validate the hypothesis mentioned above. We provide here a brief overview of tests and algorithms:

1. **Spike Performance Test**
   To find the threshold of the firewalls (both hardware and virtual), we carried out a spike test that verifies the firewall's stability amid a burst of simultaneous concurrent or network connections to changing time periods and degrees of traffic load. This happens as a result of an attack that overwhelms the device (such as a DDoS attack). We implemented a HTTP flooding attack and monitored CPU Utilization during this test.

2. **Endurance Performance Test**
   Similar to the spike performance test, we carried out an endurance test. This was implemented to discover whether the firewalls can withstand the necessary processing loads for a long period of time. During the endurance test, we implemented a HTTP flooding attack, and monitored the memory utilization of the firewalls.

3. **Performance Monitoring/Packet Filtering Transfer Algorithm**
   In light of the outcomes of our tests, we developed an algorithm that monitors the states of the firewalls based on performance. The algorithm invokes instructions when a newly defined threshold is reached. The algorithm also

transfers packet filtering services between the firewalls when one of the firewalls is overwhelmed. This results in more network up-time in the event of an attack, and also ensures that the firewall does not become non-responsive when overwhelmed.

1.4 Dissertation Road-map

This work introduces a new security architectural monitor used in hybrid networks. The monitor works by transferring packet filtering services from one firewall to the other when the network is under DDoS attack. The communication between the monitor and the firewalls occurs through the firewall's API. AES encryption algorithm is used to ensure that all communication is secured.

In chapter 2, we review literature and research focused on network architectural security, NFV's, cyber threats (including DDoS attacks), and proposals for how DDoS attacks can be mitigated. We then examined commercial DDoS mitigation solutions that prevent and respond to flooding attacks. We finished the chapter by discussing the disadvantages of using commercial DDoS mitigation providers, and how they do not provide a substantive solution to DDoS attacks.

Chapter 3 discusses the traditional packet filter architecture. The chapter begins with an introduction to firewalls, and discusses how hardware firewalls differ from virtual firewalls. We then explained the problem statement and how we set up the traditional architecture. We concluded the chapter by presenting the results of our test.

Chapter 4 discusses the virtual packet filter architecture. We started by explaining NFVs and current issues with virtualization. We highlight the problem statement and explained our test parameters and results for the virtual architecture.

In chapter 5, we presented the packet filtering monitor/algorithm. We explained the monitor's design, how the monitor works, and evaluate the monitor's performance.

We concluded by summarizing our findings and proposing areas of future study in Chapter 6.

1.5 Dissertation Limitations

To the best of our knowledge, no existing research explains how monitoring firewall performance is used to mitigate DDoS attacks. This study focused only on Layer 7 - application level DDoS attacks, primarily based on HTTP traffic. HTTP traffic is the protocol utilized for communication between a client and a web server (Cisco, 2014; Dieter, 1999). All tests were conducted in a controlled, monitored environment. The virtualized firewall and the web server run on a dedicated server with configurations that are explained in later chapters. This work focus on internal threats, with the assumption that an intruder has gained access to the network. The process of gaining access to networks is beyond the scope of this study. The network is configured in IPv4 instead of IPv6 because IPv4 is the most generally utilized version of the Internet Protocol (Bade & Vanduhe, n.d.) and IPv6 features (e.g. larger address space, optimized DHCP, IPSec, optimized mobility feature (Bade & Vanduhe, n.d.)) are beyond the scope of this dissertation.

The test focuses on mitigating the application layer DDoS attacks, which are volumetric HTTP floods generated using different load and stress testing tools. The HTTP floods targets firewalls not other network devices. This study is not limited to specific vendors. The choice of devices used are solely based on availability, and the same process can be applied to any other vendor devices. This dissertation focuses on accessing a web server in which traffic must pass through two checkpoints (the firewalls) before reaching its destination.

Lastly, this dissertation does not attempt to provide a guide for malicious activity. We presented the report in an ethical manner, such that a reader cannot gain knowledge of how to implement DDoS attacks. However, if this dissertation provides any hint that an adversary decides to implement or exploit, it is the sole responsibility of the adversary and neither that of the author nor committee members.

2. RELATED WORK

2.1 Introduction

As of July 2015, the internet has connected an estimated of 1.03 billion hosts worldwide (ISC, 2015). The internet is an assembly of different networks accessible by dissimilar users in different ways. That means that users can access information using the internet regardless of national and geographical boundaries. This convenience and ease of accessibility to information, however, comes with security issues. These issues include, theft and tampering information. These issues make all information online vulnerable to unauthorized access and usage.

One of the most important aspects of technology is whether it favors offense or defense (R. Anderson, 2001). The difficulties of developing secure systems using a penetrate-and-patch methodology have been familiar to the security community since the Anderson report in the early 1970's (J. P. Anderson, 1972).

Within recent years we have seen an expansion in the widespread adoption of commercial security technologies by governmental, military and commercial organizations, due to their convenience, and ease of use. With increasing reliance on third-party security resources, also comes an increasing vulnerability to information meant to be protected (J. M. Anderson, 2003; Venter & Eloff, 2003). Although security threats can range from psychological operations (social engineering) to physical attacks on computers, one aspect of Information Security that most concerns computer users is defending information against disruption or disabling the computerized functions and resources that support an organization's operations (Jajodia, Ammann, & McCollum, 1999).

Many of the threats to Information Security share common characteristics (Peltier, 2005). We are going to discuss some of the threats known today, but it is worthwhile mentioning that these threats do not represent a complete list of Information security threats; considering new forms of attacks are discovered everyday. As it relates to this dissertation; Information Security threats can be categorized as follows: External and Internal.

- **External Threats** - This is the hacker threat, whether it is a single person, or a nation state. This type of threat comes from external sources, someone not connected to the network implements the attack. This types of attack includes DDoS attack (Spears, 2006).
- **Internal Threats** - This is a type of threat that happens inside the network. Internal threats do not only apply to malicious actives. User error and ignorance play a large role in trusted individuals putting networks and systems at risk to outside agents. Firewalls, intrusion detection systems, and other boundary defense mechanisms are ineffective when circumvented by insiders.

2.2 Firewall Architectural/Implementation Security

Everyday new research is being conducted within the security field. Network security is among the top research theme in information technology. Z. Yang, Qiao, Liu, Yang, and Wan (2010) focused on a collaborative trust model of firewall-through based on cloud computing. In this research, existing trust models and firewall innovation were studied. The researchers implemented a methodology using cloud computing that assess dynamic setting and presents the meaning of risk sign in firewalls. The model has three advantages: there are distinctive security strategies for various domains, the model considers the exchange setting, the verifiable information of entity dynamically impacts the estimation of trust worth. Finally, the trust model is synchronous with the firewall and does not break the firewall's local control policies. To confirm the dependability and accuracy of the proposed trust model, a test is done. The test result demonstrates that the trust model is robust and surpasses ordinary trust models in different areas, because it effectively control unauthorized access within cloud computing environment.

Moyer and Schultz (1996) presents an orderly approach for firewall penetration testing that advances the perspective that firewall testing ought to look at not just the capability of a firewall to stop attack from outside threats, but also the resistance of the whole system that the firewall ensure against external threats. As a result, therefore, testing should take an orderly approach to guarantee that it is complete, and to decrease the risk of threats and/or interference to the system and its hosts. This study introduced a requirement for an efficient approach to guide firewall testing.

Network firewalls, routers and switches utilize a rule database to choose which packet will be permitted into and out of a network. By filtering packets, the firewall, routers and switches can enhance security and execution. Be that as it may, as the extent of the rule list builds, it gets hard to keep up and validate the rules (Hazelhurst, Attar, & Sinnappan, 2000). Hazelhurst (2000) studied an algorithm for analyzing firewall and router access lists. The algorithm is another representation of rule list, and also a presentation of how Boolean expression can be utilized to analyze rule sets. Eronen and Zitting (2001) presented an expert system for analyzing firewall rules that looks into the problem of analyzing firewall configurations, using a tool that comprehends Cisco access lists, it is actualized utilizing Eclipse, (a constraint logic programming language). The study utilizes logic statements to express information about networking, firewalls, and basic configuration mistakes. Configuring network devices (especially from different vendors) to work in unison can be difficult. Notwithstanding reverse-engineering or figuring out of existing setup is hard. To overcome some of these difficulties, Mayer, Wool, and Ziskind (2000) studied a firewall analysis algorithm that composed and actualized a novel firewall analysis tool. The tool permits the administrator to effectively find and test the global firewall policy (either an implemented policy or an arranged one). The tool utilizes an insignificant depiction of the network topology, and specifically parses the different vendor-particular low-level configuration list. It interfaces with the client through a query-and-answer session, which is done at a higher point of abstraction. A common question the apparatus can answer is “from which machines can our DMZ be accessed, and with which implementations?” The tool compliments existing vulnerability examination tools, as it can be utilized before a policy is implemented. It works on a more justifiable level of abstraction, and it manages all firewalls simultaneously (Mayer et al., 2000).

Firewalls and routers must perform packet classifications at high speeds to productively execute their capacities, for example, firewalls and diffserv. Arrangement can be found on a self-assertive number of fields in the packet header. Rapidly classifying an arbitrary number of fields is known to be hard, and has a poor scenario complexity. Qiu, Varghese, and Suri (2001) analyzes two fundamental approaches; backtracking search and set pruning attempts. The researchers propose several new techniques to assist and enhance the two fundamental approaches, including: backtracking search using small memory utilization, a novel compression algorithm, pipe-lining the inquiry, and trading-off easily amongst backtracking and set pruning. The research quantifies the performance gains for every approach utilizing actual databases. The study demonstrates that on actual firewall databases, the schemes with the advancements are ideal in time and capacity.

Guillen, Sossa, and Estupiñán (2012) demonstrates how performance results between closed and open source routing approaches are vital parameters for network architects. The study breaks down execution in convergence time, throughput and delay between routing approaches in view of virtual software router (VSR) and routing approaches in view of proprietary hardware routers (PHR). The outcome demonstrates that VSR have better convergence times compared to hardware routers and the throughput performance is better on PHR. Waziri Jr, Mirzoev, and Shropshire (2014) looked into the comparison of control and hardware based filtering architectures in order to identify the most effective architecture, a control test with no firewall was conducted, followed by a hardware based packet filtering architecture. The study used HTTP packets generated by a load testing tool. The results focused mainly on endurance and the spike tests for the two architectures.

Sheth and Thakker (2011) did a comparative research that evaluates different types of firewall operation, the operational conditions and performance results of shortcomings in firewall operations. Moreover, the study analyzes reported issues with existing firewalls. Detailed analysis and correlation is done in terms of cost, security, operational ease, and execution of open source packet filter (PF) firewall, checkpoint SPLAT, and Cisco ASA in a testing environment with laboratory generated monitored and controlled traffic. Different throughputs and connection statistics were utilized as benchmark for performance comparison. The outcomes showed that Cisco ASA outperforms other firewalls in terms of performance. Checkpoint SPLAT and OpenBSD PF likewise gives sensibly competitive performances.

Conventional firewalls depend on topology restrictions and controlled network to implement traffic filtering. Firewalls can't filter unidentified packets, so all clients on the internal side are trusted. While this model has functioned admirably for little to medium size networks, networking advancements, for example, expanded availability, higher line speeds, extranets, and working from home undermine and make access control out of date. To address the issue of traditional firewalls, the idea of distributed firewalls has been proposed. In this plan, security policy is still centrally defined, however implementation is left to the individual endpoints. IPSec might be utilized to convey credentials that express parts of the overall network policy. On the other hand, the credentials might be obtained through out-of-band means. S. Ioannidis et al. (2000) presents the implementation and design of a distributed firewall based on KeyNote trust management system and OpenBSD[1] to specify, distribute and resolve policy.

Firewalls are important tools for securing private networks. Be that as it may, by just deploying firewalls, administrators are a long way from securing their networks. Bad configurations results in breaches and network vulnerabilities. Specifically, conflicting filtering rules leads to blocking legitimate traffic or allowing undesirable packets. Abbes, Bouhoula, and Rusinowitch (2008) shows another characterization strat-

[1]An open source UNIX operating system

egy to recognize conflicting access rules inside a firewall. The technique forms an arrangement of filtering rules that have a variable number of fields. A field has a scope of values, represented by an internal or variable length bit string that may cross with relating field scope of different rules. Keeping in mind the end goal is to recognize overlaps, the study sorted out the states of each filtering rule in a manner that can rapidly isolate colliding rules.

Lihua, Jianning, and Zhendong (2006) presents a static analysis tool for firewall modeling and analysis by regarding firewall configurations as important programs. The tool applies static analysis methods to check misconfiguration, for example, violation of policies, irregularities, and inefficiencies in individual firewalls and among distributed firewalls. Firewalls performs typical model checking of the firewall designs for all conceivable IP packets along every single conceivable data paths. Typical model checking is both sound and complete due to the limited way of firewall configurations. The tool is actualized by modeling firewall rules utilizing binary decision diagrams which have been utilized effectively as part of the hardware verification and model checking. The tool is utilized to reveal several real misconfiguration in networks, some of which have been confirmed and amended by network administrators.

Chomsiri and Pornavalai (2006) proposes a strategy to analyze the firewall rule-set or policy using relational algebra and a raining 2D-Box model. Rules analysis can find every one of the abnormalities in the firewall rule-set in the way that is typically utilized by numerous firewall devices, like, Cisco Access Control List, Iptables, IPchains, or Check Point Firewall-1. While the current analyzing strategies consider the peculiarities between any two rules in the firewall rule-set, researchers consider more than two rules together in the meantime to find the abnormality. In this way, it is conceivable to find the hidden anomalies in the firewall rule-set. Analysis results can be utilized with the proposed rules-combination technique displayed in the research to minimize the firewall rule without changing the policy. At the end, the research developed an application based on the proposed analyzing strategy. The application could help administrators examine and adjust a complex firewall policy with fewer mistakes.

Until recently, the causes of decreased effectiveness and restricted usage of new security frameworks have been the inadequate execution of hardware that executes access control, difficult analysis, and a configuration that conforms to corporate security policy requirements. Without the utilization of specific solutions that permit effective functioning of data security systems and their coordination with other network applications, a secured corporate network infrastructure is impossible to achieve. Zaborovsky and Titov (2009) issue is considered from three points of view: the decision of the distributed hardware platform to enhance firewall performance; the portrayal of security approach by method for an organization-based access control mode; and automating the process of firewall rules formation taking into account high-level depiction of access policy requirements.

Kayssi, Harik, Ferzli, and Fawaz (2000) presents a firewall plan for IP networks utilizing a field-programmable gate array (FPGA). The FPGA actualizes, accepts, or denies rules of the firewall. A hardware-based firewall offers the benefit of speed rate over a software firewall, notwithstanding direct interfacing with network devices, for example, an Ethernet. The research indicates how the rules are translated to VHDL and then implemented in hardware, and how the hardware is used to filter network traffic in a packet-by-packet method, or based on connecting information, with speeds of more than 500,000 packets per second (Kayssi et al., 2000).

Golnabi, Min, Khan, and Al-Shaer (2006) shows an arrangement of techniques and algorithms to examine and oversee firewall policy rules: (1) data mining techniques to deduce effective firewall policy rules by mining the traffic log based on its frequency; (2) filtering-rule speculation to decrease the quantity of approach rules by generalization: and (3) a procedure to recognize any obsolete rule and a set of couple predominant rules, to create another arrangement of efficient firewall policy rules. Anomaly detection based on mining uncovered numerous hidden abnormalities by analyzing the firewall policy rules, bringing about two new sorts of anomalies. As a result of these systems, network security administrators can consequently review and update the rules.

From every one of these studies, we understand how a firewall is the effective innovation of today's network security defense; we also understood how maintaining firewall rules is complex, error-prone, expensive and ineffective for large networks. These firewall rules are generally custom-designed and handwritten; as a result, they are in consistent need of tuning and approval, because of the dynamic way of network traffic, constant changing network environment, and its market demands.

2.3 Network Security/Function Virtualization

Networks are deployed to make computers more accessible to the outside world. Making computers more accessible to the outside world is a mixed blessing (Dieter, 1999). More interactions are possible, but so are unwelcome interactions. One may therefore wish to control how users are able to connect to a network system, how users on the network access data, and how data is protected when it travels through the network.

Networks are the communication infrastructure of data transmission between nodes in a distributed system. Data meant to be sent by an application in one node has to be prepared for transport, transmitted as a sequence of electronic or optical signals, reassembled, and presented to an application program at the receiver's end. Network protocols have to find a route from sender to receiver; they have to deal with the loss or corruption of data, and also with the loss of connection. It is a good practice to address these concerns one at a time with a layered architecture, application protocols at the top, and protocols that physically transmit bits of information to the bottom.

Network management protocols provide the necessary support so that the data generated by other protocols are efficiently delivered to the intended recipients. Management protocols, for example, may check the availability of intermediate nodes between sender and receiver, find optimal connections, or resolve logical network addresses to physical addresses. Other protocols are used to remotely configure network nodes, the software running on these nodes are becoming more and more complex. Hence, network security increasingly relies on securing management protocols and nodes in the network. The fact that network nodes are located in protected sites is no longer a guarantee for security (Dieter, 1999).

"Network security refers to any activity designed to protect a network, specifically, these activities protect the usability, reliability, integrity and safety of a network and data. Effective network security targets a variety of threats and stops them from entering or spreading on a network" (Cisco, 2014).

Regardless of whether a company is on the Internet or not, security measures must be applied to the network. These security measures may be as simple as requiring users to regularly change their passwords or may involve using the network operating system and third-party utilities to restrict access and enforce policies (Blacharski, 1998).

Issues in Security

Since the advent of computers and networking, different forms of cyber attacks have been in place. Common attack techniques are classified: Some attackers gain system knowledge or personal information, such as, spying and phishing. Others meddle with system designated functions, for example, virus infections, worms and Trojans. Also, others exhaust the system resources rendering rendering services unavailable. This can be brought about by denial of service (DoS) attack. Different types of network interruptions also exist, for example, land attacks, smurf attacks, password stealing, social engineering, use of viruses and worms, bugs & indirect accesses, verification & protocol failures, information leakage, bot-nets, eavesdropping, data modification, spoofing, sniffing, software & hardware misuse, TCP hijacking, teardrop attacks and so on (Bhavya, 2008). For the purpose of this study, we are going to focus only on the DoS form of attack.

Considerable research has been completed lately towards virtualization and cloud security. With server combination and desktop virtualization, essentially more activity stays inside the data center racks, prompting blind spots in network security appliances. Current network security devices, designed in view of scale-up standards, can't keep pace with the expanded bandwidth dispensed to the servers, and the expanding volume of threats at all layers of the network stack. Likewise, high versatile workloads and expanding intelligence in the virtual and hypervisor layer makes it progressively hard for static network devices to interlock with dynamic policy changes and on-the-fly re-purposing of resources to serve different workloads, applications, or clients.

The researchers Basak, Toshniwal, Maskalik, and Sequeira (2010) highlights another pattern in the industry to virtualize network security devices inside security virtual appliances (SVA's), which can then be placed on hosts, and other distributed security function for network flows across the cluster. The methodology replaces single choke-point based physical security devices like firewalls. IP address management, flow monitoring, and data leakage are monitored using a distributed virtual counterparts running on slices of x86, integrated with compute workloads, with the capacity to take advantage of traffic going through all virtual machines. Cloud computing can convey both software and hardware as on-demand assets and services over the internet (Huang & Yang, 2010). Without a doubt, one of the noteworthy concerns in cloud computing is security.

Wu, Ding, Winer, and Yao (2010) focused on security of virtual networks in virtualized environments. The research presents an outline of security issues in virtual machines. The issues that exist in a virtual network are additionally being discussed and analyzed based on Xen platform. The outcome displays an inventive virtual network structure planned to control the intercommunication among virtual machines incorporated to physical machines with higher security. With distributed systems becoming more prevalent in modern computing, there is a need to scale exponentially with their use and integration. Up to this point, the x86 architecture does not allowed traditional trap-and-emulate virtualization. x86 virtual machine monitors have rather utilized parallel interpretation of the guest kernel code. Be that as it may, both intel and AMD have now introduced architectural extensions to support classical virtualization. The research "comparison of software and hardware methods for x86 virtualization" compares a current software VMM and another VMM intended for emerging hardware support. Surprisingly, the hardware VMM frequently shows lower performance than the software VMM.

Adams and Agesen (2006) conducted a study of software and hardware techniques for x86 virtualization and studied architectural level events such as page table overhauls, context switches, and I/O; the study discovered their cost incomprehensibly distinctive among native software VMM, and hardware VMM execution. The research results demonstrates that the hardware support neglects to give a recognizable performance advantages for two primary reasons. First, it offers no support for MMU virtualization; second, it neglects to exist with existing software techniques for MMU virtualization. The study looks ahead to developing methods for addressing the MMU virtualization issue with regards to hardware-assisted virtualization.

Manohar (2013) surveyed virtualization techniques, types of hypervisors, and building private clouds with virtualization. It additionally examines the security of cloud computing and introduced the optical network as an access network and its devices in the data centers as energy efficient centers. Virtualization assumes a noteworthy part in helping organizations reduce cost, and in the mean time guarantees enhanced productivity, better utilization, and adaptability of existing hardware. Reuben (2007) presents a literature on different security issues inside virtualization technologies. The study mainly focuses on open security vulnerabilities that virtualization conveys to the cyber environment. The study focused on security issues that are unique for virtual machines and security threats that are common to all virtualization technologies available in the market. The study finishes up with a several discourse of a few security vulnerabilities in the virtualized environment.

2.4 Cyber Threats Implementation

Distributed Denial of Service attack (DDoS) is a noticeable threat to cloud and virtualized environments. Traditional defense approaches cannot be easily applied to the cloud environment because of their moderately low proficiency and large data space. Thus, (Dou, Chen, & Chen, 2013) exhibited a Confidence-Based Filtering strategy (CBF). The research method is deployed in two periods; the non-attack period, and the attack period. More specifically, legitimate packets are gathered amid the non-attack period for extracting attribute pairs to produce a nominal profile. The CBF technique is advanced by computing the score of a specific packet amid the attack time frame to decide whether to discard it. In conclusion, extensive simulations are conducted to assess the feasibility of the CBF method. The outcome demonstrates that CBF has a high scoring speed, small storage requirement and an acceptable filtering precision, making it suitable for real-time filtering in cloud environment.

Negi, Mishra, and Gupta (2013) proposes an enhanced CBF approach featuring a modification to the confidence-based filtering technique researched for the cloud computing environment. It took into account connection designs that mitigate DDoS attacks in the cloud. The modification presents additional bandwidth and tries to increase the processing speed of victim's server.

Farahmandian et al. (2013) reviewed and compared the existing methods used to mitigate DDoS attacks on cloud computing. SYN flooding attacks are an example of Distributed Denial of Service (DDoS) attack. Early detection is desirable however traditional passive detection strategies are done in the early stages because of their dependence on passive sniffing an attacking signature. Xiao, Chen, He, and Sha (2005) captures attacking signatures utilizing an active probing scheme that obtains the delay of routers by sending packets containing a unique Time-to-Live set for the

IP headers. The aftereffects of the test are utilized to perform SYN flooding detection, which is reliable and has minimal overhead. This methodology is more independent than different techniques that require the participation of network devices. Different tests demonstrate that this delay probing approach accurately recognizes half-open connections caused by SYN flooding attacks from those emerging from different causes at an early stage. The researchers Lonea, Popescu, and Tianfield (2013) focused on detecting DDoS attack in cloud computing environments. The proposed solution is to consolidate the evidence obtained from Intrusion Detection Systems (IDS) deployed in the virtual machines of the cloud systems with a data fusion methodology on the front end. In particular, when the attack appears, the VM-based IDS will yield alarms, which will be stored in the MySQL database set inside the Cloud Fusion Unit (CFU) of the front end server. The research propose a quantitative solution for analyzing alerts generated by the IDSs, utilizing the Dempsters Combination Theory (DST) operations in 3-valued rationale and Fault Tree Analysis (FTA) for the mentioned flooding attacks. At the last step, the solution utilizes the Dempsters combination rule to fuse evidence from multiple independent sources.

