SECURITY ASSESSMENT OF SELECT COMPUTER SYSTEMS UNDER DISTRIBUTED DENIAL OF SERVICE ATTACKS PERFORMED BY BOTNETS

A Thesis

by

HUGO ARMANDO HERRERA MARTINEZ

Submitted to the Graduate School of
The University of Texas Rio Grande Valley
In partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE ENGINEERING

May 2019

Major Subject: Electrical Engineering

SECURITY ASSESSMENT OF SELECT COMPUTER SYSTEMS UNDER DISTRIBUTED DENIAL OF SERVICE ATTACKS PERFORMED BY BOTNETS

A Thesis
by
HUGO ARMANDO HERRERA MARTINEZ

COMMITTEE MEMBERS

Dr. Sanjeev Kumar
Chair of Committee

Dr. Wenjie Dong
Committee Member

Dr. Yoonsu Choi
Committee Member

May 2019

ABSTRACT

Herrera Martinez, Hugo Armando, Security Assessment of Select Computer Systems Under Distributed Denial of Service Attacks Performed by Botnets, Master of Science (MS), May, 2019, 157 pp., 2 tables, 112 figures, 109 references.

With the passage of time Internet connectivity has been found on more electronic devices than in previous decades. This continuous development may be perceived as a double-edged sword. While the world becomes more Internet-connected and quality of life increases, the lack of proper security protocols in some of these electronic devices permits illegitimate parties to take control of them for their own malicious purposes.

This thesis studies the impact of compromised Internet-connected devices on commonly used server operating systems: Microsoft's Windows Server 2012 R2 Datacenter, Windows Server 2016 Datacenter, and Apple's macOS 10.13.6 High Sierra deployed on Apple's Mac Pro Server (Mac Pro Mid2010). Their performance under prevalent Distributed Denial of Service Attacks will be evaluated at equal attack traffic loads while utilizing three networks of simulated botnets. While the server was under attack connection rate, network traffic, processor utilization, processor temperature, and memory utilization were evaluated for computer systems under consideration.

DEDICATION

The completion of my master's studies would not have been possible without the love, and support of my family. I would like to dedicate my thesis to my parents, Hugo Armando Herrera Peña, Dora Alicia Herrera Martinez, for their support, patience and unconditional love, to my brother Carlos Alejandro Herrera, and my sisters Cecilia Herrera and Sofia Herrera, by whom I continue to become motivated to become a better version of myself, to be an exemplary role model for you all. I would also like to thank my grandmother Cecilia Renteria Ortiz and my aunt Veronica Renteria Ortiz for their support, and faith in me. All your love and support has enriched me immensely and with it I continue to find the strength, and determination to complete this degree and look forward to the next chapter of my life.

ACKNOWLEDGMENTS

I would like to formally express my gratitude:

Dr. Sanjeev Kumar, chair of my thesis committee, advisor, for all his patience, advice and faith in my abilities I will always be grateful to him. It has been a great pleasure learning from his lectures. I now have a greater understanding of the fundamental of networking, and security in virtual environments. His advice, input, and comments, on my thesis helped to ensure the quality of intellectual work. Thank you so much for all your encouragement and support.

I would also like to thank my dissertation committee member Dr. Wenjie Dong, and Dr. Yoonsu Choi, for their willingness to serve as committee members and their considerations.

My deepest appreciation goes the faculty of Electrical Engineering at the UTRGV for their confidence in me to become a graduate assistant for the Electrical Engineering department. Thank you for giving me this great opportunity. Additionally, I thank the technology department in the Edinburgh Independent School District for the opportunity to work with them as technology intern. Experiences with both institutions allowed me to grow as a professional and were influential in allowing me to fund my graduate studies. To my peers and friends during graduate school, for all the technical discussion we had during this journey. Thank you!

Work in this thesis is based upon the grants awarded to Dr. Kumar by the National Science Foundation (NSF) under Grant No. 0521585 and the Bentsen Lloyd Jr. Endowment Chair in Engineering Fellowship.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER I

INTRODUCTION

With the passage of time, our world has become more Internet connected than ever before. All of this is owed to the rise and the perpetual innovation of the Internet. In the last decade, it has become commonplace to live surrounded by electronic devices with Internet accessibility also known as Internet of Things (IoT) devices. In a world that continues to become reliant on sophisticated web-based solutions, the topics of computer and network security have gained significant importance. Often personal data ranging from the mundane to sensitive information travels from our platforms and into the cloud back and forth while the user trusts on the reliability of the service. The reliability and security provided by the service is dependent on the server's hardware and software controlling the data sent and received. The software in these servers must manage their hardware resources in an efficient manner to maintain the reliability of the service. IoT devices are defined as any object able to transmit or receive data automatically through the Internet, they provide an increase in quality of life, as they share more of our personal information than ever before, however the security of our data and these devices is rarely certain. With the rise of IoT devices and the increase of quality of life it must be worth noting that also a new level of risk rises with it, due to the magnitude of interconnectedness the consequences of known risk intensifies. Individuals may use exploits in large segments IoT devices to gain control of them, thus recruiting the devices into a botnet with the intent of attacking other computers or devices for malicious intent [1].