On the topic of DDoS detection, H. Wang, Zhang, and Shin (2002) proposes a simple and vigorous mechanism for detecting SYN flooding attacks. Rather than monitoring traffic flow at the front (like firewall or proxy) of a server, they identify the SYN flooding attacks of LEAF routers that connects end hosts to the Internet. The ease of the detection mechanism lies in the stateless and low computation overhead, which makes the detection mechanism itself immune to flooding attacks. The detection mechanism depends on the protocol behavior of TCP SYN-FIN (RST) pairs, and is an instance of successive change point detection. To make the detection mechanism insensitive to site and access pattern, a non-parametric cumulative sum method is applied, thus, making the detection component for the most part relevant and its deployment much less demanding. The efficiency of this detection component is validated by trace-driven simulations. The evaluation results show that the detection mechanism has short detection precision. Because of its closeness to the flooding sources, the component not only sets alarm upon detection of continuous SYN flooding attacks, additionally it uncovers the location of the flooding sources without resorting to expensive IP trace-back. There have been numerous proposals to shield against SYN flooding, and some require notable TCP changes. A few arrangements that endeavor to determine the TCP shortcoming are now publicly available. Ricciulli, Lincoln, and Kakkar (1999) subjectively analyzes these arrangements of TCP SYN flooding defense. The study refines the analysis of the random drop approach and drives a straightforward and general approach to enhance its performance. At last, the study presents (through both analytical and packet-level simulations) the efficacy of the random drop approach in a variety of operating conditions.

TCP-based flooding attacks are a typical type of Distributed Denial of Service (DDoS) attacks that abuse network resources and can result in serious threats to the Internet (Chen & Yeung, 2006). Incorporating IP spoofing-random, subnet, and fixed varieties makes it harder to shield against attacks. Subnet-spoofing is the most troublesome type of DDoS to counteract. Chen and Yeung (2006) proposed a simple and efficient method to detect and protect against TCP SYN flooding attacks under various IP spoofing types, including subnet spoofing. The method makes use of a capacity efficient data structure and change-point detection method to distinguish complete three-way TCP handshakes from incomplete ones. Simulation experiments reliably demonstrate that the strategy is both productive and compelling in protecting against TCP-based flooding attacks under various IP spoofing types.

Shin, Kim, and Jang (2005) proposes D-SAT: detecting SYN flooding attack by two-stage statistical approach. A basic and robust way to detect SYN flooding attacks is by monitoring the network activity. Rather than dealing with all ongoing traffic on the network, D-SAT only monitors SYN count and the proportion between SYN and other TCP packets. D-SAT identifies SYN flooding and discovers victims more precisely in its second stage. To make the recognition mechanism and detection more easy, D-SAT utilizes a cumulative sum methodology in statistical process control. It makes the recognition mechanism more applicable and easier to implement. D-SAT additionally uses AFM (Aggregation Flow Management) to identify victims quickly and precisely. The trace-driven simulation results shows that D-SAT system is reliable and easy to deploy and D-SAT also demonstrate that it recognizes SYN flooding precisely and finds attack within a short detection time.

Denial of Service

A denial of service (DoS) attack is an attack that clogs up so much resources (CPU, Memory etc) from the target system. This usually causes the system to crash, reboot, or generally deny any services to legitimate clients. DoS attacks are extremely normal; to be certain, every server can undoubtedly experience such an attack at any given time. In a situation where the attack is coordinated across many hijacked system (zombies) by a single attacker (master), the attack is referred to as DDoS (Carl et al., 2006). DDoS threats come in many varieties, some of which target the underlying server infrastructure. Others exploit vulnerabilities in applications and communication protocols. Unlike other kind of cyberattacks, DDoS are typically launched to establish a long-term foothold and hijack sensitive information. Denial of service assaults do not attempt to breach your security perimeter. Rather, they attempt to make your services, website, and servers unavailable to legitimate users. In some cases, however, DoS is also used as a smokescreen for other malicious activities, or to take down security appliances (e.g., web application firewalls).

Classification of Denial of Service

DoS attacks can be classified into five categories (Douligeris & Mitrokotsa, 2004). These categories are: *Network Device level, OS level, Application level, Data Flood, and Protocol Feature attack.*

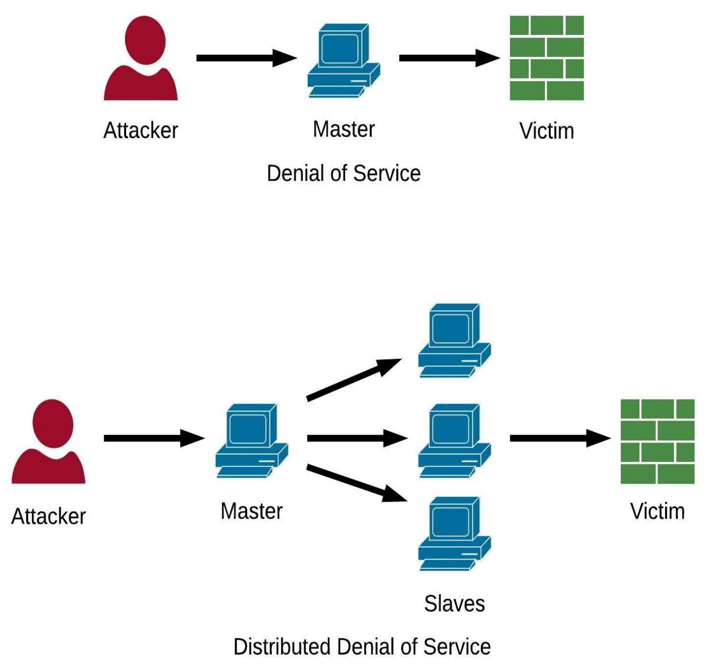


Fig. 2.1. DoS & DDoS Attack Types

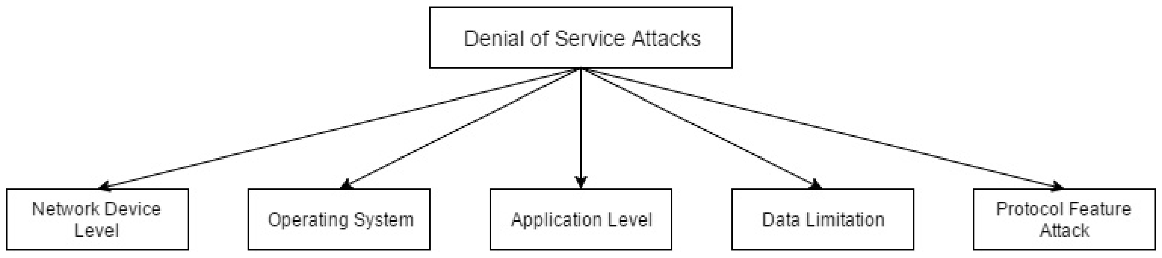


Fig. 2.2. Classification of DoS attacks

Network Layer Attacks - Layer 3 DoS

A DoS attack at the network level is caused either by exploiting a bug, software vulnerability, or exhausting the hardware resources of the network device (Waziri Jr & Shropshire, 2015).

Layer 3 attacks utilize specialized packets designed to cause resource intensive processing, slow response on target devices, or the disruption of TCP state information. These attacks influence issues in Layer 3 protocols and devices in order to cause significant disruption with much less attacker bandwidth than a volumetric attack. It is moderately simple; nonetheless, one needs to filter most Layer 3 attacks as they can be filtered with simple signatures and usually consume much less bandwidth than a pure volumetric attack (Waziri Jr & Shropshire, 2015).

Attack vectors in this category include UDP flood, SYN flood, NTP amplification, DNS amplification, and more. Any of these can be used to prevent access to servers, while also causing severe operational damages, such as account suspension and massive overage charges.

DDoS attacks are almost always high-traffic events, commonly measured in gigabits per second (Gbps) or packets per second (Pps). A large network layer assault can exceed 600 Gbps [2]; however, 20 to 40 Gbps are enough to completely shut down most network infrastructures.

Application Layer Attacks - Layer 7 DoS

DoS attacks at this level attempt to make a machine or service out of order either by exploiting particular bugs in network applications that are running on the target host or by utilizing such applications to deplete the resources of the victim. An example of this is the **finger-bomb**.

Layer 7 attacks exploit application layer commands that cause slow processing or crashes with the goal of disrupting the service of a targeted application. Layer 7 attacks normally target HTTP; either HTTP request that cause the web application to perform resource exhaustive processing or vulnerability in unpatched versions of the web servers. These attacks are considerably more hard to profile and filter at the network, and frequently require changes to web applications themselves. Basic Layer

[2]Arbor Network Report 2015

7 attacks include Slowlirs, R-U-Dead-Yet, and XDoS (zadjmool, 2013). Among other attack vectors in this category includes HTTP floods, slow attacks (e.g., Slowloris or RUDY), and DNS query flood attacks.

The size of application layer attacks is normally measured in request per second (Rps), with no more than 50 to 100 Rps required to exhaust most average sized websites (Davidowicz, 1999). In this dissertation, we implemented this type of attack.

Motivation & Causes

The purposes of DDoS attacks vary. A small percentage (on the positive side) of them can be accidental, caused by a badly configured system or as a demonstration to potential customers of DDoS protection solutions. However, on the negative side; there is often a personal intention behind the majority of these attacks. Some can be used as a diversion for attackers who want to try and steal information from certain systems, or for financial market manipulation or even fame, because hackers might want to boast that they managed to successfully attack a well known target or competitor. Moreover, online gaming, gambling, and social network related reasons are also motives behind such attacks. One of the biggest motivations behind DDoS attacks is “hacktivism”. Hacktivism is mainly driven by political and ideological disputes.

Severity

The biggest attack reported by a respondent in 2015 was 500 Gbps, with different respondents reporting attacks of 450 Gbps, 425 Gbps, and 337 Gbps. This proceeds with the pattern of significant development in the top-end size of DDoS attacks year-after-year. In 2014, 20 percent of respondents reported attacks of more than 50 Gbps. Interestingly in 2015 about one-fourth of the same respondents report attack sizes of more than 100 Gbps, stressing the scale of the DDoS problem. Clients remain the main focus in more than 66% of DDoS attacks. Once more, the extent of respondents seeing attacks focusing on cloud-based service has increased from 19% in 2013, to 29% in 2014, and to 33% in 2015.

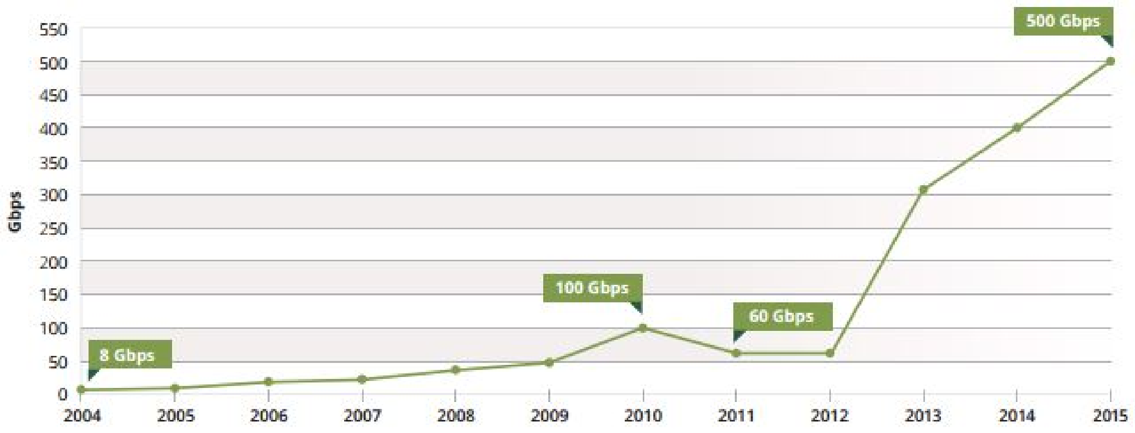


Fig. 2.3. DDoS Peak Attack Size Year after Year

Attackers have proceeded with the 2014 pattern of utilizing reflection/amplification strategies to exploit vulnerabilities in NTP, SSDP, and different protocols. Numerous respondents reported events at 200+ Gbps. This continues the trend of significant growth in the top-end size of DDoS attacks yearly.

Implementation

For DoS attacks to be implemented, an attack needs to be perpetrated towards the target. Depending on the type of victim, there are different forms of DoS attacks. For ethical reasons, this dissertation does not provide a guide on how to implement DoS attacks; however we presented the different forms of attacks as follows (Mirkovic & Reiher, 2004): *UDP Flood, ICMP (Ping) Flood, SYN Flood, Ping of Death, Slowloris, NTP Amplification, HTTP Flood, etc.*

HTTP Flood Attack

HTTP Flooding is an application layer attack. It overwhelms a target web server with a substantial amount of HTTP requests (Byers et al., 2004; Estevez-Tapiador, García-Teodoro, & Díaz-Verdejo, 2005), slowing or completely disrupting the regular web server traffic (Das, Sharma, & Bhattacharyya, 2011). In this dissertation, we implement a HTTP Flood attack. During this attack, an attacker exploits the HTTP GET or POST request sent when a HTTP client, like a web browser, talks to an application or server.

The attacker uses a botnet to send the victim's server a large amount of GET (pictures or scripts) or POST (file or forms) request with the expectation of overwhelming its resources and capabilities. The victims web server gets inundated, attempting to answer every request from the botnet, which drives it to allocate its maximum resources to handle the traffic. This prevents legitimate requests from reaching the server, causing a denial of service.

2.5 Commercial DDoS Defense Architectures

As a result of the various DDoS targets, types of attacks and severity, it is necessary to have methods to mitigate these attacks and help secure the availability of critical services. Nowadays, there are different popular commercial mitigation/protection providers that attempt to accomplish the problem posed by DDoS attacks. However, most commercial providers do not successfully achieve their stated goals. Commercial mitigation providers can provide a high detection success rate, but they tend to be very costly depending on the size of the attack traffic; also, they tend to have issues with clients regarding traffic being routed to the providers, which violates client privacy.

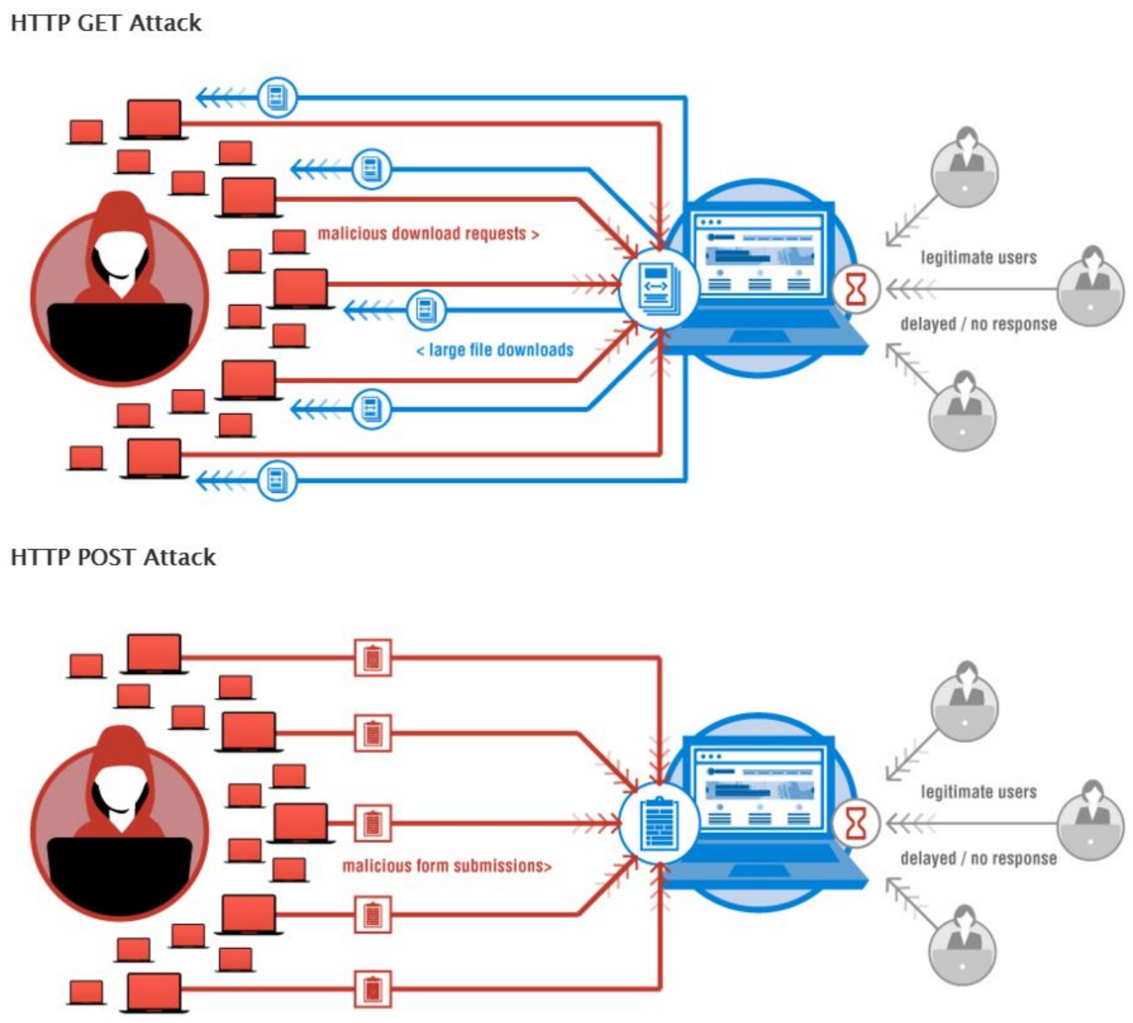


Fig. 2.4. HTTP GET & POST Flood Attack

2.5.1 Mitigation Approach

Discarding traffic altogether is not considered a mitigation practice, considering that legitimate traffic can also be dismissed. An Access Control List of firewalls is a choice for protection; however, this is only successful in filtering already known attacks by examining the protocols used. DDoS attacks are becoming more sophisticated, using valid protocols, and rendering the filtering process unsuccessful when it comes to SYN, SYN/ACK, and others forms of DDoS attacks. Another way of using routers as a form of protection is to use Unicast Reverse Path Forwarding (uRPF), which can be used to block IP addresses outside of the target's subnet. Nonetheless, if an attacker utilizes spoofed IP addresses from the same subnet, little to nothing can be done. Also, legitimate end-user traffic is blocked and the DDoS attack succeeds.

According to Cisco, another popular opinion is that firewalls and Intrusion Detection Systems (IDS) are inadequate forms of protection against DDoS attacks (Cisco, 2004). Firewalls are used inline and attackers target their low session handling abilities. Usually, they do not filter spoofed traffic and they can also be used to reject traffic from certain protocols. However, the attacking side can still use valid protocols during a DDoS attack. The same applies to IDS; they can provide excellent detection for Application Layer attacks but not against valid protocols. Moreover, as the term suggests, IDSs only function as a detection mechanism.

2.5.2 Mitigation Techniques

Contrary to privacy issues, commercial solutions have a high success rate in mitigation. Such companies offer different types of plans according to clients' needs and budgets. Services such as prevention, monitoring and traffic handling can be provided. There are mainly two different types of traffic handling correlated to the layers of attack; Layer 3/4 and Layer 7 solutions (Incapsula, 2014).

Layer 3/4 mitigation techniques are based on Border Gateway Protocol (BGP) IP address range swings. The target of the DDoS attack can decide to stop announcing their IP address range to the global Internet; in turn, that particular company announces it for them, so that they receive all of the traffic intended for the client, whether it is malicious or not. This operates in a distributed model with data centers across the world; traffic is then washed as it goes through special purpose built appliances to filter illegitimate traffic out with the use of specific algorithms. Once the traffic is washed it is rerouted back to the client (On-Ramping) over a Generic Routing Encapsulation (GRE) tunnel.

Layer 7 mitigation techniques also function in a distributed model. By having multiple data centers at different Internet Exchanges, clients can point the DNS entry of their websites to these companies who, in return, handle the requests where each packet is inspected. Based on the signatures, illegitimate traffic can be detected and discarded. Next, legitimate traffic is sent back to end-user browsers based on their geographical location.

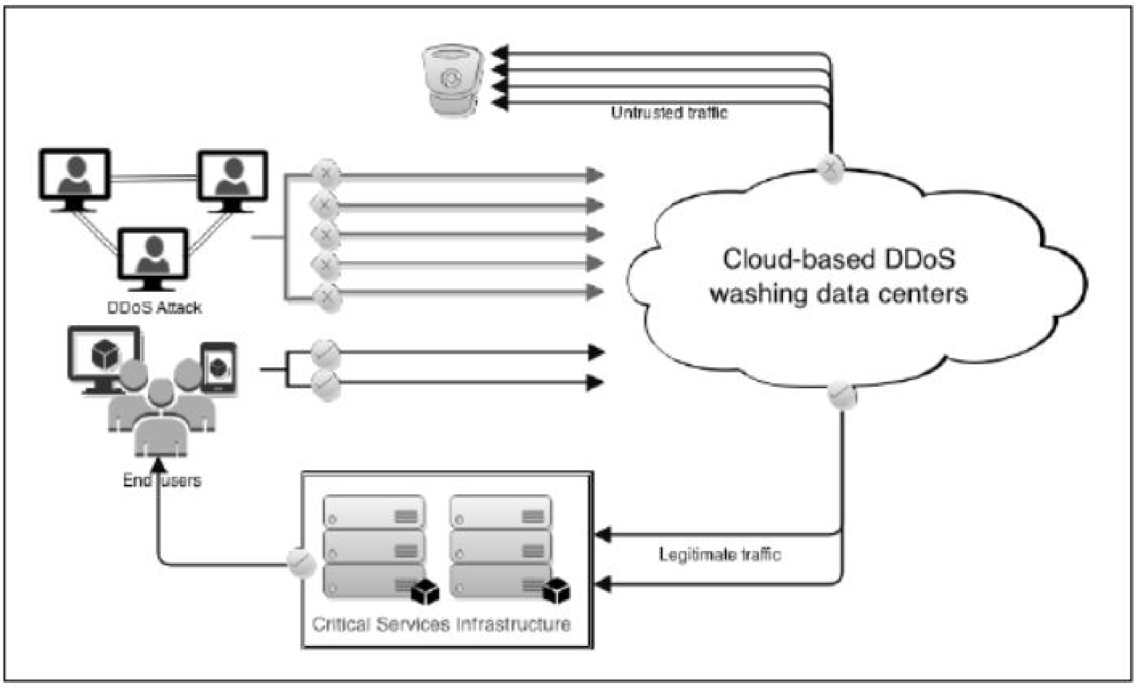

Untrusted traffic
DDoS Attack
Cloud-based DDoS washing data centers
End users
Legitimate traffic
Critical Services Infrastructure

Fig. 2.5. Commercial-Based DDoS Mitigation Technique

Due to the immense amount of available bandwidth, both legitimate and malicious traffic is accepted. The traffic is then washed using algorithms to examine which packet protocols are used. The DDoS traffic is discarded and the legitimate traffic is sent to the Critical Service Infrastructure and returned to the end-users.

2.5.3 Drawbacks of Commercial Solutions

Few commercial mitigation techniques prove to be efficient; many are imperfect, and also depending on the customer, they can be quite expensive (Verisign, 2014a). First and foremost, the algorithms used to monitor traffic are not flawless, which means that along with DDoS traffic, sometimes legitimate traffic is discarded. These mitigation solutions offer DDoS detection and contact the client when they detect a significant rise in their traffic to ask if they should take measures (Agarwal, Dawson, & Tryfonas, 2003). Up to thirty minutes can pass after the detection has provided results and the BCP swing of the IP range has taken place, during which the victim is under attack and unable to react (Verisign, 2014b).

Application Layer solutions have a different disadvantage. Because of their distributed model which can have a replica of the client's web service in any of their data centers, it is rather unsafe to use for services that implement SSL due to the fact that the Private Keys need to be shared. For small customers, this might not be an issue; however, when customers are financial and governmental organization, privacy issues are far more consequential. Recently a solution to this problem was developed (CloudFlare, n.d.) which tries to provide a mitigation solution without sharing Private Keys. Nevertheless, the matter of privacy still remains when bringing a third party into the equation. End-users of critical services trust that their financial and private data is handled by an entity that is stable and impermeable.

Lastly, these companies offer contracts based on bandwidth. If a customer, for example, chooses for contract of mitigating attacks of up to 40 Gbps and it is attacked by a larger DDoS attack than the terms of the contract, the prices of mitigation increase excessively. According to security firm Imperva, sixty percent of US companies experienced DDoS attacks during 2013. DDoS mitigation solutions can cost from $5,000 to over $100,000 US dollars per hour (Incapsula, n.d.). Furthermore, these figures do not include possible damages to credibility and customer satisfaction. The consequences of a DDoS attack can thus be disastrous for any organization or company.

2.5.4 Prevention & Response

The primary line of defense is to ensure DDoS doesn't take place. Hosts must be securely shielded from substantial and expert attack implants. There are in fact known signatures and filtering methods to recognize these attacks. Another method is monitoring network traffic for known attack messages sent amongst attackers and masters. On the active side, cyber-informants and cyber-spies can be used to intercept attack plans. For instance, Gibson clearly showed how he effectively spied on attack plans within a group of agents (Gibson, 2001).

This line of defense alone is clearly deficient. There are always don't care users and careless clients who leave their devices vulnerable to DDoS agent implants. Internet Service Providers (ISPs) and enterprise networks do not have motive to monitor for attack packets. Besides, spying on attack plans, such as the one presented in (Gibson, 2001) requires an in-depth knowledge of specific methods of launching DDoS attacks, which may also be changed later on, to avoid spying (Chang, 2002).

Attack source traceback and identification is normally an after-the-fact response to a DDoS attack. IP traceback refers to the problem, as well as the solution, of the actual source of any packet sent across the Internet without relying on the source information from the packet. There are mostly two approaches to deal with the IP traceback issue. One is for routers to record information about packets for later traceback request (Snoeren et al., 2001). Another is for routers to send extra information about the packets to the packet destinations via either the packets (Savage, Wetherall, Karlin, & Anderson, 2000) or another channel, such as ICMP messages.

However, it is in-feasible to utilize IP traceback to stop a continuous DDoS attack. First, current IP traceback solutions are not always able to trace packet origins (e.g., those behind firewalls and network address translators). Additionally, IP traceback is ineffective towards reflector attacks in which the attack packets originate from legitimate sources. Regardless of the fact that the attack sources can be effectively traced, preventing them from sending attack packets is another troublesome task, particularly when they are scattered in different autonomous systems (AS). Nevertheless, IP traceback could be very useful in recognizing the attacker and collecting evidence for post-attack law enforcement (Chang, 2002).

2.6 Flooding Mitigation

A Distributed Denial of Service (DDoS) attack utilizing Botnets became widely used in the Internet because of its efficiency and easy implementation. Zahid, Belmekki, and Mezrioui (2012) presents a new architecture for detecting DDoS/Brute forcing attack and destroying the botnet behind. The architecture stops DDoS attacks based on Botnet command and control and identifies the botmaster machine. The architecture is composed of DDoS attacks detection agents, spies and central agents that coordinate with each other during the attack trace-back process. The trace-back procedure depends on hacking techniques in order to infiltrate the Botnet and get information about the attacker and the bots utilized. The architecture is intended for Internet Service Providers (Zahid et al., 2012). Botnets are the predominant mechanisms for facilitating the distributed denial of service (DDoS) attacks on computer networks or applications. To date, Botnet-based DDoS attacks on the application layer are the latest and most problematic trend in network security threats. Botnet-based DDoS attacks in the application layer limit resources, shorten revenue, and yield client dissatisfaction.

DDoS attacks are among the most troublesome problems to resolve online, especially when the target is a Web Server. Alomari, Manickam, Gupta, Karuppayah, and Alfaris (2012) introduces a thorough research of Botnet-based DDoS attacks on application layers, particularly on the web server and decreased incidents of such attacks. Botnet-based DDoS attack incidents and significant revenue losses were also described. The study provides a better comprehension of the problem, current solution space, and future research scope to safeguard against such attacks.

These days, we are witnessing a significant increment in distributed denial-of-service (DDoS) attacks that flood victims from different sources. Intrusion detection and filtering are required mechanisms to combat against these attacks and properly secure networks. Nonetheless, the current detection technique for DDoS attacks works in isolation. Saad, Nait-Abdesselam, and Serhrouchni (2008) proposed a productive and collaborative architecture that takes into consideration the placement and cooperation of defense techniques to properly address major security challenges. The utilization of content-based, distributed hash-table algorithm allows improved scalability and load balancing of an entire system. The architecture has been executed on ISA entities using table protocol with a promising performance.

Harris, Konikoff, and Petersen (2013) gives a survey of the DDoS landscape and analyzes the application of the kill-chain concept to the DDoS threat. Utilizing the concept of detect, deny, disrupt, degrade and destroy. The paper investigates ways that this chain can be disrupted. An outline of the emerging DDoS threat is provided and considerations are offered for extra technology and research with the potential to significantly reduce the current DDoS threat.

Xuan, Chellappan, Wang, and Wang (2004) analyzed the secure overlay services architecture under intelligent DDoS attacks. They proposed a secure overlay service architecture to provide reliable communication amongst customers and a target under DDoS attacks. The SOS architecture utilizes an arrangement of overlay nodes arranged in three hierarchical layers that control access to the target. Even though the architecture is novel and functions admirably under basic over-flooding based attacks, it is observed to be vulnerable under more intelligent attacks. The architecture works by introducing more layering flexibility to the original architecture. To understand the impacts of the amount of layers, neighbors per node and the node distribution per layer under these two attack models, two intelligent DDoS attack models are defined and an analytical methodology was developed. The result clearly shows that performance is indeed sensitive to the design features that interact with each other to impact overall system performance.

Keromytis, Misra, and Rubenstein (2004) proposes a design called secure overlay services (SOS) that proactively prevents denial of service (DoS) attacks, targeted towards supporting emergency services or similar communications. The design utilizes a combination of secure overlay tunneling, routing via hashing and filtering. It minimize the likelihood of successful attacks by: 1. performing intensive filtering near the protected network edges and pushing the attack point perimeter into the core network, where routers can deal with the volume of attack traffic; and 2. introducing randomness and anonymity into the forwarding architecture, making it difficult for an attacker to target nodes along the way to a particular SOS-protected network. Utilizing basic analytical models, the research evaluates the possibility that an attacker can effectively launch a DoS attack against an SOS protected network. The analysis shows that such an architecture reduces the probability of a successful attack to a minimal probability. The performance measurements using a prototype implementation shows an increment in end-to-end latency by a factor of two for the general case, and an average recovery time of less than 10 seconds.

Beitollahi and Deconinck (2012) analyzed well-known countermeasures against distributed denial of service attacks. The study provided an in-depth analysis of each DDoS countermeasure and explained the strengths and challenges of each technique. The paper designed a countermeasure against the defense mechanism from the attackers perspective. The study is assumed to help potential victims choose appropriate countermeasures against DDoS attacks based on the methodology presented, as well as the requirements they need to deploy the techniques.

Tariq, Malik, Abdulrazak, and Hong (2011) researched the packet flood attack and presented a collaborative peer to peer defense mechanism for DDoS attacks. The proposed solution identifies the attack at the victim edge router and sends alert messages to its neighboring nodes which permits them to proactively defend themselves. Simulation results demonstrate the efficiency of the solution, with less false positives at the victim edge router and less damage to the network due to the proactive defense approach.

Tupakula and Varadharajan (2003) studied counteracting DDoS attacks in multiple ISP domains using routing arbiter architecture. They focus on preventing DDoS attacks in multiple ISP domains. Other methods used cluster analysis. K. Lee, Kim, Kwon, Han, and Kim (2008) proposes a strategy for proactive detection of DDoS attack by exploiting its architecture; the selection of handlers, agents, communication, compromise, and attack. They proposed DDoS attack detection method using cluster analysis. The study investigated the strategies of DDoS attack and then chose variables based on these features. After that, a cluster analysis for proactive detection of the attack was performed. The researchers experimented with a 2000 DARPA Intrusion Detection Scenario Specific Data Set in order to assess their methodology. The outcomes shows that each phase of the attack scenario is well partitioned and can detect precursors of a DDoS attack as well as the attack itself.

Garg and Chawla (2011) used data mining instead of cluster analysis. Garg and Chawla (2011) present various significant areas where data mining techniques seem to be a strong approach for detecting and preventing DDoS. Douligeris and Mitrokotsa (2004) presents a simple approach to handling DDoS problems by creating a classification of DDoS attacks and DDoS mechanisms. Each attack and defense system category are described and the advantages and disadvantages of each proposed scheme are outlined. The objective of the paper is to introduce some order into the current attack and defense mechanisms, so that a better understanding of DDoS attacks can be achieved; also advanced and more effective algorithms and procedures to combat these attacks may be created. Koutepas, Stamatelopoulos, and Maglaris (2004) introduces Distributed management architecture for cooperative detection and reaction to DDoS attacks. Koutepas et al. (2004) proposes a cooperative intrusion detection framework focused on mitigating DDoS attack by introducing distributed overlay early-warning network. The objective is to minimize detection and reaction times and automate response, involving as many networks as possible along the attack path, which can then be detected locally without trace-back procedures. The fundamental building block is the cooperative anti-DDoS entity and a modular software system deployed in each participating network domain that supports secure message exchanges and local responses tailored to individual site policies. The study explains the operation and deployment of prototype, provides a survey of the approaches against DDoS and compares the approaches to related work.

Pushback is a tool for mitigating distributed denial of service (DDoS) attacks. DDoS attacks are treated as a congested-control problem, but because most congestions are caused by malicious hosts not obeying traditional end-to-end congestion control, the issue must be taken care of by routers. Functionality is added to each router to detect and preferentially drop packets that most likely are attributed with an attack. Upstream routers are likewise configured to drop such packets (henceforth the term pushback) in order to transfer legitimate traffic. J. Ioannidis and Bellovin (2002) presents an architecture for pushback, its execution under FreeBSD, and suggestions for how such a system can be implemented in routers (J. Ioannidis & Bellovin, 2002). Ando, Miwa, Kadobayashi, and Shinoda (2008) presents a load balancing system for mitigating DDoS attacks using live migration of virtual machines. Ando et al. (2008) applies virtual machine monitor to modify the virtualized operating system, and afterward outlines the detailed countermeasure for DoS attacks utilizing live migration.

Honeypot is a trap used to communicate with potential attackers to divert, detect, or prevent such attacks and guarantee uninterrupted availability of service. Deshpande (2015) prevented distributed denial of service attacks using virtualized honeypots. Deshpande (2015) gives insight into the issues introduced by distributed denial of service attacks, current solutions that use honeypots, and how a mesh of virtualized honeypots can be used to mitigate distributed denial of service attacks. Srivatsa, Iyengar, Yin, and Liu (2008) mitigated application-level denial of service attacks in web server using a client-transparent approach. Srivatsa et al. (2008) proposes handling DoS attacks by using a twofold approach. First, the researcher performs admission control to limit the amount of concurrent clients served by the online service, invisible to unauthorized clients by hiding the port number on which the service accepts incoming request. Second, admission control carries out a congestion control which allows admitted clients to allocate more resources to other approved clients. Congestion control is achieved by setting a client's priority level in response to the client's requests in a way that incorporates application-level semantics. The article displays a detailed assessment of the proposed arrangement utilizing two sample applications, Apache HTTPD and the TPCW benchmark (running on Apache Tomcat and IBM DB2). The study demonstrates that the proposed solution results in low performance overhead and is resilient to DoS attacks.

The need to keep an attacker unmindful of attack mitigation efforts is a very important component of protection against denial of services (DoS) and distributed denial of services (DDoS) attacks, because it helps to dissuade attackers from changing their attack patterns. DDoS protection can be achieved in two parts. The first is a fake server that provides a service function or receives attack traffic as a substitute for a legitimate server. The second is a decoy network that restricts attack traffic to the peripherals of a network, or reroutes attack activity to fake servers.

Okada, Hazeyama, and Kadobayashi (2014) proposes the use of a two-stage map table expansion Locator/ID Separation Protocol (LISP) to understand the fake network. It explains and showed how LISP can be used to deploy an oblivious DDoS mitigation mechanism by adding an extension to the LISP Map Server. Together with fake servers, this approach can end DDoS activity on the ingress end of a LISP-enabled network. At last, the paper verified the effectiveness of the proposed mechanism through simulated DDoS attacks on a simple network topology. The test results demonstrate that the mechanism could be deployed within a few seconds, and the attack traffic can be terminated without incurring overhead on the MapServer.

Shameli-Sendi, Pourzandi, Fekih-Ahmed, and Cheriet (2015) uses taxonomy of distributed denial of service mitigation approaches for cloud computing to concentrate on how to mitigate DDoS attacks. It presents a new taxonomy of DDoS mitigation, then discusses the principle elements of existing DDoS mitigation approaches and clarifies their functionalities in the cloud environment. Afterwards, it indicates how the current DDoS systems fit into the network topology of the cloud. Finally, the survey paper presented some of these DDoS mechanisms in detail, and compares their behavior in the cloud. The goal is to show how these characteristics bring an original perspective into existing DDoS mechanisms, and give researchers new experiences into how to mitigate DDoS attacks in cloud computing. Bhardwaj, Subrahmanyam, Avasthi, and Sastry (2015) proposes three tier network architecture to mitigate DDoS attacks on hybrid cloud environments. They use a multi-tiered network design based on hybrid cloud solution that has premise solution acceptable to the organization's IT security and operations team, as well as public cloud infrastructure capable of handling large sized DDoS attacks targeted towards hybrid cloud servers.

Miao, Yu, and Jain (2014) highlights a few novel elements and advancement pattern among DDoS attacks: 1) Large-scale. These attacks have the volume of up to hundred gigabits per second against a single cloud service. 2) Diverse attacks. The attacks range from network-layer (e.g. SYN surge, UDP surge) to application-layer (e.g. HTTP GET, SQL infusion) with varied characteristics for volume, number of connections, and packet header signature (e.g., TCP flag, port). 3) Fast ramp-up rate. The attack traffic ramps up quickly and influences the target cloud service usually within a minute. In response to these challenges, the attack detection and mitigation system needs to: 1) have adequate capacity to handle attack volume: 2) support the detection of diverse range of attacks: and 3) have accurate and quick attack detection with low damage to legitimate traffic.

To identify attacks, cloud operators usually use commercial hardware devices, such as Firewalls, IDS and DDoS-protection tools in the network. There are three issues with these hardware boxes. First, these hardware devices cannot address overwhelming attacks in cloud scale. For instance, Firewall and IDS look at the states and detailed signature of packets. They cannot handle attacks with high volume. DDoS-protection appliances verify only significant traffic at the network-layer; they can address larger attack volumes, but are not capable of handling virtual scale attacks of hundred of gigabits per second traffic. Second, these hardware boxes present unfavorable cost and capacity trade-offs. For example, the DDoS-protection appliance typically costs up to a million dollars per box annually. Third, since these devices run vendor specific software, they limit how operators can configure them to handle the increasing diversity of attacks.

There are commercial attack mitigation services (CloudFlare, Prolexic etc.,) that redirect web service and enterprise traffic through a dedicate high-capacity network for attacks detection and mitigation. However, most clients do not want their traffic to be re-routed considering the private concerns (Miao et al., 2014).

To address this issue, one paper proposes another paradigm for attack-prevention-as-a-service that uses commodity VMs for attack detection and mitigation. It introduces the NIMBUS service, which combines the elasticity of cloud computing resources with the algorithm found in software-defined networks (SDN). NIMBUS scales resource usage with traffic requests, to handle diverse attack efficiently and without the exposure of private clients traffic (Miao et al., 2014).

2.7 Chapter Summary

Indispensable research and studies have been conducted regarding the Information Security field as a whole. Everyday new threats and defense mechanisms evolve, and research is being conducted to understand the problems and provide solutions. In this chapter, we reviewed several studies relating to this dissertation. The purpose of this related literature is to understand the current state of the problem we are trying to solve. To the best of our knowledge, research has yet to address the issue of network downtime as a result of firewall failures. However, a number of studies that addresses DoS attacks, network security and firewall configurations have been conducted. Based on these studies, we decided to implement the proposed study.

We also explained how commercial DDoS mitigation solutions are sophisticated techniques; but, depending on the attack and the target, they may not always be sufficient. One major draw-back of commercial-based solutions is the fact that they are also susceptible to attack. Attackers are finding ways to infiltrate DDoS mitigation companies. Once an attacker bypasses these commercial companies, it becomes much easier to gain access to a network. Also there is an issue of having third parties monitor network traffic; therefore, we conclude that the best DDoS mitigation technique is for an innovative solution.

3. TRADITIONAL ARCHITECTURE PERFORMANCE TEST

3.1 Introduction

The Internet is an exciting and informative place to browse and explore. It is the great frontier and grandiose achievement of mankind. In reality, the World Wide Web is merely a collection of routers and servers that make up the largest wide-area network (WAN) in recorded history. The collection of networking gears provides mail servers, websites, and other information storage and retrieval systems which are all connected to the Internet and accessible to every person connected. It has even been said that the Internet contains the collective institutional knowledge of mankind.

The rapid expansion of the internet has provided tremendous opportunities to access an unparalleled amount of data. An organization connects to the internet to gain access to information and to share information with the public; once a company connects its private network to the internet, that organizations private information becomes vulnerable to hackers. When private networks are connected to the internet, the risk are great. However, using some security measures, one can share public information and still protect private information. One of these measures is to install a firewall between the private network and the internet (Blacharski, 1998).

A firewall is a security device that sits on the edge of your Internet connection and functions as an Internet Border Security Officer. It constantly monitors all incoming and outgoing connection traffic (Ierace, Urrutia, & Bassett, 2005; Thomas & Stoddard, 2011).

The use of firewalls is no longer confined to servers, websites, or commercial companies. Even if you simply dial your ISP or use PPP (Point-to-Point protocol) to surf the internet, you simply cannot do so without a firewall.

In other words, a firewall acts as a shield to protect your system from untrusted, non-reliable systems connected to the Internet. Conceptually, it drives from the firewalls-barriers made of fire-resistant material-used in vehicles. A firewall on your PC, however, listens to all ports on your system for any attempts defined by set of rules. To phrase it more technically; a firewall is a piece of software, hardware, or both that allows only selected packets to pass from the internet to your private network or system. There are different types of firewalls and firewall generations. For the purpose of this dissertation, we focus on Hardware and Virtual based firewalls that belong to the Third Generation of firewalls.

Firewall Appliances (Hardware-Based)

These are firewalls that come hardened in a box. These types of firewalls provide services from a host attached to the internal network using a separate router. In this architecture, packet filtering provides the primary security, preventing people from going around proxy servers to make direct connections.

Host Firewalls (Virtual-Based)

These are firewalls that are installed on hosts themselves. These firewalls are installed just like an OS. They are mostly installed on servers. They are normally used in conjunction with other firewalls.

Third Generation Firewalls - SMLI

Stateful Multi-Layer Inspection (SMLI) represents a third generation of firewall technology. This new class of firewall can be applied internally and externally, over different protocol boundaries, and with numerous advanced functions. SMLI is similar to the application gateway model; it examines all seven layers of the OSI model. Instead of relying on a proxy, SMLI relies on a traffic screening algorithm optimized for high throughput. Each packet is examined and compared against known states of friendly packets. SMLI examines the entire packet (both address and application data). Because SMLI does not use a proxy, it overcomes the performance problems of the application gateway model.

The SMLI solution enables all applications to run natively over the firewall, because no proxies or modifications are necessary. The user also does not face additional passwords or validation procedures, so the solution is transparent to the end user (Blacharski, 1998).

Access Controls

All information that flows across the Internet uses TCP/IP. In turn, this information is sent in small pieces known as packets. In the early days of the internet, filtering based on packets was common; in many cases, routers in many networks still use packet filtering. The methods used to configure and deploy packet filters on Cisco ASA and routers is known as an access control list (ACL). There are two main types of ACLs: the standard ACL, which filters based on IP address, and extended ACLs, which look further into packet headers (Thomas & Stoddard, 2011). An access list is essentially a list of conditions that categorize packets. They can be really helpful when you need to exercise control over network traffic. An access list is the tool of choice for decision making in packet filtering (Waziri Jr, 2014).

One of the most common and easy to understand uses of the access list is filtering unwanted packets when implementing security policies. Access to the internet brings corporations the advantage of widespread access to share information. However, it also brings the risk of attack and unauthorized access. Access control is a critical part of security policy that must be implemented in the firewall (Panko, 2010).

Access control policy specifies what and who can enter or exit the corporate network. To maintain access control, the security administrator must have a clear picture of all services and applications available. Earlier packet filtering routers could not do this task because they were unaware of applications. Second generation firewalls, or application proxies, are application-aware, but consume a great deal of overhead. Later technologies use stateful inspection and other advanced techniques build on this application-layer awareness, while providing faster support for new services. Access control should not only address what can pass into the network, but should also allow the security admin to be able to the specify rules for what time each user can access which service. The two main types of access lists are as follows:

Standard Access Lists

This uses just the source IP address in an IP packet as the condition test. All choices depend on the source IP address. This implies standard access lists essentially allow or deny a whole suite of protocols. They don't recognize the numerous types of IP packets, for example, Web, Telnet, UDP, etc.

Extended Access Lists

Extended access lists can assess other fields in the layer 3 and 4 headers of an IP packet. They can evaluate source and destination IP addresses, the protocol field in the Network layer header, and the port number at the Transport layer header. This gives extended access lists the capacity to make more in-depth inspection and granular decisions when controlling traffic. More in-depth description about standard and extended access lists is available in Cisco books.

3.2 Problem Statement

Information security professionals find themselves working against misconceptions and popular opinions formed from incomplete data, for example, the possibility that internal network security can be secured simply by deploying a firewall. A firewall is a perimeter defense and it is not designed to combat the threat within. First and

Second Generation firewalls do not secure against malicious code issues like viruses and Trojan horses (Garfinkel, Spafford, & Schwartz, 2003; NSTISSC, 2000; Smith, 2015), although some are capable of scanning for telltale signs. Configuring packet-filtering rules has a tendency to be a complicated process, in the course of which errors can easily occur leading to holes in the defense. In addition, testing the configured rules has a tendency to be lengthy and a difficult process due to the shortcomings of current testing tools. Ordinary packet-filtering routers cannot enforce some security policies simply because the necessary information is unavailable.

Configuring a firewall can be troublesome when the goal is to guarantee maximum security and functionality. Fortunately, most decent firewalls now come with a reasonable defaults access list. Some say you only need a hardware firewall. The issue is that no firewall can prevent legitimate traffic from any source (wanted or unwanted). This is potentially particularly problematic if you have file or print sharing enabled, considering attackers out there are testing all the time for this exploitable back door into a computer system.

Because of the known issues with firewalls, which is not having the ability to configure access control that would mitigate DDoS at the same time allow legitimate traffic to pass through. Firewalls are prone to DDoS attacks. From our literature review, we know that firewall performance depends on its available resources. DDoS attack exhaust available resources from its target. We decided to find a way to mitigate DDoS attacks that target firewalls. First, we started by testing the performance of a hardware firewall to see how much it can endure.

A paper on this work has been published in the proceedings of IEEE SouthEastCon 2015 (Waziri Jr & Shropshire, 2015).

3.3 Traditional Architecture Overview

The traditional test architecture utilizes one firewall, as shown in Figure 3.1. The aim of this test was to find the endurance and spike results of a hardware firewall. A specialized load testing tool (JMeter) was configured in a master/slave configuration. The load testing tool was used to implement a HTTP flooding attack. The attack was implemented by generating heavy HTTP traffic which targets a web server. The hardware firewall sits inline between the attacker system and the web server, making all traffic pass through the firewall. Using the ACL[1] shown in Table 3.1, HTTP traffic is allowed access, and hence the firewall doesn't block any HTTP traffic. This is done to allow access to the web-server, considering that DDoS attacks are implemented using legitimate traffic.

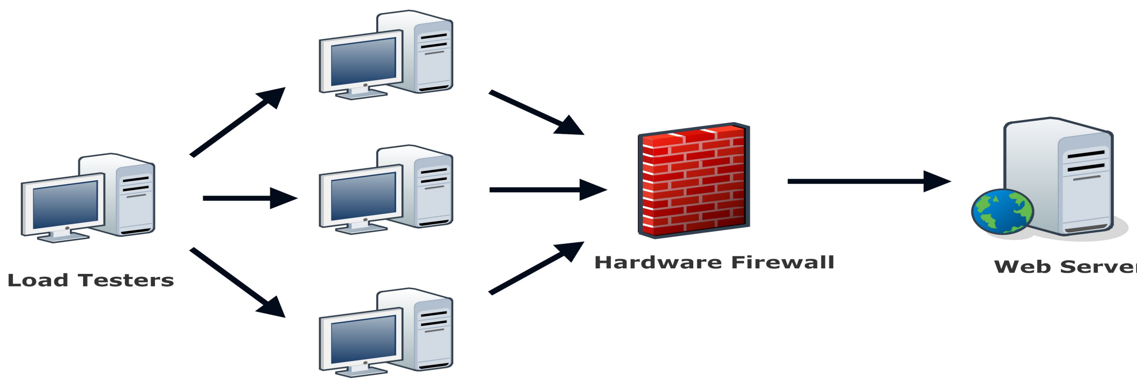


Fig. 3.1. Traditional Environment Architecture

3.4 Implementation & Configuration Setting

To set up the test-bed used in the traditional architecture, we used the following technologies and configurations:

Internet Protocol Version

We used the Internet Protocol version 4 (IPv4) to set up the IP addresses of all devices in the environment. The reason behind the choice of IPv4 and not the

[1]See Appendix A1

newer IPv6 is because IPv4 is the most widely used Internet Protocol (Gupta, 2010). That makes IPv4 more relevant to our study than IPv6. Also previous studies have shown that TCP-Flood, UDP-Flood, and ICMP-Flood types of DDoS attacks are the only types of attacks that change behavior between IPv4 and IPv6 and not HTTP-Flood (X. Yang, Ma, & Shi, 2007). The major difference between IPv6 and IPv4 are Optimized DHCP, IPSec, Larger address space, and Optimized mobility feature (Bade & Vanduhe, n.d.; Baker, Iturralde, Le Faucheur, & Davie, 2001; Nikander, Gurtov, & Henderson, 2010). We configured the architecture using a static IP address because it eliminates the plug and play networking provided by DHCP, thereby ensuring more security (Sitaraman, Mann, Dos Santos, Lou, & Bhasham, 2002). For a complete list of all the IP addresses used refer to Appendix A.

Application Protocol

We used Hypertext Transfer Protocol (HTTP) to flood the devices; it is the only legitimate protocol allowed in our Access Control configuration. We used HTTP because it is the protocol used to communicate between a client and a web-server (Casilari, Gonzblez, & Sandoval, 2001). A client submits a HTTP request message to the server, and the server respond with resources such as HTML files and other contents (Mah, 1997). Because we are using HTTP, we decided to implement a HTTP-Flood DDoS attack.

Packet Filters Access Control

For the hardware firewall in the traditional architecture, we blocked incoming traffic on eth0, the outside Ethernet port. We allowed only ICMP, HTTP, and TCP Port 80, because ICMP is used to Ping the network (Deering, 1991). HTTP is used to communicate with the web server (Mah, 1997) and TCP port 80 is the endpoint used to communicate HTTP used in WWW (Cole, 2011). Table 3.1 shows the access

Table 3.1
Access Control List of Traditional Packet Filter

Command	**List**
Allow	Config# access-list 110 permit TCP any eq 80 host 10.10.10.0/24
Allow	Config# access-list 110 permit ICMP any any
Allow	Config# access-list 110 permit HTTP any any
Deny	Config# access-list 110 deny udp any any eq 520
Deny	Config# access-list 110 deny ip any host X.X.X.X
Deny	Config# access-list 110 deny ospf any any
Deny	Config# access-list 110 deny host X.X.X.X
Deny	Config# access-list 110 deny tcp any any eq 21
Deny	Config# access-list 110 deny tcp any any eq 22
Deny	Config# access-list 110 deny tcp any any eq 25
Deny	Config# access-list 110 deny tcp any any eq 110
Deny	Config# access-list 110 deny tcp any any eq 143
Deny	Config# access-list 110 deny udp any any eq 135
Deny	Config# access-list 110 deny tcp any any eq 445
Deny	Config# access-list 110 deny tcp any any eq 1434
Deny	Config# access-list 110 deny tcp any any eq 4444
Deny	Config# access-list 110 deny tcp any any eq 4899
Deny	Config# access-list 110 deny udp any any eq 135
Config	access-group 110 in interface 'outside'
Apply	'outside' inbound traffic on eth0/0

control list applied to the inbound Ethernet port of the firewall. A complete list of denied and allowed access lists can be found in the Appendix.

Table 3.2
Hardware Server Resource

Resource	**Availability**
RAM	32GB
CPU	Intel Quad Core 4.66GHz
Storage	2TB

Deployed Hypervisor

A web server is the final destination of our HTTP-Flood attack. The web server runs as a virtual machine in a virtualized environment. And for every virtualized environment to run virtual machines, there must be a hypervisor that is being used to create and runs the virtual machine (ESXi, n.d.). In this dissertation we used a VMWare ESXi hypervisor. The hypervisor run on a dedicated hardware server with the configuration shown in Table 3.2.

Load Tester - JMeter

A load tester (JMeter) was used to generate the HTTP-Flood attack. We used a fully-featured web application test suite that can simulate a variety of real-life user behavior. Using the load tester, we were able to generate approx 3,250 request per second of HTTP traffic. The configuration used in the load tester can be found in Table 3.3 and 3.4. A full installation guide can be found in (JMeter, n.d.).

The configuration was composed of a sequence of set-up components (No. of Threads, Ramp-Up, and Loop Count) that determines how the load test will be simulated. Table 3.4 shows the configuration components and parameters used to generate the HTTP request packets.

Table 3.3
JMeter Configuration Components

No. of Threads(users)	No. of users JMeter will attempt to simulate.
Ramp-Up (in secs)	Duration of time JMeter will take to distribute the start of thread
Loop Count	No. of times to execute the test.

HTTP Request Defaults

We added a HTTP Request Defaults. The *HTTP Request Defaults* configuration element is used to set default values for HTTP Request in our test plan. This is useful because we want to send multiple HTTP requests to the server as part of our test. In the HTTP Request Defaults, under the Web Server section. We added the IP address of the web server **10.10.10.80**. Figure 3.3 shows the HTTP flood request packets.

Table 3.4
JMeter Configuration/Test Parameters

No. of Threads (users)	32500.
Ramp-Up (in secs)	10.
Loop Count	250
Generated Packet	3,250/sec.

Web Server

We used an Apache HTTP Server, because it is the most popular web server (Project, n.d.). We installed and configured the web server inside a Ubuntu Linux distribution. We configured the Apache web server by placing directives in plain text configuration files. These directives are separated between the following files and directories. The directives are:

1. **apache2.conf**: the main Apache2 configuration file. Contains settings that are global to Apache2.

2. **httpd.conf**: historically the main Apache2 configuration file, named after the HTTPd daemon. The file no longer exists. In older versions of Ubuntu the file might be present but empty, as all configuration options have been moved to the directories mentioned below.

3. **conf-available**: this directory contains available configuration files. All files that were previously in /etc/apache2/conf.d should be moved to /etc/apache2/conf-available.

4. **conf-enabled**: holds symlinks to the files in /etc/apache2/conf-available. When a configuration file is symlinked, it will be enabled the next time apache2 is restarted.

5. **envvars**: file where Apache2 environment variables are set.

6. **mods-available**: this directory contains configuration files to load modules and configure them. Not all modules have specific configuration files, however.

7. **mods-enabled**: holds symlinks to the files in /etc/apache2/mods-available. When a module configuration file is symlinked, it will be enabled the next time apache2 is restarted.

8. **ports.conf**: houses the directives that determine which TCP ports Apache2 is listening.

9. **sites-available**: this directory has configuration files for Apache2 Virtual Hosts. Virtual Hosts allow Apache2 to be configured for multiple sites that have separate configurations.

10. **sites-enabled**: similar to mods-enabled, sites-enabled contains symlinks to the /etc/apache2/sites-available directory. Similarly when a configuration file in sites-available is symlinked, the site configured will be active once Apache2 is restarted.

11. **magic**: instructions for determining MIME type based on the first few bytes of a file.

We installed the Apache web server inside a Ubuntu Linux distribution using this command:

```
sudo apt-get install apache2
```

And we configured the web server using these commands:

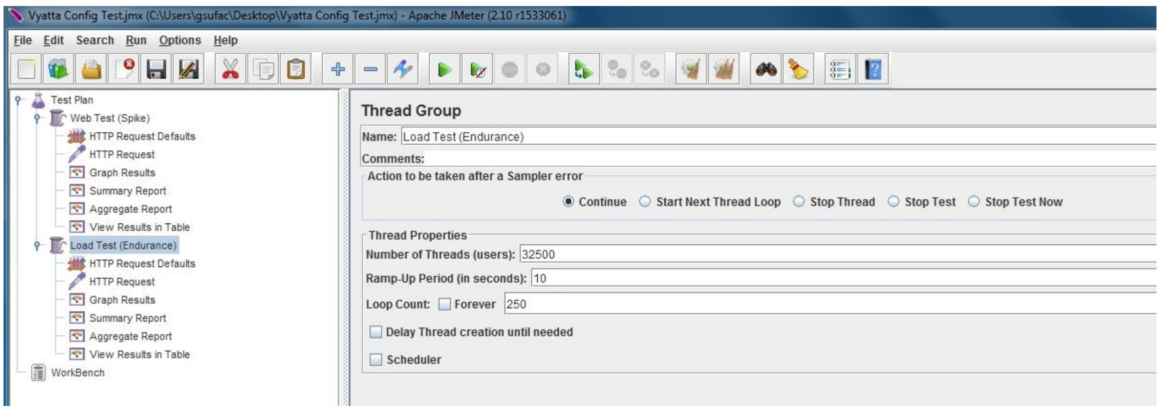
```
sudo /etc/init.d/apache2 start #start webserver
sudo /update-rc.d apache defaults  #runs webserver from autostart
gksu gedit /etc/apache2/sites-available/site1 #this enable the .htaccess file
sudo /etc/init.d/apache2 restart #this restarts apache
```

3.5 Implementing the Traditional Test

In order to implement the test, the hardware firewall was configured to allow HTTP traffic, and the load testing tool (Jmeter) - an open source application, which is a 100% pure Java application designed to load test functional behavior - measures performance and test web applications[2] used to generate HTTP traffic. The traffic was targeted towards the firewall, and configured in a master/slave configuration. Three computers were serving as Masters and 59 as slave. The “Number of Threads” (threads are used to simulate concurrent connections to your server application) is set to 32500. Each thread will execute the test plan in its entirety, completely independently of other test threads. The “Ramp-Up Period” (ramp-up period tells JMeter

[2]http://jmeter.apache.org/

how long to take to “ramp-up” to the full number of threads chosen. If 10 threads are used, and the ramp-up period is 100 seconds; JMeter will take 100 seconds to get all 10 threads up and running). Figure 3.2 depicts the load tester configuration settings.


Vyatta Config Test.jmx (C:\Users\gsufac\Desktop\Vyatta Config Test.jmx) - Apache JMeter (2.10 r1533061)
File Edit Search Run Options Help

Test Plan
- Web Test (Spike)
  - HTTP Request Defaults
  - HTTP Request
  - Graph Results
  - Summary Report
  - Aggregate Report
  - View Results in Table
- Load Test (Endurance)
  - HTTP Request Defaults
  - HTTP Request
  - Graph Results
  - Summary Report
  - Aggregate Report
  - View Results in Table

WorkBench

Thread Group

Name: Load Test (Endurance)

Comments:

Action to be taken after a Sampler error

[✓ Continue] Start Next Thread Loop Stop Thread Stop Test Stop Test Now

Thread Properties

Number of Threads (users): 32500

Ramp-Up Period (in seconds): 10

Loop Count: ☐ Forever 250

☐ Delay Thread creation until needed

☐ Scheduler

Fig. 3.2. JMeter configuration Settings

For the attack to be implemented, the load tester (JMeter) connects to our Apache web server, and establishes a TCP connection using a three-way handshake:

```
JMeter sends a SYN packet to Apache web server.
    Apache web server sends a SYN ACK packet to JMeter.
    JMeter sends an ACK packet to Apache web server.
```

HTTP Flood involves opening up a valid TCP connection with Apache Web server, and then sending a request.

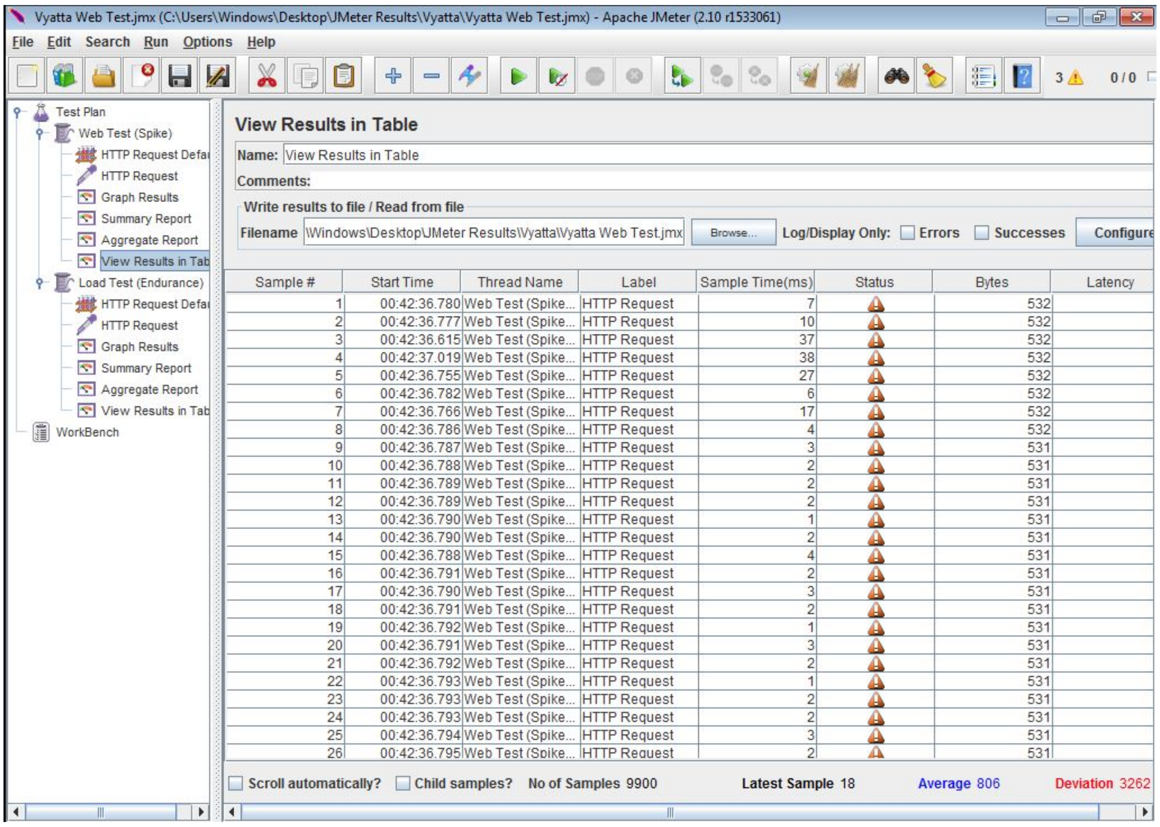
Immediately after the TCP connection is opened between JMeter and Apache web server, allowing free communication between the two, JMeter starts sending HTTP GET requests to the Apache web server using the default parser.

```
org.apache.jmeter.protocol.http.parser.LargeBasedHtmlParser.
```

Part of HTTP-GET Flood request sent by JMeter was captured using Wireshark:

```
10.10.10.22:42728 -> 10.10.10.80:80 [AP] GET/HTTP/2 Host:10.10.10.79 index.php
10.10.10.22:40962 -> 10.10.10.80:80 [AP] GET/HTTP/2 Host:10.10.10.79 index.php
10.10.10.22:51486 -> 10.10.10.80:80 [AP] GET/HTTP/2 Host:10.10.10.79 index.php
10.10.10.22:55300 -> 10.10.10.80:80 [AP] GET/HTTP/2 Host:10.10.10.79 index.php
10.10.10.22:56396 -> 10.10.10.80:80 [AP] GET/HTTP/2 Host:10.10.10.79 index.php
```

The HTTP GET Flood legitimately downloads a picture (pic1.jpg) from the Apache Web server's index page. It establishes a full session and actually receives all the data from the web page. Because our traditional firewall sits inline between the load tester (client) and the web server, the firewall has to process every GET request that passes through. Approximately 3,250 request were sent per second, ultimately over flooding the firewall because it couldn't process every request sent. Figure 3.3 shows JMeter sending the HTTP request packets.



View Results in Table

Name: View Results in Table

Comments:

Write results to file / Read from file

Filename \Windows\Desktop\JMeter Results\Vyatta\Vyatta Web Test.jmx

Log/Display Only: ☐ Errors ☐ Successes

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Latency
1	00:42:36.780	Web Test (Spike...	HTTP Request	7	⚠	532	
2	00:42:36.777	Web Test (Spike...	HTTP Request	10	⚠	532	
3	00:42:36.615	Web Test (Spike...	HTTP Request	37	⚠	532	
4	00:42:37.019	Web Test (Spike...	HTTP Request	38	⚠	532	
5	00:42:36.755	Web Test (Spike...	HTTP Request	27	⚠	532	
6	00:42:36.782	Web Test (Spike...	HTTP Request	6	⚠	532	
7	00:42:36.766	Web Test (Spike...	HTTP Request	17	⚠	532	
8	00:42:36.786	Web Test (Spike...	HTTP Request	4	⚠	532	
9	00:42:36.787	Web Test (Spike...	HTTP Request	3	⚠	531	
10	00:42:36.788	Web Test (Spike...	HTTP Request	2	⚠	531	
11	00:42:36.789	Web Test (Spike...	HTTP Request	2	⚠	531	
12	00:42:36.789	Web Test (Spike...	HTTP Request	2	⚠	531	
13	00:42:36.790	Web Test (Spike...	HTTP Request	1	⚠	531	
14	00:42:36.790	Web Test (Spike...	HTTP Request	2	⚠	531	
15	00:42:36.788	Web Test (Spike...	HTTP Request	4	⚠	531	
16	00:42:36.791	Web Test (Spike...	HTTP Request	2	⚠	531	
17	00:42:36.790	Web Test (Spike...	HTTP Request	3	⚠	531	
18	00:42:36.791	Web Test (Spike...	HTTP Request	2	⚠	531	
19	00:42:36.792	Web Test (Spike...	HTTP Request	1	⚠	531	
20	00:42:36.791	Web Test (Spike...	HTTP Request	3	⚠	531	
21	00:42:36.792	Web Test (Spike...	HTTP Request	2	⚠	531	
22	00:42:36.793	Web Test (Spike...	HTTP Request	1	⚠	531	
23	00:42:36.793	Web Test (Spike...	HTTP Request	2	⚠	531	
24	00:42:36.793	Web Test (Spike...	HTTP Request	2	⚠	531	
25	00:42:36.794	Web Test (Spike...	HTTP Request	3	⚠	531	
26	00:42:36.795	Web Test (Spike...	HTTP Request	2	⚠	531	

☐ Scroll automatically? ☐ Child samples? No of Samples 9900 Latest Sample 18 Average 806 Deviation 3262

Fig. 3.3. HTTP GET Flood Request

3.6 Evaluation & Analysis

As shown in Figure 3.4, the traditional test which uses real HTTP traffic generated from the load tester configured in Master/Slave on 62 computers. The packet drop was at 76.5%, the CPU Utilization was at 67%, and the memory utilization at 81%. This is the level at which the hardware firewall became non-responsive, the hardware firewall was maxed out, which results in its failure with a downtime of about 30 seconds, and about 56 seconds recovery time, as shown in Table 3.5. As the attack load increases, the CPU usage of the hardware firewall increases. Table 3.8 shows the CPU utilization based on the time required. In addition to the CPU utilization, the memory utilization also reaches a maximum of 81%. The packet drop was high based on the received and transmitted packets, as shown in Table 3.6.

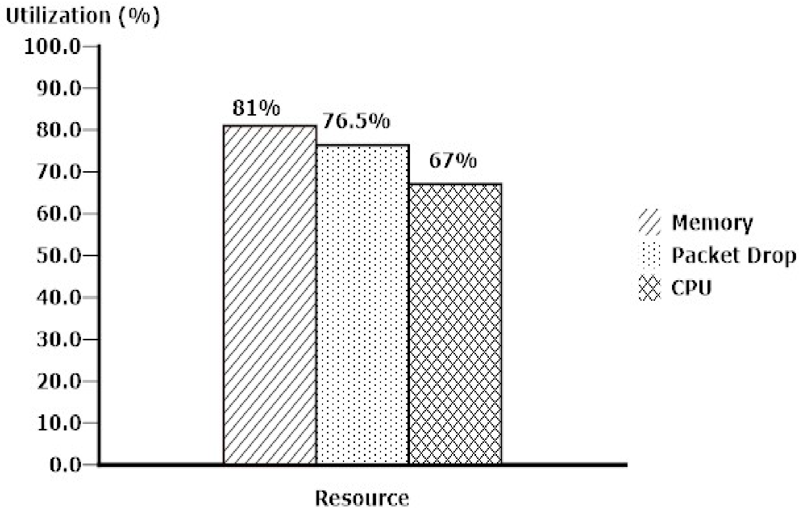


Fig. 3.4. The Traditional Architecture Results

3.6.1 Downtime & Recovery Period Analysis

Table 3.5 shows the downtime and recovery time of the hardware packet filter following its failure. The hardware firewall was non-responsive for about 30 seconds before it became responsive. It took about 56 seconds to recover to its working state.

The recovery period followed the test (HTTP flood attack) suspension. Part of the longer duration in the recovery period was as a result of the boot sequence.

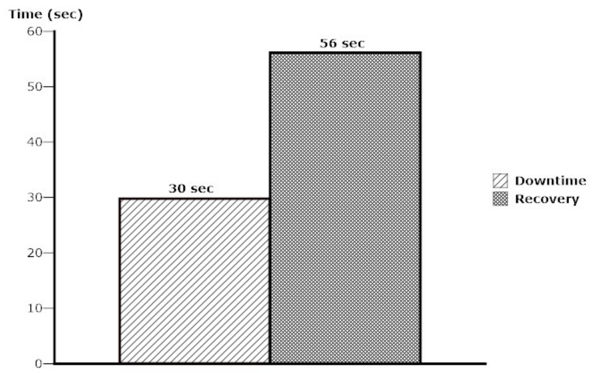


Fig. 3.5. Recovery & Downtime Periods

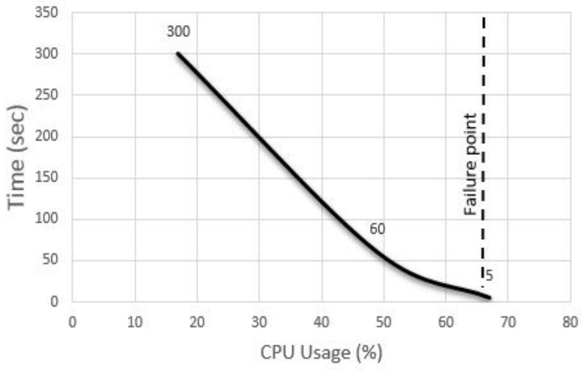


Fig. 3.6. CPU Usage over Time

Table 3.5
Time Period Analysis

Action	**Time (secs)**
Downtime	30
Recovery Time	56

Table 3.6
Packet Transmission Analysis

Flow Direction	**Outside - eth0**	**Inside - eth1**
Received	39357350	11877268
Transmitted	10256915	25143799

Table 3.7
Memory Utilization Analysis - 8GB

	Usage in %
Free Memory	19%
Used Memory	81%

Table 3.8
CPU Utilization & Time Required Analysis

Time (secs)	**CPU Usage**
300secs	17%
60secs	49%
5secs	67%

3.7 Defined Thresholds

Due to the fact that the hardware firewall fails under heavy traffic as presented, we decided to define new set of thresholds, different than those set by the firewall vendor. When these thresholds are reached, the device is rendered incapable of performing as required; hence, an action must be taken to avoid failure. The newly defined threshold based on the results collected are presented in Table 3.9.

Table 3.9
Traditional Packet Filter: Newly Defined Threshold

Packet Drop	$\geq 10\%$
Memory Utilization	$\geq 80\%$
CPU Utilization	$\geq 65\%$

3.8 Chapter Summary

This chapter highlights our test-bed implementation, and hardware firewall outcomes. From the result, we defined new thresholds to be used in subsequent chapter.

4. VIRTUALIZED ARCHITECTURE PERFORMANCE TEST

4.1 Introduction

Cloud computing represents one of the most significant shifts in information technology we are likely to see in our lifetimes. Customers are both excited and nervous about the prospects of Cloud Computing. They are excited by the opportunities to reduce capital costs, divest infrastructure management, and focus on core competencies. Most of all, they are excited by the agility offered by the on-demand provisioning of computing and the ability to align information technology with business strategies and needs more readily. However, customers are also very concerned about the risks of Cloud Computing if it is not properly secured. And the loss of direct control over systems for which they are nonetheless accountable.

Security controls in cloud computing are, for the most part, no different than security controls in any IT environment. However, because of the cloud service models, operational models, and the technologies used to enable cloud services, cloud computing may present different risks than traditional IT solutions to an organization.

Some of the security threats to cloud computing outlined by Cloud Security Alliance (CSA) are: Abuse and nefarious use of Cloud computing, Insecure API's, Malicious Insider, Shared Technology Vulnerabilities, Data Loss/Leakage, Account, Service & Traffic Hijacking, Unknown risk profile & Hypercall Threats (Alliance, 2011). NIST defined Cloud Computing as:

"*A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction*" (Mell & Grance, 2011).

In a nutshell, cloud computing is a way of separating an application from the operating system and hardware. The Cloud Security Alliance (CSA) defined cloud computing as:

"*an evolving term that describes the development of many existing technologies and approaches to computing into something different. Cloud separates application and information resources from the underlying infrastructure, and mechanisms used to deliver them*" (Alliance, 2011).

4.1.1 Virtualized Environments

Cloud Computing services are usually backed by large-scale data centers composed of thousands of computers. Such data centers are built to serve many users and host many disparate applications. For this purpose, hardware virtualization can be considered as a perfect fit to overcome most operational issues of data center building maintenance (Buyya et al., 2010). Virtualization allows running multiple operating systems and software stacks on a single physical platform. Figure 4.1 shows a software layer, the hypervisor also known as a virtual machine monitor (VMM), which mediates access to the physical hardware, presenting each guest operating system (VM) a set of virtual platform interfaces.

Virtualization has been a key enabling technology for the evolution of cloud computing in its current form (Irvine, Robin, et al., 2000; Popek & Goldberg, 1974). In particular, a hardware virtualization has enabled IaaS providers to efficiently use the available hardware resources in order to provide computing and storage services to their clients.

Type I VMM

Type I VMM, also known as a bare-metal, runs on a bare machine. It is an operating system with virtualization mechanisms. A type I VMM runs directly on

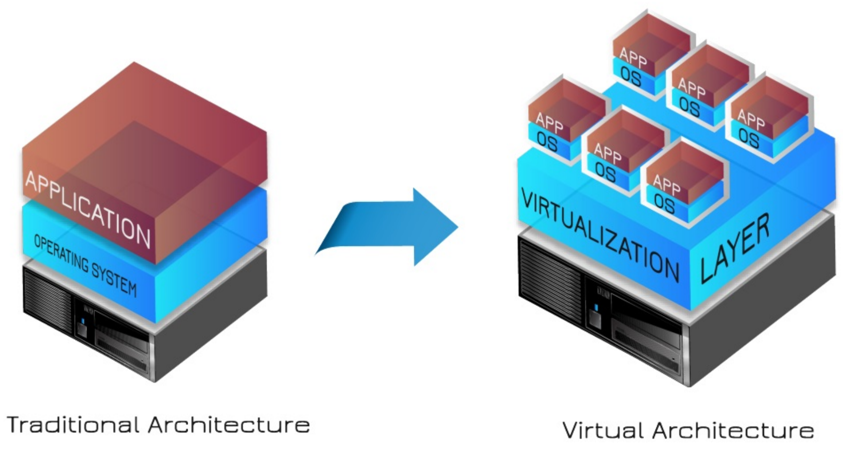


Fig. 4.1. Traditional vs. Virtualized Environments.

the machine hardware. It is an operating system or kernel that has mechanisms to support virtual machines. It performs scheduling and resources allocation for all virtual machines in the system and requires drivers for hardware peripherals.

4.1.2 Network Functions Virtualization

Network function virtualization is an initiative to virtualize network services being carried out by proprietary, dedicated hardware. NFV decreases the proprietary hardware needed to launch and operate network services [1]. NFV allows network operators to instantiate middleboxes in virtual machines and place those VMs at arbitrary locations in the network (ESTI, n.d.). Current approaches to NFV still treat middleboxes as monolithic entities, and do not explore how the constituent components of a middlebox might be decomposed into smaller modules (Anwer et al., 2015).

Because of the various partners with clashing objectives and strategies, changes to the current network designs are presently constrained to basic incremental additions; deployment of any new radically different technology is next to impossible. To battle off this solidification, network virtualization has been portrayed as a diversifying

[1]TechTarget - NFV defined

attribute to the future inner-networking paradigm. By introducing a plurality of heterogenous network architectures cohabiting on a shared physical substrate, network virtualization promotes innovations and diversified applications. A paper surveyed the existing technologies and wide-array of past and state-of-the-art projects on network virtualization followed by a discussion of major changes in the area (Chowdhury & Boutaba, 2010).

Network function virtualization (NFV) has drawn significant attention from both industry and academia as an important shift in telecommunication service provisioning. By decoupling network functions (NFs) from the physical devices on which they run, NFV has the potential to lead to significant reductions in operating expenses (OPEX) and capital expenses (CAPEX) that facilitate the deployment of new services with increased agility and faster time-to-value. The NFV paradigm is still in its infancy and there is a large spectrum of opportunities for the research community to develop new architectures, systems, and applications, to evaluate alternatives and trade-offs in developing technologies for its successful deployment. After discussing NFV and its relationship with complementary fields of software-defined networking (SDN) and cloud computing. A study surveyed the state-of-the-art NFV, and identify promising research directions in the area. The study also presents an overview of key NFV projects, standardization efforts, early implementations, use cases, and commercial products (Mijumbi et al., 2015).

Middlebox hardware appliances are known to come with a number of problems, such as being costly, difficult to manage, and inflexible in their functionality. NFV has alleviated all such problems and due to NFV's flexibility, several platforms are in place. A group of researchers introduced ClickOS a high performance, virtualized software middlebox platform. ClickOS virtual machines are small (5MB), boot quickly (20 milliseconds), add little delay (45 microseconds), and over one hundred hosts can concurrently run while saturating a 10Gb pipe on a commodity server. The study implements a wide range of middleboxes, including a firewall, a carrier-grade NAT, and a load balancer to show that ClickOS can handle packets in the millions per second (Martins et al., 2014). Others presented EmPOWER, an experiment test-bed which aims at offering an open platform on top of which novel concepts can be tested at scale (Riggio, Rasheed, & Granelli, 2013).

Another article presents the analysis, design, and first implementation of the routing function in a virtualized manner. Considering the current co-existence of IPv4 and IPv6 and the possibilities brought into the arena by OpenFlow-enabled infrastructures, the article describes the design of the virtualized routing protocol, its enabled simple management and signaling messages overhead avoidance in the control plane level, and the different scenarios considered to validate the virtualized function. In essence, the study describes the first implementation of the functional NFV concept through the virtualization of the routing function over an OpenFlow network. The different scenarios validated in the article are used to demonstrate the applicability of the NFV-powered implementation proposed into actual production environments (Batalle, Ferrer Riera, Escalona, & Garcia-Espin, 2013). Another study presents a measurement to characterize the impact of virtualization on the networking performance of the Amazon Elastic Cloud Computing (EC2) data center. The study measures the processor sharing, packet delay, TCP/UDP throughput, and packet loss among Amazon EC2 virtual machines. The results show that although the data center network is lightly utilized, virtualization can still cause significant throughput instability and abnormal delay variations. The study concludes with the implications of its findings on several classes of applications (G. Wang & Ng, 2010).

4.2 Problem Statement

The study proves that virtualization can result in overhead due to decreased performance (Sahoo, Mohapatra, & Lath, 2010). This happens because performance is often being compromised due to flexibility. Virtual machines have the capability of sharing resources through resource management (Beloglazov & Buyya, 2010). This works when one virtual machine borrows resources from another idle virtual machine.

This resource sharing capability of virtual machines comes with security issues when a running virtual machine borrows all of the available resources, making idle virtual machines completely disengage without the capability of running. Other associated problems identified with resource sharing are hypercall and hyperthreats. The running virtual machine keeps on requesting resources until there is none available. Hence all other functions are rendered ineffective (Shropshire, 2015). However, with these known issues, multiple studies prove that when network functions are virtualized, they perform better than traditional hardware devices (Guillen et al., 2012; Qiu et al., 2001; Sheth & Thakker, 2011; Waziri Jr et al., 2014; Waziri Jr & Shropshire, 2015).

Single point of failure (SPOF) is always an issue. Even though virtual machines are decoupled from the hardware, VM's are still dependent on the hardware running the hypervisor (server). Failure in the hardware automatically results in a failure of all virtual machines (Menascé, 2005; Pfaff et al., 2009; Sahoo et al., 2010).

In this dissertation, we tested both hardware and virtual firewalls; we also found that virtual packet filters to be more reliable under stress. However, because a hardware firewall is used to protect the underlying hardware (server) used by the hypervisor, whatever affects the traditional firewall affects the virtualized environment. To find the threshold of the virtual firewall, we implemented the same test carried out on the traditional firewall, using a different load testing tool.

4.3 Virtual Architecture Overview

Similar to the traditional architecture, the aim of the virtual test was to find the weakness and limitations of the virtual firewall. The virtual firewall was installed as a VM in a virtualized environment and connected inline prior to the web server. However, in the case of the virtual test, a different load testing tool[2] was used to generate the HTTP flood packets. The reason for using a different testing tool was

[2]High Orbit Ion Cannon

the limitation of resources imposed by the virtual environment. Figure 4.2 depicts the virtual architecture. The ACL[3] used in the virtual firewall is similar to that of the hardware firewall allowing HTTP traffic, ICMP, and TCP Port 80, while blocking all other incoming connections.

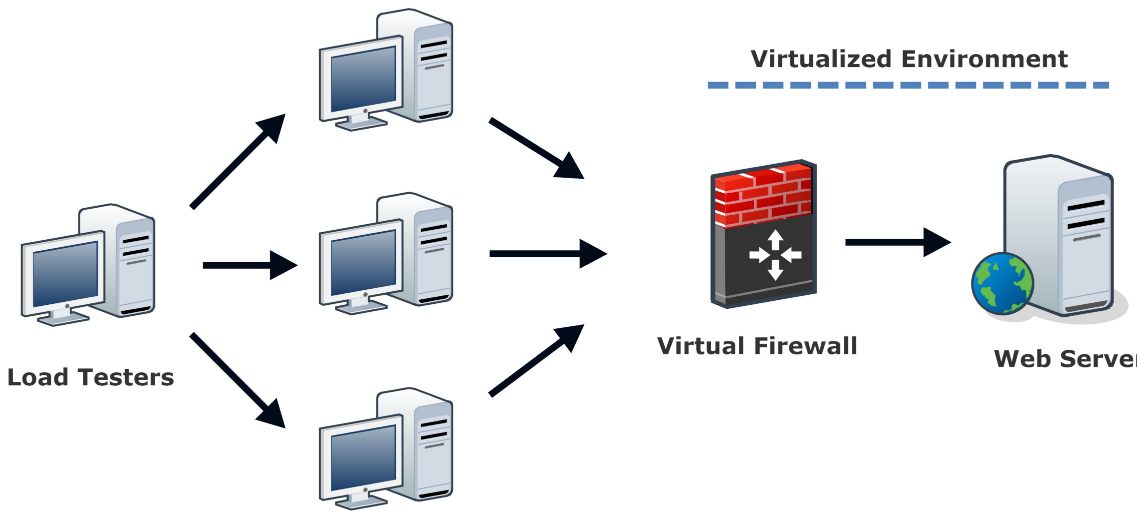


Fig. 4.2. Virtual Environment Architecture

4.4 Implementation & Configuration Settings

Load Tester - HOIC

We used HOIC as the HTTP Flood load tester. HOIC is an open source network stress testing and denial of service attack application written in BASIC. It is designed to attack as many as 256 URLs simultaneously Unlike the normal master/zombie architecture of DDoS attacks, HOIC works based on bandwidth availability. Using HOIC, we were able to generate HTTP traffic of approximately 1Gbps per thread. These were generated from 249 Cannons and 2 Threads. Table 4.1 shows our HOIC configuration settings and the resulting output per thread generated.

[3]See Appendix A2

Table 4.1
Configuring HOIC for Virtual Load Testing

Target URL/IP	10.10.10.79
Power	HIGH.
Booster	GenericBoost.hoic
Threads	2
Cannons per Thread	249
Output per Thread	>1GB

1. **URL** - The address of the Apache Web server
2. **Power** - This sets the velocity. We set the power to 'HIGH', making it 8 request/sec for each thread.
3. **Booster** - The script used to generate the HTTP Flood. We used the 'GenericBoost.hoic' script.
4. **Threads** - The number of users HOIC will attempt to simulate.

Building a Test Plan - HOIC

The test plan is composed of a sequence of the components presented above. The components determines how the traffic will be generated. We configured these details. Table 4.1 presents the configuration values for the virtual firewall load testing.

Similar to the Traditional Test, the HTTP GET Flood we initiated is legitimately downloading a picture (pic1.jpg) from the Apache Web server, meaning it establishes a full TCP session and actually receives all the data from the web page because the virtual firewall sits inline between the load tester (client) and the web server. The firewall must process every GET request that passes through. More than 1GB of traffic were generated per threat per session. We finally over-flooded the virtual firewall because it couldn't process each an every request that was sent.

From the settings configured, the HTTP packets that targeted the Apache web server were captured as:

```
- - 72.192.214.223 - "GET / HTTP/2" "10.10.10.79" 200 21124
- - 72.192.214.223 - "GET / HTTP/2" "10.10.10.79" 201 21124
- - 72.192.214.223 - "GET / HTTP/2" "10.10.10.79" 202 21124
- - 72.192.214.223 - "GET / HTTP/2" "10.10.10.79" 203 21124
- - 72.192.214.223 - "GET / HTTP/2" "10.10.10.79" 204 21124
- - 72.192.214.223 - "GET / HTTP/2" "10.10.10.79" 205 21124
```

4.4.1 HTTP GET Flood Request

HOIC request are not static; they randomly request from different sources, based on bandwidth. The HTTP Flood request is presented below:

```
GET / HTTP/2
Accept: */*
Accept-Language: en
Referer: GenericBoost.hoic
User-Agent: HOIC/4.0 (CLR 1.1.4322)
If-Modified-Since: Fri, 10 Jun 2016 11:59:59 GMT
Host: 10.10.10.79
```

The request specifies "HTTP/2," which is the successor of HTTP/1.1 that was standardized in 2015. HTTP/2 enables a more efficient use of network resources and a reduced perception of latency by introducing header field compression and allowing multiple concurrent exchanges on the same connection (Belshe, Thomson, & Peon, 2015). The host header can be analyzed using packet analysis tools, such as wireshark, which is beyond the scope of this dissertation. However, we can see the host address as **"10.10.10.79."**

Virtual Packet Filter Access Control

Similar to the traditional firewall, we blocked all incoming traffic on eth0 (the outside ethernet port) of the virtual firewall. We allowed ICMP, HTTP, and the TCP Port 80. This allows for end to end communication and pinging between the client and web-server, Below are the Access Control List applied to eth0:

- Rule 1:

```
Vyatta# set firewall name FWRULES-1 rule 1 action reject
Vyatta# set firewall name FWRULES-1 rule 1 source address X.X.X.X
Vyatta# set firewall name FWRULES-1 rule 1 protocol TCP
```

- Rule 2:

```
Config# set firewall name FWRULES-1 rule 2 action reject
Config# set firewall name FWRULES-1 rule 2 source address X.X.X.X
Config# set firewall name FWRULES-1 rule 2 protocol TCP
```

- Rule 3:

```
Config# set firewall name FWRULES-1 rule 3 action reject
Config# set firewall name FWRULES-1 rule 3 protocol UDP
Config# set firewall name FWRULES-1 rule 3 destination port 520
```

- Rule 4:

```
Config# set firewall name FWRULES-1 rule 4 action reject
Config# set firewall name FWRULES-1 rule 4 source address X.X.X.X
Config# set firewall name FWRULES-1 rule 4 protocol IP
```

- Rule 5:

```
Config# set firewall name FWRULES-1 rule 5 action reject
Config# set firewall name FWRULES-1 rule 5 protocol OSPF
```

- Rule 6:

```
Config# set firewall name FWRULES-1 rule 6 action reject
Config# set firewall name FWRULES-1 rule 6 source address X.X.X.X
Config# set firewall name FWRULES-1 rule 6 protocol TCP
```

- Rule 7:

```
Config# set firewall name FWRULES-1 rule 7 action accept
Config# set firewall name FWRULES-1 rule 7 protocol TCP
Config# set firewall name FWRULES-1 rule 7 destination port 80
```

- Rule 8:

```
Config# set firewall name FWRULES-1 rule 8 action reject
Config# set firewall name FWRULES-1 rule 8 protocol TCP
Config# set firewall name FWRULES-1 rule 8 destination port 21
```

- Rule 9:

```
Config# set firewall name FWRULES-1 rule 9 action reject
Config# set firewall name FWRULES-1 rule 9 protocol TCP
Config# set firewall name FWRULES-1 rule 9 destination port 22
```

- Rule 10:

```
Config# set firewall name FWRULES-1 rule 10 action reject
Config# set firewall name FWRULES-1 rule 10 protocol TCP
Config# set firewall name FWRULES-1 rule 10 destination port 25
```

- Rule 11:

```
Config# set firewall name FWRULES-1 rule 11 action reject
Config# set firewall name FWRULES-1 rule 11 protocol TCP
Config# set firewall name FWRULES-1 rule 11 destination port 110
```

- Rule 12:

```
Config# set firewall name FWRULES-1 rule 12 action reject
Config# set firewall name FWRULES-1 rule 12 protocol TCP
Config# set firewall name FWRULES-1 rule 12 destination port 143
```

- Rule 13:

```
Config# set firewall name FWRULES-1 rule 13 action reject
Config# set firewall name FWRULES-1 rule 13 protocol UDP
Config# set firewall name FWRULES-1 rule 13 destination port 135
```

- Rule 14:

```
Config# set firewall name FWRULES-1 rule 14 action reject
Config# set firewall name FWRULES-1 rule 14 protocol TCP
Config# set firewall name FWRULES-1 rule 14 destination port 445
```

- Rule 15:

```
Config# set firewall name FWRULES-1 rule 15 action reject
Config# set firewall name FWRULES-1 rule 15 protocol TCP
Config# set firewall name FWRULES-1 rule 15 destination port 1434
```

- Rule 16:

```
Config# set firewall name FWRULES-1 rule 16 action reject
Config# set firewall name FWRULES-1 rule 16 protocol TCP
Config# set firewall name FWRULES-1 rule 16 destination port 4444
```

- Rule 17:

```
Config# set firewall name FWRULES-1 rule 17 action reject
Config# set firewall name FWRULES-1 rule 17 protocol TCP
Config# set firewall name FWRULES-1 rule 17 destination port 4899
```

- Rule 18:

  ```
  Config# set firewall name FWRULES-1 rule 18 action accept
  Config# set firewall name FWRULES-1 rule 18 protocol ICMP
  ```

- Rule 19:

  ```
  Config# set firewall name FWRULES-1 rule 19 action accept
  Config# set firewall name FWRULES-1 rule 19 source address 10.10.10.0/24
  Config# set firewall name FWRULES-1 rule 19 protocol TCP
  ```

- Rule 20:

  ```
  Config# set firewall name FWRULES-1 rule 20 action accept
  Config# set firewall name FWRULES-1 rule 20 protocol HTTP
  Config# set firewall name FWRULES-1 rule 20 destination address 10.10.10.79
  ```

- Apply to interface and commit:

  ```
  Config# set interfaces ethernet eth1 firewall in name FWRULES-1
  Config# commit
  ```

- To show firewall rules:

  ```
  Config# show firewall name FWRULES-1
  ```

- To show interface rules:

  ```
  Config# show interfaces ethernet eth1 firewall
  ```

By default, the virtual firewall rules are not stateful. The firewall has a default drop rule that is active when a default action is not specified. To enable our stateful rules and ensure that incoming traffic on eth0 for our test session is allowed. We established these new commands:

```
firewall {
name clienttoserver {
default-action drop
rule 5 {
action accept
destination {
port 80
}
protocol tcp
state {
established enable
new enable
related enable
}
}
}
name servertoclient {
default-action drop
rule 5 {
action accept
state {
established enable
related enable
}
}
```

4.5 Evaluation & Analysis

The virtual firewall was still working at an optimum performance with 99% memory utilization. The virtual firewall was responsive because of the resource sharing management capabilities of virtualized environments. We then continue to flood the virtual firewall with unlimited traffic generated from HOIC until the CPU Utilization reached 71%, at which point the virtual firewall eventually became non-responsive. Table 4.2 and Figure 4.3 show the virtual firewall resource usage during the attack before it failed.

Table 4.2
Virtual Resources Usage - 8GB

	Memory	**CPU - Hypervisor Monitored**	**Packet Transferred**
Total	8GB	4.66GHz	>250GB of Packets
Used (%)	99.17%	71%	-
Free (%)	0.83%	29%	-

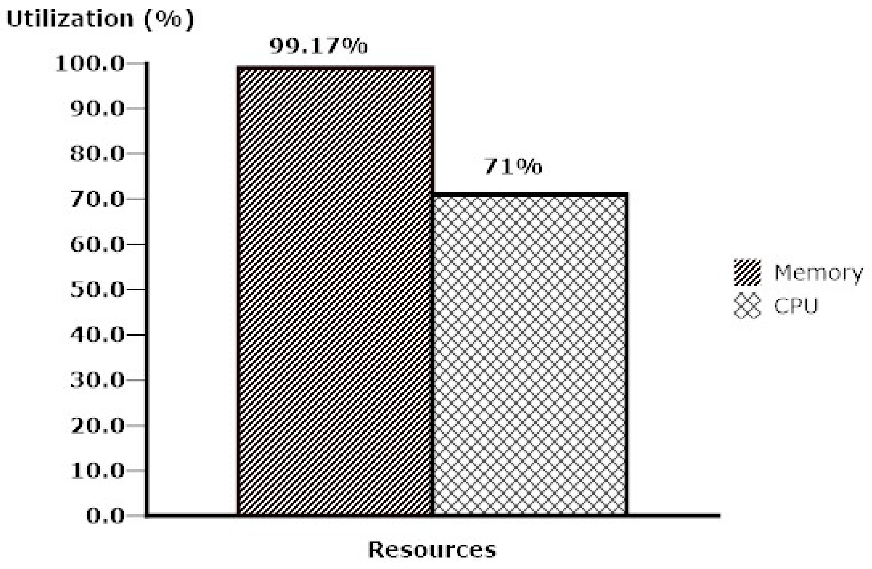


Fig. 4.3. Virtual Architecture Results

Table 4.3
Time Period Analysis for Virtual Packet Filter

Action	**Time (secs)**
Downtime	12
Recovery Time	28

From these results and previous studies, we see that the virtual firewall does a better job at responding to DDoS attacks than the traditional firewall, thereby making it a better choice when it comes to mitigating DDoS and transferring the hardware filtering services.

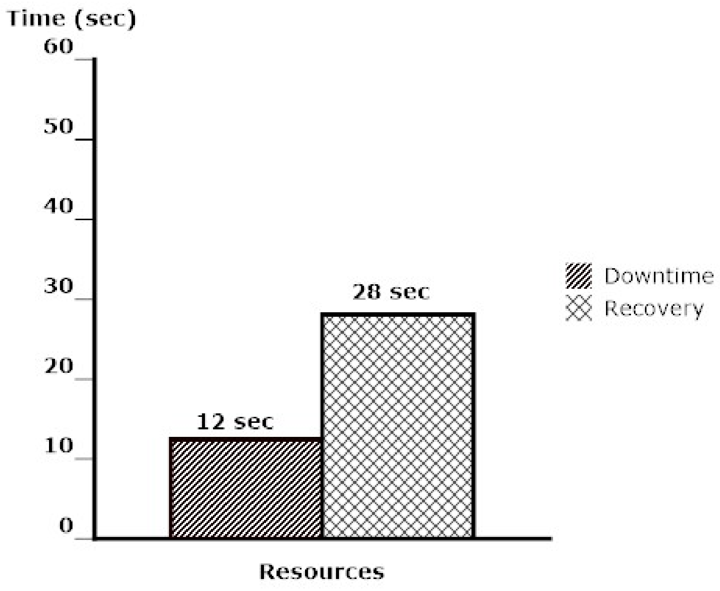


Fig. 4.4. Virtual Downtime & Recovery Time Analysis

4.5.1 Downtime & Recovery Period Analysis

Table 4.3 shows the downtime and recovery time of the virtual firewall after it failed. The virtual firewall was non-responsive for about 12 seconds before it became responsive. It took about 28 seconds to recover back to its working state. The recovery period was after the test was suspended. The recovery time for the virtual firewall was better than that of the hardware firewall because the hypervisor was responding; hence the virtual firewall doesn't have to deal with hardware boot-up.

4.6 Chapter Summary

This chapter starts by explaining Network Function Virtualization and introduces the problems faced by virtualized architectures. We then highlight our virtual architecture set-up, its implementation and configuration, and conclude by explaining the outcome of stressing the virtual firewall.

5. PERFORMANCE MONITOR

5.1 Background

Our performance monitor is not the first performance monitor, or even the first firewall performance monitor. But to the best of our knowledge it is the first to provide a means of mitigating DDoS attacks by monitoring firewall performance. The paper visual firewall (C. P. Lee, Tros, Gibbs, Beyah, & Copeland, 2005) seeks to aid the configuration of firewalls and monitoring of networks by providing four simultaneous views that display varying levels of detail and time scales as well as correctly visualizing firewall reactions to individual packets. The four implemented views are: Real-Time Traffic, Visual Signature, Statistics, and IDS Alarm. These views provide the levels of detail that system administrators need to properly monitor their systems in passive or active manners. The paper visualized several attacks, and made sure that individuals unfamiliar with networking concepts can quickly distinguish between benign and malignant traffic patterns with minimal instruction. Another monitoring paper (Yuan & Mills, 2005) proposes a method for early detection. Using a few observation points, the proposed method can monitor the macroscopic effect of DDoS flooding attacks and then inform more detailed detection systems when a DDoS attack possibly arises in transit or source networks.

Others focused on packet monitoring in Cloud Environment to prevent DDoS attacks (Chouhan & Peddoju, 2013) using Hop Count Filtering. The approach of Hop Count Filtering provides a network independent and readily available solution to prevent DDoS attacks in Cloud environments. The method decreases the unavailability of cloud services to legitimate clients, reduce number of updates, and saves computation time. The approach is simulated in a CloudSim toolkit environment and corresponding results are produced. Others monitored firewall traffic using Intrusion Detection Systems (Asarcıklı, 2005).

However, none of these papers provided a mitigation solution to DDoS attacks targeting firewalls in traditional and virtual environments. Hence, the birth of our performance monitor as the first firewall monitor DDoS mitigation solution.

5.2 Performance Monitor Design

The performance monitor is mirrored to the hardware and virtual firewall. The monitor runs on a dedicated system as a virtual machine. Based on the results of both the traditional and virtual test, instructions were given. The monitor was coded in python and the threshold at which the firewalls can migrate packet filtering services was defined in the code. No GUI was used. The monitor connects with the firewalls through their respective API's. The communication between the monitor and the firewalls is encrypted using AES. MySQL database was connected to store the heuristics based on the defined thresholds. Figure 5.1 shows how the monitor connects to the firewall.

When either the traditional or virtual firewall becomes overwhelmed during the test, the monitor reroutes the traffic meant for the failed firewall to the other firewall, thereby allowing the overwhelmed firewall to resume to its working state, hence, downtime as a result of DDoS attack is minimized.

5.3 Implementation & Deployment

5.3.1 API Implementation

Implementing the API connections for both firewalls is based on the guide provided by the firewall vendors[1]. The typical request flow for the REST PUT/POST/DELETE API request are:

[1]http://www.cisco.com & http://www.brocade.com/

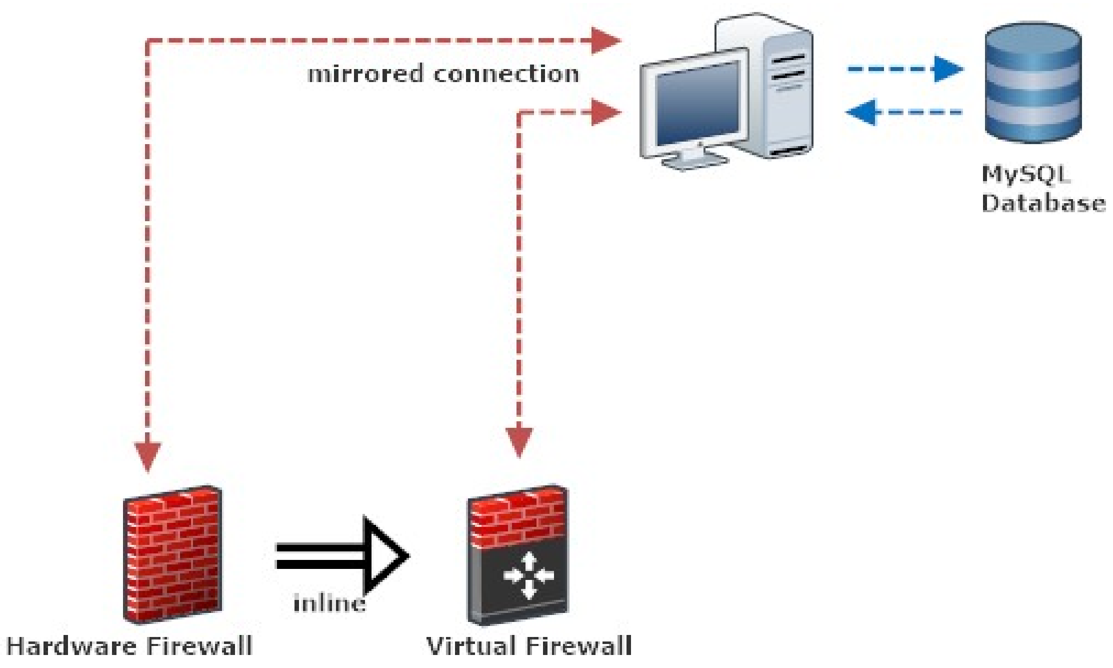


Fig. 5.1. Performance Monitor Architecture

- REST Client establishes SSL connection to the firewall
- REST Client sends API request with basic authentication header to ASA.
- Firewall server validates and processes clients request.
- Firewall HTTP server opens the connection to REST Agent using a TCP channel, and writes the HTTP request to the REST Agent.
- Firewall HTTP server waits for REST Agent processs response.
- REST Agent processes API request, picks the session/user info, and invokes CLI command requests to Admin handler listening on localhost port in Firewall. REST Agent includes the session/user info in the request.
- Admin handler processes the CLI commands and collects the result output.

- Admin handler sends the response for the CLI commands request to REST Agent.
- REST Agent prepares the response for REST API request and sends to the Firewall HTTP server.
- Firewall HTTP server forwards the response to the client. Server doesn't process the response received from the REST Agent process.

Basic Authorization header must be added to every REST API request and authentication will be performed for every request. It is recommended to use Certificate Authority (CA) issued certificates on the firewalls, so that REST API clients can validate the server certificates of the firewalls during the SSL connection establishment. The process of how the monitors API communicates with our firewalls is presented in Figure 5.2

The communication between the firewalls and the performance monitor is established using the firewall respective RESTAPI. REST stands for Representation State Transfer. It is a term coined by Roy Fielding in his dissertation (Fielding, 2000) to refer to a software architectural style. The REST architectural style describes six constraints: Uniform Interface, Stateless, Cache-able, Client-Server, Layered System, and Code on Demand. This constraint states that a REST API should be driven by nothing but hypermedia. This is how a web browser interacts with well-behaved web applications, where the browser transitions to different pages based on the selected hyperlinks and actions present in the pages. Since HTTP has been used to transfer information that is not hypertext, this constraint is often ignored by REST API designs. Instead of defining API in terms of hypermedia, some so-called REST APIs are modeled as a set of interfaces implemented by resources. Although this design supports the REST uniform interface constraint, it inevitably creates fixed resource names, types and hierarchies that violate the REST API design rules prescribed by Roy Fielding (Fielding, 2000). This kind of violations leads to an API that depends on the out-of-band information, instead of hypermedia, to drive the interactions be-

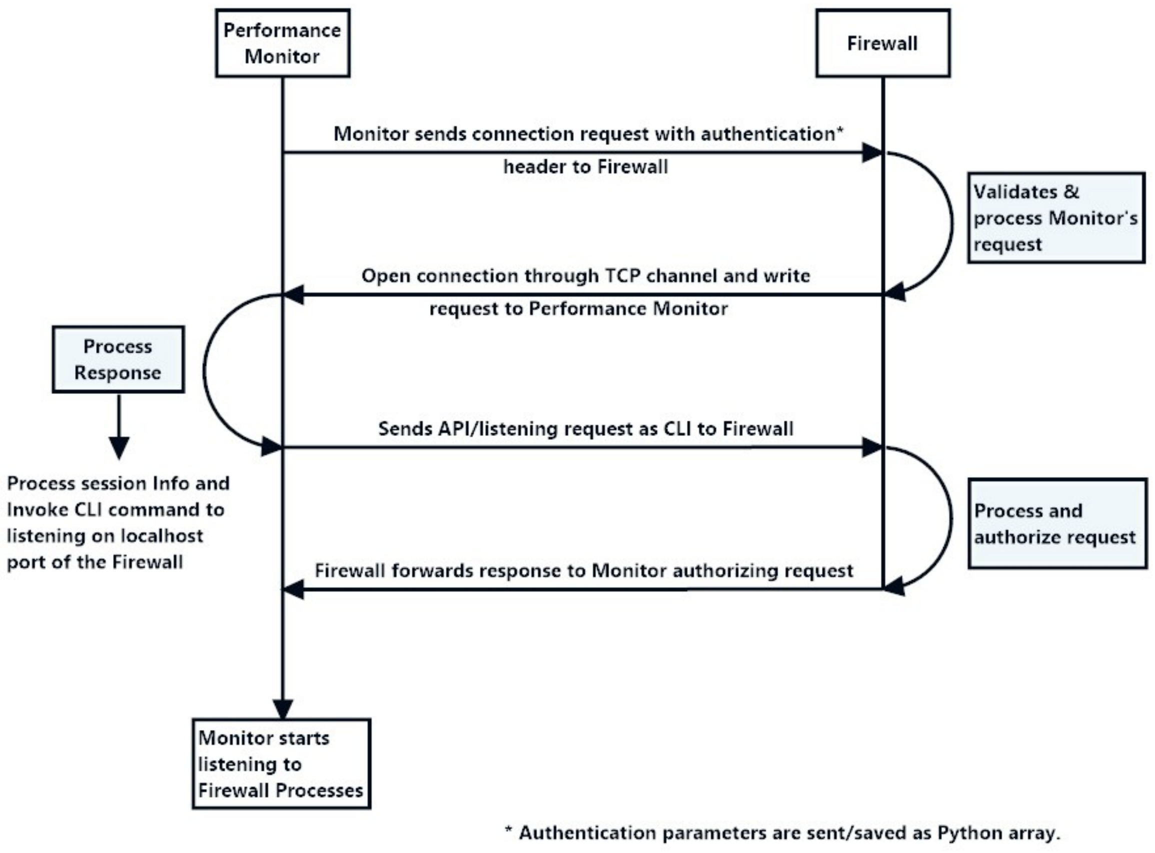

Performance Monitor
Firewall
Monitor sends connection request with authentication* header to Firewall
Validates & process Monitor's request
Open connection through TCP channel and write request to Performance Monitor
Process Response
Process session Info and Invoke CLI command to listening on localhost port of the Firewall
Sends API/listening request as CLI to Firewall
Process and authorize request
Firewall forwards response to Monitor authorizing request
Monitor starts listening to Firewall Processes
* Authentication parameters are sent/saved as Python array.

Fig. 5.2. API Call Process

tween components (Li & Chou, 2011). To properly understand how RESTAPIs work is beyond the scope of this dissertation.

5.3.2 MySQL Database Implementation

To store the thresholds and heuristic conditions used to migrate packet filtering services between the firewalls, we used a MySQL database. To do that, we installed the MySQL driver, because python doesn't come with MySQL; by default, it comes with SQLite. We installed the MySQL package. The implementation and use of MySQL is beyond the scope of this dissertation. However, the commands we used to install MySQL database in Ubuntu debian distros is:

```
sudo apt-get install python-mysqldb
```

After installation, we then used MySQL inside python like any other package. This is how we imported and connected MySQL to python:

```
#!/usr/bin/python
import MySQLdb
db =  MySQLdb.connect(host="localhost",
user="dissertation",
        passwd="dissertation_password",
        db="dissertation_db")
```

5.3.3 AES Implementation using PyCrypto

In other to secure the RESTAPI connection between the firewalls and the performance monitor, an AES implementation of python using PyCrypto[2] was used (Buchmann, 2013; Ferguson, Schneier, & Kohno, n.d.; Katz & Lindell, 2014; Lindell, 2005; Schneier, 1997). This implementation is based on the guide provided in "A Working Introduction to Crypto using PyCrypto" (Isom, 2011).

5.4 Performance Analysis

The performance monitor was analyzed by integrating it into the firewalls, by conducting the same test, and flooding the firewalls with HTTP traffic. The performance monitor proved to be effective after we surpassed the generated HTTP traffic that flooded the hardware firewall, and the firewall was still responsive. However, considering that the virtual firewall has a better performance than the hardware firewall, the operation was smooth. The packet filtering process of the hardware firewall was transferred to the virtual firewall. The process by which the performance monitor makes decision is presented in Figure 5.3.

[2]https://www.dlitz.net/software/pycrypto/api/2.6/

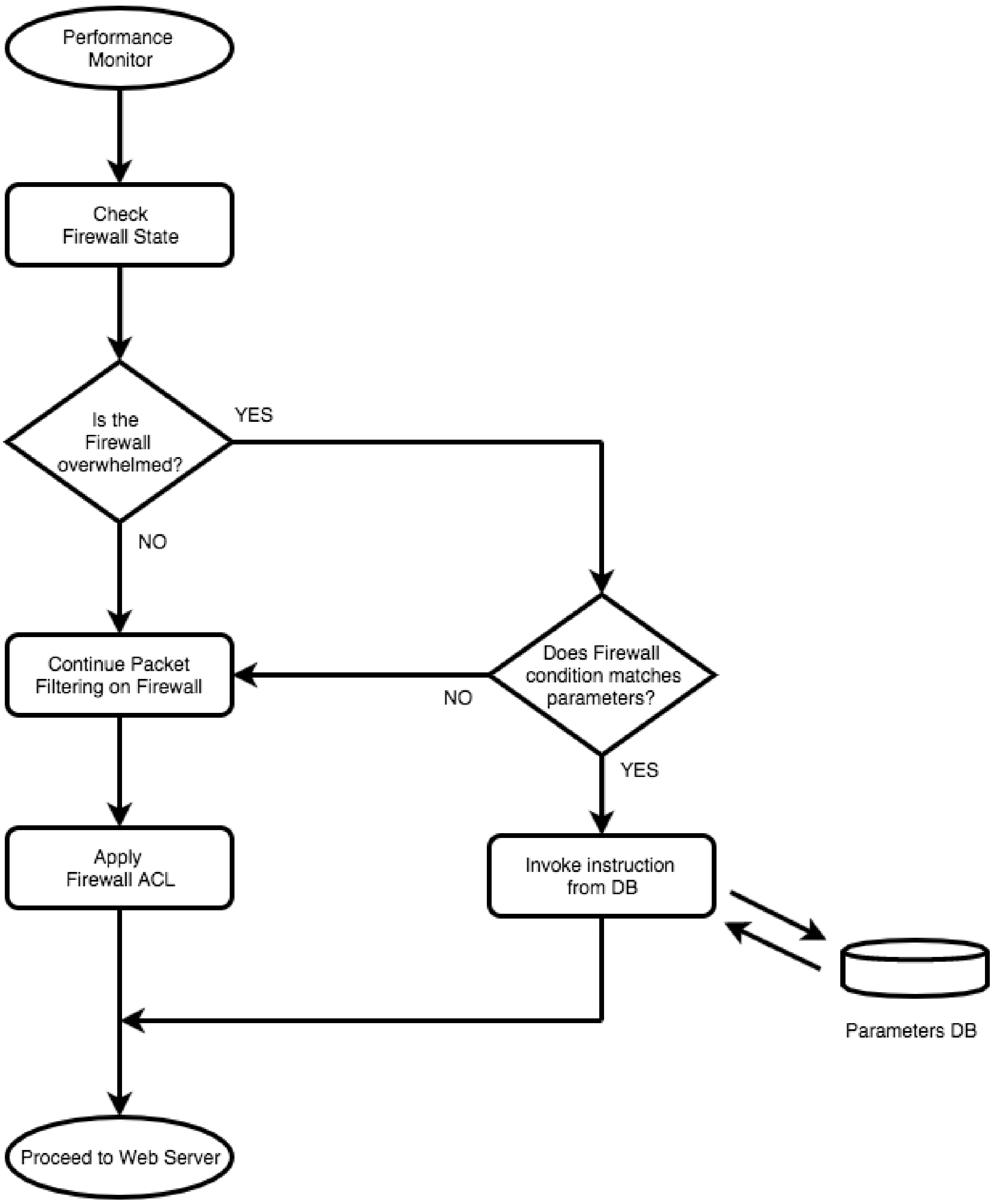

Performance Monitor
Check Firewall State
Is the Firewall overwhelmed?
YES
NO
Does Firewall condition matches parameters?
NO
YES
Continue Packet Filtering on Firewall
Apply Firewall ACL
Invoke instruction from DB
Parameters DB
Proceed to Web Server

Fig. 5.3. Performance Monitor/Migration Decision Flow

5.4.1 Filtering Service Transfer

The monitor starts by checking the firewall states to see if there is an anomaly, then it proceeds to make decisions; if the firewall is overwhelmed, it checks its parameters and invokes the instructions that were defined, then transfers the filtering services. If the firewall is not overwhelmed; it applies the firewall's access list and continues operation by proceeding to its designated destination.

5.4.2 Performance Comparison

Table 5.1 shows the performance results of the devices before and after using the monitor. The memory utilization of the hardware firewall considerably reduced from 81% to 34% without the monitor. The CPU utilization reduced from 67% to 28%.

Table 5.1
Comparative Analysis of Performance Results

Architecture	Mem Utilization	CPU Utilization	Packet Drop
Traditional	81%	67%	76.5%
Virtual	99.17%	71%	-
Traditional with PM*	34%	28%	-
Virtual with PM*	98.2%	73.6%	-

However, for the virtual firewall, the change was a minimal decrease. The performance of both Memory & CPU of the virtual device with the monitor connected decreased from 98.2% to 73.6% respectively, compared to 99.17% and 71% without the monitor. The performance decrease is as a result of the monitor being added to the topology. This shows a success; most importantly, downtime was avoided, considering none of the devices failed.

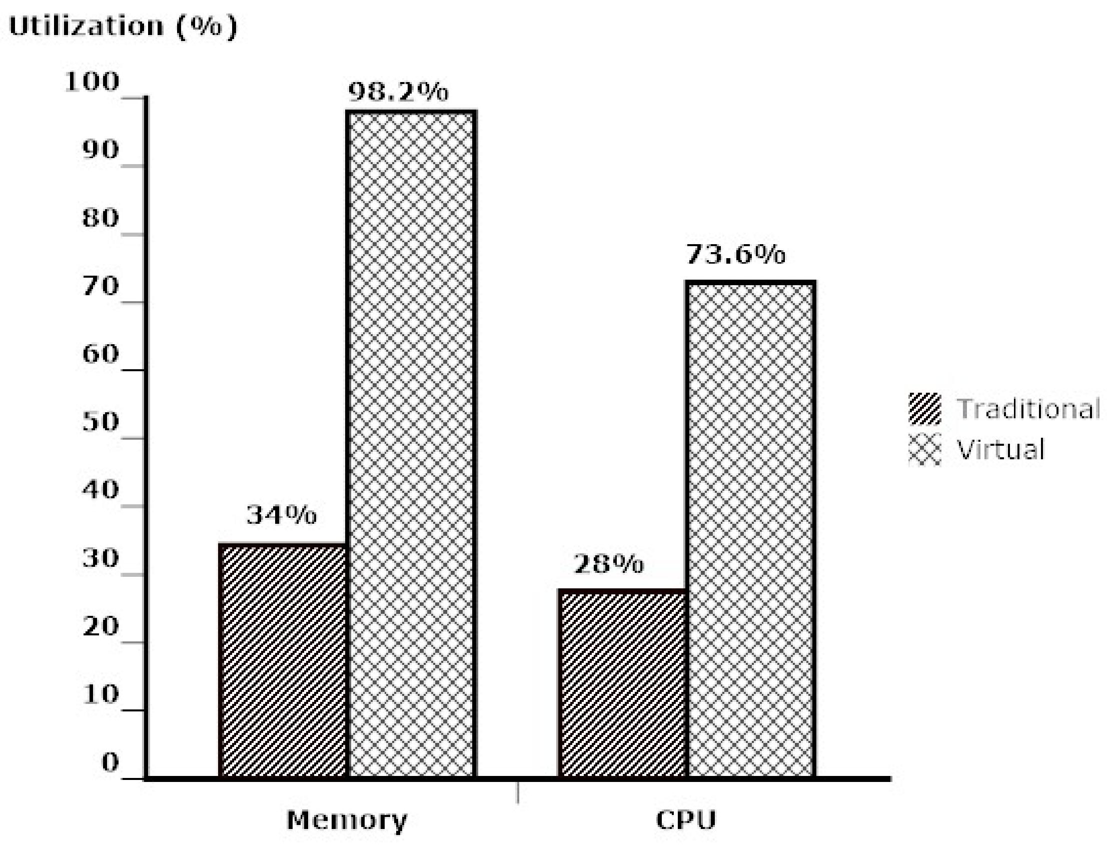


Fig. 5.4. Comparative Performance Analysis

5.4.3 Thresholds & Heuristic Rules/SLA

Based on the analysis and thresholds identified from both firewalls, we defined these heuristics and embedded them as part of the monitor's instructions. These are the actions the performance monitor does whenever one of the set thresholds is reached. These rules can always be defined based on SLA if such topology were to be adopted.

- For Hardware Instructions:

  - If Packet Drops is high, then migrate packet filtering service to virtual firewall, otherwise continue on hardware firewall.

- If CPU Utilization is high, then migrate packet filtering service to virtual firewall, otherwise continue on hardware firewall.

- If Memory Utilization is high, then migrate packet filtering service to virtual firewall, otherwise continue on hardware firewall.

- If none of the defined heuristics is met, then continue packet filtering service on hardware firewall.

Algorithm 1 Hardware & Virtual Migration

```
Require: Packet Drop - $PD$, CPU Utilization - $CU$, Memory Utilization - $MU$
Input: Defined Threshold values into Monitor $(PD, CU, MU)$
Output: Compare Current State with Defined Thresholds $(PD, CU, MU)$
 1: while $PD \geq 10\%, CU \geq 65\%, MU \geq 80\%$ do
 2:   Initialization Migration Instructions
 3:   if $CurrentPD \geq DefinedPD$ then
 4:     Instruction $MigrateFilteringServives$ {Migrate Packet Filtering
        Services between Firewalls}
 5:     Else $ContinueFilteringOnDevice$
 6:   end if
 7:   if $CurrentCU \geq DefinedCU$ then
 8:     Instruction $MigrateFilteringServives$ {Migrate Packet Filtering
        Services between Firewalls}
 9:     Else $ContinueFilteringOnDevice$
10:   end if
11:   if $CurrentMU \geq DefinedMU$ then
12:     Instruction $MigrateFilteringServives$ {Migrate Packet Filtering
        Services between Firewalls}
13:     Else $ContinueFilteringOnDevice$
14:   end if
15: end while
```

- For Virtual Instructions:

  - If Packet Drops is high, then migrate packet filtering services to hardware firewall otherwise continue on virtual firewall.

  - If CPU Utilization is high, then migrate packet filtering services to hardware firewall, otherwise continue on virtual firewall.

- If none of the defined heuristics is met, then continue packet filtering services on hardware firewall.

5.5 Chapter Summary

In this chapter, we introduced a firewall performance monitor. The monitor connects to two firewalls and injects instructions when certain conditions are met. We discuss the design, implementation, deployment, and performance analysis. The performance monitor is not a specific method of DDoS mitigation, but a means for organizations to test their devices, by stressing them and finding the device threshold, then using those thresholds to create a heuristic that can be used to configure the monitor. We explain how the communication between the monitor and the devices is encrypted using AES by utilizing the PyCrypto implementation of python. We showed how the monitor stores data using the SQL database.

6. CONCLUSION

6.1 Connecting the Dots: Justifying the Problem Statement

This dissertation focused on carrying out different types of tests to obtain the results of the research objectives stated earlier. We implemented a monitor in a secured architecture with two firewalls from different platforms, one using a virtual-based and the other a hardware-based. The monitor monitors the state of the firewalls and invoke certain instructions based on the firewalls state. Both the firewalls and monitor are designed to work together in order to provide an optimized packet filter architecture and minimize downtime when the firewalls become overwhelmed due to a distributed denial of service (DDoS) attacks. To achieve this, two different tests (spike and endurance) at three stages were conducted. A web-server was built to serve as the target of the HTTP traffic.

First we implement and test the traditional environment which consists of the hardware firewall. We attacked the firewall using a HTTP flood DDoS attack to find the hardware firewall weakness and define its threshold. Second, we carried out the same test in the virtual environment, which consists of the virtual firewall. The second test was also aimed at finding the virtual firewalls weaknesses and limitations. That was achieved by flooding the firewall with a HTTP flood DDoS attack. The firewall's threshold was determined at the end. The last phase was implementing the monitor that monitors the states of both firewalls.

The monitor was designed to switch packet filtering between the firewalls when one of the firewalls reaches its threshold. Switching the packet filtering ensures that downtime is avoided within the network, allowing the firewall to return back to its stable state before the filtering processes resumes. The monitor monitors the firewall's state based on the heuristic rules fed to it. Those heuristic rules and threshold can be defined based on SLA. Each organization uses different vendors when it comes to network devices; hence, a standard threshold cannot be defined. Each device must go through the same process, then define the threshold.

The threshold of each device was determined based on CPU Utilization and Memory Utilization.

6.2 Summary

In this dissertation, we looked into how DDoS attacks result in network downtime, costing organizations billions of dollars. Ease of authorized availability and access to information is one of the core foundation of computer security. According to census.gov[1], 97% of our day-to-day data, such as health information, education information, etc. are all stored digitally, thereby making authorized access to such data of paramount importance. DDoS attacks cause network downtime, making it hard for authorized users to access to important data. We highlighted the main issues with DDoS attacks in earlier chapters.

In addition, we test different network architectures and present a framework which could be used to mitigate DDoS attacks and provide more up-time for computer networks in chapters 3 and 4.

In chapter 5, we presented a framework that is used to mitigate DDoS and provide more up-time in a network. We designed a tool (Firewall Performance Monitor) which listens to the performance of the network devices and then executes instructions when certain conditions are met.

We concluded the dissertation by explaining the design and implementation of the firewall performance monitor.

[1]Census Internet Statistics

6.3 Recommendations & Future Work

The framework we presented provides one of the first non-commercial approaches to mitigating flooding attacks on firewalls and ensuring more network up-time. The use of the monitor has shown promising results in enhancing network security systems.

Learning from this study, security administrators can optimize the performance of firewalls in a network by first evaluating CPU utilization, Memory Utilization, Packet Drops, and other available resources. Furthermore, security administrators can implement the same methods we presented to ensure maximum network availability. Keeping in mind that the heuristics the monitor utilizes to make decisions is dependent on the thresholds defined by the network devices, security administrators can decide on each device threshold to meet its demands. The monitor's heuristics can only be defined based on service level agreements.

An interesting area of future research could focus on the monitor itself; considering it runs as a continuous script with infinite max loop. It would be interesting to see it developed as a software running with a simplified GUI, thereby making the heuristics definition more simplified. Having the monitor run in a closedbox would be another focus for future research; that would enhance control and limit access to the monitor's source-code, thereby ensuring greater security.

Another interesting area of future research is a broader application of same technique presented in this study; The monitor should be deployed other network devices susceptible to DDoS attacks, and not just to network firewalls. It would be interesting to apply the same method to a larger network, to deploy on a large-scale, real world setting and analyze the results. With this approach, our monitor has the potential of being the cutting-edge in finding an innovative solution to DDoS attacks.

LIST OF REFERENCES

LIST OF REFERENCES

Abbes, T., Bouhoula, A., & Rusinowitch, M. (2008). An inference system for detecting firewall filtering rules anomalies. In *Proceedings of the 2008 acm symposium on applied computing* (pp. 2122–2128).

Adams, K., & Agesen, O. (2006). A comparison of software and hardware techniques for x86 virtualization. *ACM Sigplan Notices*, *41*(11), 2–13.

Agarwal, S., Dawson, T., & Tryfonas, C. (2003). *Ddos mitigation via regional cleaning centers* (Tech. Rep.). Sprint ATL Research Report RR04-ATL-013177.

Alliance, C. (2011). Security guidance for critical areas of focus in cloud computing v3. 0. *Cloud Security Alliance*.

Alomari, E., Manickam, S., Gupta, B., Karuppayah, S., & Alfaris, R. (2012). Botnet-based distributed denial of service (ddos) attacks on web servers: classification and art. *arXiv preprint arXiv:1208.0403*.

Anderson, J. M. (2003). Why we need a new definition of information security. *Computers & Security*, *22*(4), 308–313.

Anderson, J. P. (1972). *Computer security technology planning study. volume 2* (Tech. Rep.). DTIC Document.

Anderson, R. (2001). Why information security is hard-an economic perspective. In *Computer security applications conference, 2001. acsac 2001. proceedings 17th annual* (pp. 358–365).

Ando, R., Miwa, S., Kadobayashi, Y., & Shinoda, Y. (2008). 4-2 a load balancing system for mitigating ddos attacks using live migration of virtual machines. *Journal of the National Institute of Information and Communications Technology*, *55*(2/3), 27–32.

Anwer, B., Benson, T., Feamster, N., & Levin, D. (2015). Programming slick network functions. In *Proceedings of the 1st acm sigcomm symposium on software defined networking research* (p. 14).

Asarcıklı, Ş. (2005). Firewall monitoring using intrusion detection systems.

Bade, M. G., & Vanduhe, V. Z. (n.d.). Ipv6 vs ipv4.

Baker, F., Iturralde, C., Le Faucheur, F., & Davie, B. (2001). *Aggregation of rsvp for ipv4 and ipv6 reservations* (Tech. Rep.).

Basak, D., Toshniwal, R., Maskalik, S., & Sequeira, A. (2010). Virtualizing networking and security in the cloud. *ACM SIGOPS Operating Systems Review*, *44*(4), 86–94.

Batalle, J., Ferrer Riera, J., Escalona, E., & Garcia-Espin, J. A. (2013). On the implementation of nfv over an openflow infrastructure: routing function virtualization. In *Future networks and services (sdn4fns), 2013 ieee sdn for* (pp. 1–6).

Beitollahi, H., & Deconinck, G. (2012). Analyzing well-known countermeasures against distributed denial of service attacks. *Computer Communications*, *35*(11), 1312–1332.

Beloglazov, A., & Buyya, R. (2010). Energy efficient resource management in virtualized cloud data centers. In *Proceedings of the 2010 10th ieee/acm international conference on cluster, cloud and grid computing* (pp. 826–831).

Belshe, M., Thomson, M., & Peon, R. (2015). Hypertext transfer protocol version 2 (http/2).

Bhardwaj, A., Subrahmanyam, G., Avasthi, V., & Sastry, H. (2015). Three tier network architecture to mitigate ddos attacks on hybrid cloud environments. *arXiv preprint arXiv:1512.02005*.

Bhavya, D. (2008). Network security: History, importance, and future. *University of Florida Department of Electrical and Computer Engineering*.

Blacharski, D. (1998). *Network security in a mixed environment with cdrom.* IDG Books Worldwide, Inc.

Buchmann, J. (2013). *Introduction to cryptography.* Springer Science & Business Media.

Buyya, R., Broberg, J., & Goscinski, A. M. (2010). *Cloud computing: principles and paradigms* (Vol. 87). John Wiley & Sons.

Byers, S., Rubin, A. D., & Kormann, D. (2004). Defending against an internet-based attack on the physical world. *ACM Transactions on Internet Technology (TOIT)*, *4*(3), 239–254.

Carl, G., Kesidis, G., Brooks, R. R., & Rai, S. (2006). Denial-of-service attack-detection techniques. *Internet Computing, IEEE*, *10*(1), 82–89.

Casilari, E., Gonzblez, F., & Sandoval, F. (2001). Modeling of http traffic. *Communications Letters, IEEE*, *5*(6), 272–274.

Chang, R. K. (2002). Defending against flooding-based distributed denial-of-service attacks: a tutorial. *Communications Magazine, IEEE*, *40*(10), 42–51.

Chen, W., & Yeung, D.-Y. (2006). Defending against tcp syn flooding attacks under different types of ip spoofing. In *Networking, international conference on systems and international conference on mobile communications and learning technologies, 2006. icn/icons/mcl 2006. international conference on* (pp. 38–38).

Cheswick, W. R., Bellovin, S. M., & Rubin, A. D. (2003). *Firewalls and internet security: repelling the wily hacker.* Addison-Wesley Longman Publishing Co., Inc.

Chomsiri, T., & Pornavalai, C. (2006). Firewall rules analysis. In *Security and management* (pp. 213–219).

Chouhan, V., & Peddoju, S. K. (2013). Packet monitoring approach to prevent ddos attack in cloud computing. *International Journal of Computer Science and Electrical Engineering (IJCSEE) ISSN*(2315-4209).

Chowdhury, N. M. K., & Boutaba, R. (2010). A survey of network virtualization. *Computer Networks*, *54*(5), 862–876.

Cisco. (2004). *Cisco guard ddos mitigation appliances.* http://www.cisco.com/c/en/us/products/collateral/security/traffic-anomaly -detector-xt-5600a/prod white paper0900aecd8011e927.pdf.

Cisco. (2014). *What is network security?* Cisco Website.

CloudFlare. (n.d.). *Keyless ssl.* https://www.cloudflare.com/keyless-ssl.

Cole, E. (2011). *Network security bible* (Vol. 768). John Wiley & Sons.

Das, D., Sharma, U., & Bhattacharyya, D. (2011). Detection of http flooding attacks in multiple scenarios. In *Proceedings of the 2011 international conference on communication, computing & security* (pp. 517–522).

Davidowicz, D. (1999). Domain name system (dns) security. *Yahoo Geocities*.

Deering, S. (1991). Icmp router discovery messages.

Deshpande, H. A. (2015). Honeymesh: Preventing distributed denial of service attacks using virtualized honeypots. *arXiv preprint arXiv:1508.05002*.

Dieter, G. (1999). Computer security. *Chinchester ua*.

Dou, W., Chen, Q., & Chen, J. (2013). A confidence-based filtering method for ddos attack defense in cloud environment. *Future Generation Computer Systems*, *29*(7), 1838–1850.

Douligeris, C., & Mitrokotsa, A. (2004). Ddos attacks and defense mechanisms: classification and state-of-the-art. *Computer Networks*, *44*(5), 643–666.

Eronen, P., & Zitting, J. (2001). An expert system for analyzing firewall rules. In *Proceedings of the 6th nordic workshop on secure it systems (nordsec 2001)* (pp. 100–107).

Estevez-Tapiador, J. M., García-Teodoro, P., & Díaz-Verdejo, J. E. (2005). Detection of web-based attacks through markovian protocol parsing. In *Computers and communications, 2005. iscc 2005. proceedings. 10th ieee symposium on* (pp. 457–462).

ESTI. (n.d.). Network function virtualization. http://www.etsi.org/technologies-clusters/technologies/nfv.

ESXi, V. (n.d.). *Bare metal hypervisor.*

Farahmandian, S., Zamani, M., Akbarabadi, A., Moghimi, J., Zadeh, S. M. M., & Farahmandian, S. (2013). A survey on methods to defend against ddos attack in cloud computing. *system*, *6*(22), 26.

Ferguson, N., Schneier, B., & Kohno, T. (n.d.). Introduction to cryptography. *Cryptography Engineering*, 23–39.

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures.* Unpublished doctoral dissertation, University of California, Irvine.

Gandel, S. (2015). *Lloyd's ceo: Cyber attacks cost companies $400 billion every year.* http://fortune.com/2015/01/23/cyber-attack-insurance-lloyds/.

Garfinkel, S., Spafford, G., & Schwartz, A. (2003). *Practical unix and internet security.* " O'Reilly Media, Inc.".

Garg, K., & Chawla, R. (2011). Detection of ddos attacks using data mining. *International Journal of Computing and Business Research (IJCBR)*, *2*(1).

Gibson, S. (2001). *The strange tale of the denial of service attacks against grc. com.*

Golnabi, K., Min, R. K., Khan, L., & Al-Shaer, E. (2006). Analysis of firewall policy rules using data mining techniques. In *Network operations and management symposium, 2006. noms 2006. 10th ieee/ifip* (pp. 305–315).

Griffiths, J. (2015). *Cybercrime costs the average u.s firm $15 million a year.* http://money.cnn.com/2015/10/08/technology/cybercrime-cost-business/.

Guillen, E., Sossa, A. M., & Estupiñán, E. P. (2012). Performance analysis over software router vs. hardware router: A practical approach. In *Proceedings of the world congress on engineering and computer science* (Vol. 2, pp. 24–26).

Gupta, M. (2010). *Ipv4 vs. ipv6.*

Harris, B., Konikoff, E., & Petersen, P. (2013). Breaking the ddos attack chain. *Institute for Software Research.*

Hazelhurst, S. (2000). Algorithms for analysing firewall and router access lists. *arXiv preprint cs/0008006.*

Hazelhurst, S., Attar, A., & Sinnappan, R. (2000). Algorithms for improving the dependability of firewall and filter rule lists. In *Dependable systems and networks, 2000. dsn 2000. proceedings international conference on* (pp. 576–585).

Huang, W., & Yang, J. (2010). New network security based on cloud computing. In *2010 second international workshop on education technology and computer science* (pp. 604–609).

Ierace, N., Urrutia, C., & Bassett, R. (2005). Intrusion prevention systems. *Ubiquity*, *2005*(June), 2–2.

Incapsula. (n.d.). *The practical guide to choosing a ddos mitigation service.* https://www.imperva.com/docs/DSIncapsulaGuideToSelectingADDoSSolution.pdf.

Incapsula. (2014). *The top 10 ddos attack trends.* https://www.imperva.com/docs/DS Incapsula The Top 10 DDoS Attack Trends ebook.pdf.

Ioannidis, J., & Bellovin, S. M. (2002). Implementing pushback: Router-based defense against ddos attacks.

Ioannidis, S., Keromytis, A. D., Bellovin, S. M., & Smith, J. M. (2000). Implementing a distributed firewall. In *Proceedings of the 7th acm conference on computer and communications security* (pp. 190–199).

Irvine, C. E., Robin, J. S., et al. (2000). Analysis of the intel pentium's ability to support a secure virtual machine monitor..

ISC, I. S. C. (2015). *Internet domain survey, july, 2015.* https://ftp.isc.org/www/survey/reports/current/.

Isom, K. (2011). A working introduction to crypto with pycrypto.

Jajodia, S., Ammann, P., & McCollum, C. D. (1999). Surviving information warfare attacks. *Computer*, *32*(4), 57–63.

JMeter. (n.d.). *Jmeter user guide: http://jmeter.apache.org/usermanual/testplan.html.*

Katz, J., & Lindell, Y. (2014). *Introduction to modern cryptography.* CRC Press.

Kayssi, A., Harik, L., Ferzli, R., & Fawaz, M. (2000). Fpga-based internet protocol firewall chip. In *Electronics, circuits and systems, 2000. icecs 2000. the 7th ieee international conference on* (Vol. 1, pp. 316–319).

Kenney, M. (1996). Ping of death. *Insecure. org.*

Keromytis, A. D., Misra, V., & Rubenstein, D. (2004). Sos: An architecture for mitigating ddos attacks. *Selected Areas in Communications, IEEE Journal on*, *22*(1), 176–188.

Koutepas, G., Stamatelopoulos, F., & Maglaris, B. (2004). Distributed management architecture for cooperative detection and reaction to ddos attacks. *Journal of Network and Systems Management*, *12*(1), 73–94.

Lee, C. P., Tros, J., Gibbs, N., Beyah, R., & Copeland, J. A. (2005). Visual firewall: real-time network security monitor. In *Visualization for computer security, 2005.(vizsec 05). ieee workshop on* (pp. 129–136).

Lee, K., Kim, J., Kwon, K. H., Han, Y., & Kim, S. (2008). Ddos attack detection method using cluster analysis. *Expert Systems with Applications*, *34*(3), 1659–1665.

Li, L., & Chou, W. (2011). Design and describe rest api without violating rest: A petri net based approach. In *Web services (icws), 2011 ieee international conference on* (pp. 508–515).

Lihua, Y., Jianning, M., & Zhendong, S. (2006). Fireman: A toolkit for firewall modeling and analysis. In *Proceedings of the 2006 ieee symposium on security and privacy.*

Lindell, Y. (2005). Introduction to cryptography. *Lecture Notes, available at http://www. cs. biu. ac. il/ lindell/89-656/Intro-to-crypto-89-656. pdf*.

Lonea, A. M., Popescu, D. E., & Tianfield, H. (2013). Detecting ddos attacks in cloud computing environment. *International Journal of Computers Communications & Control*, *8*(1), 70–78.

Mah, B. A. (1997). An empirical model of http network traffic. In *Infocom'97. sixteenth annual joint conference of the ieee computer and communications societies. driving the information revolution., proceedings ieee* (Vol. 2, pp. 592–600).

Manohar, N. (2013). A survey of virtualization techniques in cloud computing. In *Proceedings of international conference on vlsi, communication, advanced devices, signals & systems and networking (vcasan-2013)* (pp. 461–470).

Martins, J., Ahmed, M., Raiciu, C., Olteanu, V., Honda, M., Bifulco, R., & Huici, F. (2014). Clickos and the art of network function virtualization. In *Proceedings of the 11th usenix conference on networked systems design and implementation* (pp. 459–473).

Mayer, A., Wool, A., & Ziskind, E. (2000). Fang: A firewall analysis engine. In *Security and privacy, 2000. s&p 2000. proceedings. 2000 ieee symposium on* (pp. 177–187).

Mell, P., & Grance, T. (2011). The nist definition of cloud computing.

Menascé, D. A. (2005). Virtualization: Concepts, applications, and performance modeling. In *Int. cmg conference* (pp. 407–414).

Miao, R., Yu, M., & Jain, N. (2014). Nimbus: cloud-scale attack detection and mitigation. In *Acm sigcomm computer communication review* (Vol. 44, pp. 121–122).

Mijumbi, R., Serrat, J., Gorricho, J.-L., Bouten, N., De Turck, F., & Boutaba, R. (2015). Network function virtualization: State-of-the-art and research challenges.

Mirkovic, J., & Reiher, P. (2004). A taxonomy of ddos attack and ddos defense mechanisms. *ACM SIGCOMM Computer Communication Review*, *34*(2), 39–53.

Moyer, P. R., & Schultz, E. E. (1996). A systematic methodology for firewall penetration testing. *Network Security*, *1996*(3), 11–18.

Negi, P., Mishra, A., & Gupta, B. (2013). Enhanced cbf packet filtering method to detect ddos attack in cloud computing environment. *arXiv preprint arXiv:1304.7073*.

Nikander, P., Gurtov, A., & Henderson, T. R. (2010). Host identity protocol (hip): Connectivity, mobility, multi-homing, security, and privacy over ipv4 and ipv6 networks. *Communications Surveys & Tutorials, IEEE*, *12*(2), 186–204.

NSTISSC. (2000). *National information systems security (infosec) glossary.* National Security Telecommunications and Information Systems Security Committee (NSTISSC).

Okada, K., Hazeyama, H., & Kadobayashi, Y. (2014). Oblivious ddos mitigation with locator/id separation protocol. In *Proceedings of the ninth international conference on future internet technologies* (p. 8).

Panko, R. (2010). *Corporate computer and network security, 2/e.* Pearson Education India.

Peltier, T. R. (2005). *Information security risk analysis.* CRC press.

Pfaff, B., Pettit, J., Amidon, K., Casado, M., Koponen, T., & Shenker, S. (2009). Extending networking into the virtualization layer. In *Hotnets.*

Popek, G. J., & Goldberg, R. P. (1974). Formal requirements for virtualizable third generation architectures. *Communications of the ACM*, *17*(7), 412–421.

Project, A. H. S. (n.d.). *Apache website: https://httpd.apache.org/.*

Qiu, L., Varghese, G., & Suri, S. (2001). Fast firewall implementations for software and hardware-based routers. In *Network protocols, 2001. ninth international conference on* (pp. 241–250).

Reuben, J. S. (2007). A survey on virtual machine security. *Helsinki University of Technology*, *2*, 36.

Ricciulli, L., Lincoln, P., & Kakkar, P. (1999). Tcp syn flooding defense..

Riggio, R., Rasheed, T., & Granelli, F. (2013). Empower: A testbed for network function virtualization research and experimentation. In *Future networks and services (sdn4fns), 2013 ieee sdn for* (pp. 1–5).

Saad, R., Nait-Abdesselam, F., & Serhrouchni, A. (2008). A collaborative peer-to-peer architecture to defend against ddos attacks. In *Local computer networks, 2008. lcn 2008. 33rd ieee conference on* (pp. 427–434).

Sahoo, J., Mohapatra, S., & Lath, R. (2010). Virtualization: A survey on concepts, taxonomy and associated security issues. In *Computer and network technology (iccnt), 2010 second international conference on* (pp. 222–226).

Savage, S., Wetherall, D., Karlin, A., & Anderson, T. (2000). Practical network support for ip traceback. In *Acm sigcomm computer communication review* (Vol. 30, pp. 295–306).

Schneier, B. (1997). Applied cryptography, 1996. *Cover and title pages*, 125–147.

Shameli-Sendi, A., Pourzandi, M., Fekih-Ahmed, M., & Cheriet, M. (2015). Taxonomy of distributed denial of service mitigation approaches for cloud computing. *Journal of Network and Computer Applications*, *58*, 165–179.

Sheth, C., & Thakker, R. (2011). Performance evaluation and comparative analysis of network firewalls. In *Devices and communications (icdecom), 2011 international conference on* (pp. 1–5).

Shin, S.-w., Kim, K.-y., & Jang, J.-s. (2005). D-sat: detecting syn flooding attack by two-stage statistical approach. In *Applications and the internet, 2005. proceedings. the 2005 symposium on* (pp. 430–436).

Shropshire, J. (2015). Hyperthreats: Hypercall-based dos attacks. In *Southeastcon 2015* (pp. 1–7).

Sitaraman, A., Mann, J. F., Dos Santos, M. A., Lou, S., & Bhasham, S. K. S. (2002, July 30). *Dynamic ip addressing and quality of service assurance.* Google Patents. (US Patent 6,427,174)

Smith, R. E. (2015). *Elementary information security.* Jones & Bartlett Publishers.

Snoeren, A. C., Partridge, C., Sanchez, L. A., Jones, C. E., Tchakountio, F., Kent, S. T., & Strayer, W. T. (2001). Hash-based ip traceback. In *Acm sigcomm computer communication review* (Vol. 31, pp. 3–14).

Spears, J. L. (2006). Defining information security. In *5th security conference, las vegas, nevada, the information institute, washington dc, usa.*

Srivatsa, M., Iyengar, A., Yin, J., & Liu, L. (2008). Mitigating application-level denial of service attacks on web servers: A client-transparent approach. *ACM Transactions on the Web (TWEB)*, *2*(3), 15.

Tariq, U., Malik, Y., Abdulrazak, B., & Hong, M. (2011). Collaborative peer to peer defense mechanism for ddos attacks. *Procedia Computer Science*, *5*, 157–164.

Thomas, T. M., & Stoddard, D. (2011). *Network security first-step.* Cisco Press.

Tupakula, U., & Varadharajan, V. (2003). Counteracting ddos attacks in multiple isp domains using routing arbiter architecture. In *Networks icon2003. the 11th ieee international conference* (pp. 455–460).

Venter, H., & Eloff, J. H. (2003). A taxonomy for information security technologies. *Computers & Security*, *22*(4), 299–307.

Verisign. (2014a). *Ddos protection services overview,.* https://www.verisign.com/assets/datasheet-ddos-overview.pdf?inc=www.verisigninc.com.

Verisign. (2014b). *Ddos protection services overview - what is ddos,.* http://www.nl-ix.net/docs/verisign/faq-ddos-protection-services.pdf.

Wang, G., & Ng, T. E. (2010). The impact of virtualization on network performance of amazon ec2 data center. In *Infocom, 2010 proceedings ieee* (pp. 1–9).

Wang, H., Zhang, D., & Shin, K. G. (2002). Detecting syn flooding attacks. In *Infocom 2002. twenty-first annual joint conference of the ieee computer and communications societies. proceedings. ieee* (Vol. 3, pp. 1530–1539).

Waziri Jr, I. (2014). A federated architecture for heuristics packet filtering in cloud networks.

Waziri Jr, I., Mirzoev, T., & Shropshire, J. (2014). Comparison of control-and hardware-based filtering architectures. *International Journal of Engineering Research & Innovation*, 34.

Waziri Jr, I., & Shropshire, J. (2015). A heuristic migration logic between firewalls in a federated cloud network. In *Southeastcon 2015* (pp. 1–8).

Wu, H., Ding, Y., Winer, C., & Yao, L. (2010). Network security for virtual machine in cloud computing. In *Computer sciences and convergence information technology (iccit), 2010 5th international conference on* (pp. 18–21).

Xiao, B., Chen, W., He, Y., & Sha, E. H. (2005). An active detecting method against syn flooding attack. In *Parallel and distributed systems, 2005. proceedings. 11th international conference on* (Vol. 1, pp. 709–715).

Xuan, D., Chellappan, S., Wang, X., & Wang, S. (2004). Analyzing the secure overlay services architecture under intelligent ddos attacks. In *Distributed computing systems, 2004. proceedings. 24th international conference on* (pp. 408–417).

Yang, X., Ma, T., & Shi, Y. (2007). Typical dos/ddos threats under ipv6. In *Computing in the global information technology, 2007. iccgi 2007. international multi-conference on* (pp. 55–55).

Yang, Z., Qiao, L., Liu, C., Yang, C., & Wan, G. (2010). A collaborative trust model of firewall-through based on cloud computing. In *Computer supported cooperative work in design (cscwd), 2010 14th international conference on* (pp. 329–334).

Yuan, J., & Mills, K. (2005). Monitoring the macroscopic effect of ddos flooding attacks. *Dependable and Secure Computing, IEEE Transactions on*, *2*(4), 324–335.

Zaborovsky, V. S., & Titov, A. (2009). Specialized solutions for improvement of firewall performance and conformity to security policy. In *Security and management* (pp. 603–608).

zadjmool. (2013). A risk based approach to ddos protection. *For Credit Unions and Credit Union Service Organizations*.

Zahid, M., Belmekki, A., & Mezrioui, A. (2012). A new architecture for detecting ddos/brute forcing attack and destroying the botnet behind. In *Multimedia computing and systems (icmcs), 2012 international conference on* (pp. 899–903).

Zalenski, R. (2002). Firewall technologies. *Potentials, IEEE*, *21*(1), 24–29.

APPENDIX

A. FIREWALL ACL CONFIGURATIONS

Both firewalls are configured in transparent mode. Depending on the type of device used, a guide on how to configure the firewall can be obtained from the vendor. Below are the rules we used to configure the firewalls. We blocked all incoming traffic, and only allowed authorized traffic to pass through (X represents public IP addresses, and XX represents authorized IP addresses.).

A.1 Hardware Firewall

- Config# access-list 110 deny tcp any host X.X.X.X
- Config# access-list 110 deny tcp any host X.X.X.X
- Config# access-list 110 deny udp any any eq 520
- Config# access-list 110 deny ip any host X.X.X.X
- Config# access-list 110 deny ospf any any
- Config# access-list 110 deny host X.X.X.X
- Config# access-list 110 deny tcp any any eq 21
- Config# access-list 110 deny tcp any any eq 22
- Config# access-list 110 deny tcp any any eq 25
- Config# access-list 110 deny tcp any any eq 110
- Config# access-list 110 deny tcp any any eq 143
- Config# access-list 110 deny udp any any eq 135
- Config# access-list 110 deny tcp any any eq 445
- Config# access-list 110 deny tcp any any eq 1434

- Config# access-list 110 deny tcp any any eq 4444
- Config# access-list 110 deny tcp any any eq 4899
- Config# access-list 110 permit icmp any any
- Config# access-list 110 permit tcp any host XX.XX.XX.XX
- Config# access-list 110 permit tcp any host XX.XX.XX.XX
- Apply to inbound traffic on ethernet0/0 (outside interface)
- Config# access-group 110 in interface outside

A.2 Virtual Firewall

- Rule 1:

  Vyatta# set firewall name FWRULES-1 rule 1 action reject

  Vyatta# set firewall name FWRULES-1 rule 1 source address X.X.X.X

  Vyatta# set firewall name FWRULES-1 rule 1 protocol TCP

- Rule 2:

  Config# set firewall name FWRULES-1 rule 2 action reject

  Config# set firewall name FWRULES-1 rule 2 source address X.X.X.X

  Config# set firewall name FWRULES-1 rule 2 protocol TCP

- Rule 3:

  Config# set firewall name FWRULES-1 rule 3 action reject

  Config# set firewall name FWRULES-1 rule 3 protocol UDP

  Config# set firewall name FWRULES-1 rule 3 destination port 520

- Rule 4:

  Config# set firewall name FWRULES-1 rule 4 action reject

  Config# set firewall name FWRULES-1 rule 4 source address X.X.X.X

  Config# set firewall name FWRULES-1 rule 4 protocol IP

- Rule 5:

```
Config# set firewall name FWRULES-1 rule 5 action reject
Config# set firewall name FWRULES-1 rule 5 protocol OSPF
```

- Rule 6:

```
Config# set firewall name FWRULES-1 rule 6 action reject
Config# set firewall name FWRULES-1 rule 6 source address X.X.X.X
Config# set firewall name FWRULES-1 rule 6 protocol TCP
```

- Rule 7:

```
Config# set firewall name FWRULES-1 rule 7 action accept
Config# set firewall name FWRULES-1 rule 7 protocol TCP
Config# set firewall name FWRULES-1 rule 7 destination port 80
```

- Rule 8:

```
Config# set firewall name FWRULES-1 rule 8 action reject
Config# set firewall name FWRULES-1 rule 8 protocol TCP
Config# set firewall name FWRULES-1 rule 8 destination port 21
```

- Rule 9:

```
Config# set firewall name FWRULES-1 rule 9 action reject
Config# set firewall name FWRULES-1 rule 9 protocol TCP
Config# set firewall name FWRULES-1 rule 9 destination port 22
```

- Rule 10:

```
Config# set firewall name FWRULES-1 rule 10 action reject
Config# set firewall name FWRULES-1 rule 10 protocol TCP
Config# set firewall name FWRULES-1 rule 10 destination port 25
```

- Rule 11:

```
Config# set firewall name FWRULES-1 rule 11 action reject
Config# set firewall name FWRULES-1 rule 11 protocol TCP
Config# set firewall name FWRULES-1 rule 11 destination port 110
```

- Rule 12:

```
Config# set firewall name FWRULES-1 rule 12 action reject
Config# set firewall name FWRULES-1 rule 12 protocol TCP
Config# set firewall name FWRULES-1 rule 12 destination port 143
```

- Rule 13:

```
Config# set firewall name FWRULES-1 rule 13 action reject
Config# set firewall name FWRULES-1 rule 13 protocol UDP
Config# set firewall name FWRULES-1 rule 13 destination port 135
```

- Rule 14:

```
Config# set firewall name FWRULES-1 rule 14 action reject
Config# set firewall name FWRULES-1 rule 14 protocol TCP
Config# set firewall name FWRULES-1 rule 14 destination port 445
```

- Rule 15:

```
Config# set firewall name FWRULES-1 rule 15 action reject
Config# set firewall name FWRULES-1 rule 15 protocol TCP
Config# set firewall name FWRULES-1 rule 15 destination port 1434
```

- Rule 16:

```
Config# set firewall name FWRULES-1 rule 16 action reject
Config# set firewall name FWRULES-1 rule 16 protocol TCP
Config# set firewall name FWRULES-1 rule 16 destination port 4444
```

- Rule 17:

```
Config# set firewall name FWRULES-1 rule 17 action reject
Config# set firewall name FWRULES-1 rule 17 protocol TCP
Config# set firewall name FWRULES-1 rule 17 destination port 4899
```

- Rule 18:

```
Config# set firewall name FWRULES-1 rule 18 action accept
Config# set firewall name FWRULES-1 rule 18 protocol ICMP
```

- Rule 19:

```
Config# set firewall name FWRULES-1 rule 19 action accept
Config# set firewall name FWRULES-1 rule 19 source address XX.XX.XX.XX
Config# set firewall name FWRULES-1 rule 19 protocol TCP
```

- Rule 20:

```
Config# set firewall name FWRULES-1 rule 20 action accept
Config# set firewall name FWRULES-1 rule 20 protocol TCP
Config# set firewall name FWRULES-1 rule 20 destination address XX.XX.XX.XX
```

- Apply to interface and commit:

```
Config# set interfaces ethernet eth1 firewall in name FWRULES-1
Config# commit
```

- To show firewall rules:

```
Config# show firewall name FWRULES-1
```

- To show rules on interface:

```
Config# show interfaces ethernet eth1 firewall
```

A.3 Complete Architecture

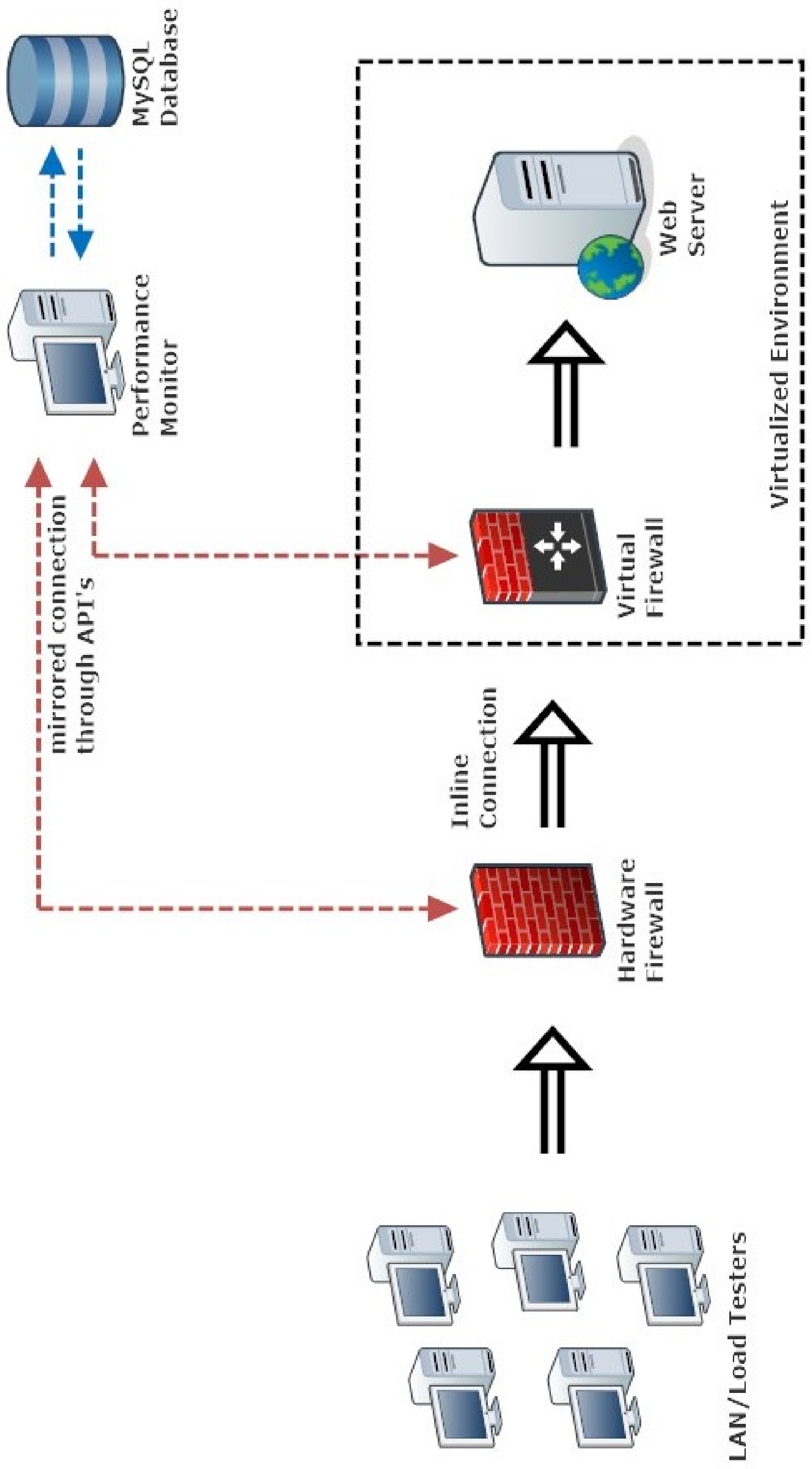


Fig. A.1. Complete Architecture I

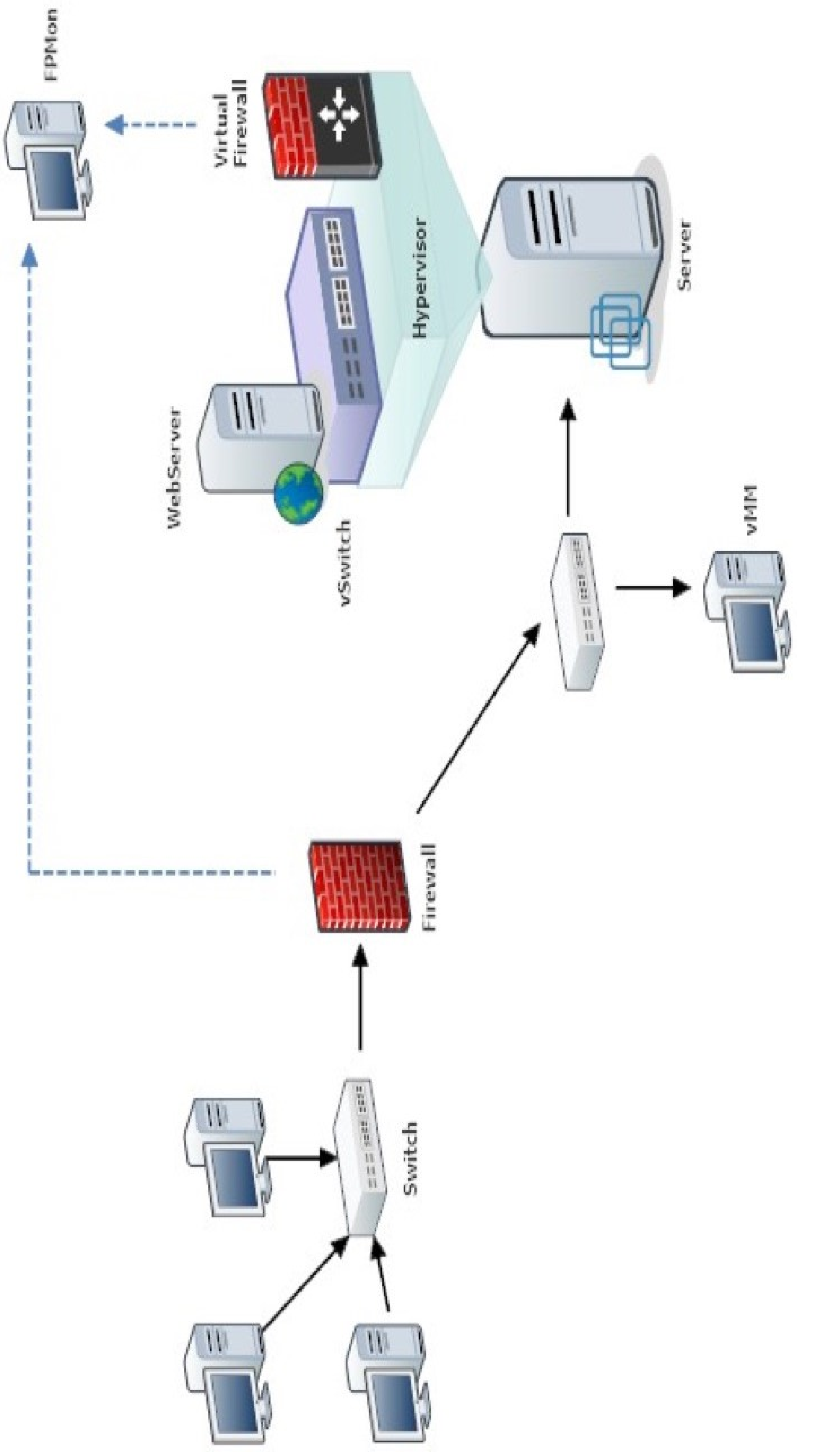


Fig. A.2. Complete Architecture II

VITA

VITA

Ibrahim Waziri, Jr. received his Ph.D. in Information Security from Purdue University in August 2016. He received his Masters Degree in Applied Engineering (Information Technology) from Georgia Southern University and a Bachelors Degree (WES Equivalent) in Electronics & Computers Engineering from Federal Polytechnic Bauchi. Ibrahim's research interests are in the area of Computer Networks and Virtualization Security.

Ibrahim has worked as a research assistant on Network Security and Cloud Computing, as an IT specialist (Graduate Assistant) for Purdue University Graduate School, and also as a teaching assistant for Network Security and Introduction to IT classes. Ibrahim has worked with several organizations, including RSA-The Security Division of EMC as a Cyber Anti-Fraud Analyst working on phishing attacks & malware analysis. In the summer of 2015, Ibrahim interned as a Cyber Security Analyst for the US International Trade Commission. Detailed information and a complete vita is available on his website at iiwaziri.com.