Distributed Denial of Service (DDoS) attacks are often carried out by botnets exploiting a vulnerability in a system or protocol. This type of cyber-attack aims to prevent legitimate users from accessing information on systems, devices, or other network resources. This attack can disrupt services such as online accounts used by banks, e-mail, websites, amongst other services dependent on computers or networks. The most common form of DDoS consists of flooding a network server with illegitimate traffic, per protocols the system under attack will attempt to respond or acknowledge all incoming traffic requests unbeknownst to it whether they are legitimate or not. The DDoS attack aims to consume the systems resources: processor power, memory, and bandwidth thus preventing legitimate users from accessing the system. While under attack even with help of security systems such as firewalls, Intrusion Detection Systems (IDS), and Intrusion Prevention System (IPS) the system may become unresponsive and in the worst-case scenario crash due to severity of the attack utilized, which can result in loss of data amongst other difficulties [2-4].

Computer security is originally defined as the protection of an automated information system objectively conserving confidentiality, integrity, and availability, which embodies the fundamental security objectives for data, information and computing services. This is known as the CIA Triad [4-5]. Under these attacks the availability, and integrity are immediately compromised furthermore if DDoS attacks are used in conjunction with other forms of cyber-attacks confidentiality can also be lost.

1.1 Motivation

In a world that is constantly evolving and becoming more Internet centric, security mechanism for all platforms have a social obligation to raise the security in all their platforms, similarly the user base also has the social obligation of securing their platforms in the best

conceivable manner. In a report from Statista it is mentioned that the expected number of IoT connected devices installed worldwide will reach 75.44 billion by 2025 [6]. Some examples to name a few implicate LED lighting, thermostats, locks, home security systems (sensors, cameras), vehicle tracking systems, health and fitness wearables among many others. All these devices with lack of proper security protocol can be prone to leak personal information, act in erratic or malicious manners all while potentially participating in the global threat of DDoS attacks. This are all legitimate devices, which can be distributed far across the globe so blocking traffic from a certain segment of the Internet or a country would not be an ideal solution in mitigating the attack.

While botnets and Distributed Denial of Service (DDoS) attacks have been around as early as the 80s their presence and ever refining development continues to be a major threat to network and computer security [3]. Early on October 2017 Symantec discovered the malware Sockbot. This malware disguises itself as several apps on Google Play pretending to provide a service while performing other illegitimate tasks in the background. Installations range from 600,000 to 2.6 million devices. Originally the malware was used to illegitimately generate ad revenue, but with this malware it is also possible to recruit devices into a botnet and force them to participate in DDoS attacks. The apps were reported to google and were removed from their service [7]. On October 19, 2017 a botnet named Reaper (or IoTroop) was discovered thanks to the efforts from the Chinese security firm Qihoo 360 and the Israeli firm Check Point. Reaper (not to be confused with the The Reaper botnet from South Korea) has been described as a botnet with a destructive potential higher than the Mirai botnet from 2016 [8,11]. The perpetrators behind the Mirai botnet took advantage of unsecured IoT devices such as IP cameras and Internet routers. The rise of the Mirai botnet can be widely attributed to human carelessness, the culprits

behind it attained their botnet by scanning large sectors of the Internet to locate open Telnet ports and utilizing default username and passwords to gain access thus recruiting a myriad IoT devices in the million range. Notably the Mirai botnet was responsible for the DDoS attacks which prevented the U.S. east coast from having any Internet accessibility [8-10]. Reaper poses a greater threat as it utilizes the same methodology behind the Mirai botnet and improves on it. Reaper will attempt to access IoT devices in the same fashion as the Mirai botnet, however devices that are not accessible thru this method will be subject to other forms of exploits and cyber-attacks in order to find a weak point allowing access, thus having potential for recruiting far more devices the Mirai botnet did during the events of 2016 [8,11].

Regularly attackers employing DDoS attack target servers where downtime and communication failures can lead to loss of revenue, reputation or simply generate chaos for a plethora or reasons. These attacks are often aimed at gaming, retail, political and/or governmental servers. The CIA website (not to be confused with the CIA Triad) was taken down on the third week of June 2011, by the LulzSec hacking group twice in the same week [12]. Since then two members of the group have been found guilty of cyber-attacks against the CIA, Sony, Fox Entertainment Group, the Arizona State Police and Britain Organized Crim Agency (SOCA) [13]. During the third quarter of 2017 three high profile cases made headlines. On August 13, Blizzard Entertainment reported an ongoing DDoS attack affecting users in online videogames. During the attack servers for World of Warcraft, Overwatch, amongst other Blizzard franchises, were affected preventing players from accessing their accounts as a result of the attack. A responsible party is yet to be found. It is suspected that the attack was performed with the similar goals as those of Lizard Squad which, in 2014, crippled and immobilized Sony's PlayStation network and Microsoft's Xbox Live [14]. The Winning Poker Network suffered

from DDoS Attacks, early in the month of September. The attacks interrupted the series and forced a temporary cancellation of tournaments until conditions improved [15]. On September 30, 2017, the UK National Lottery was subject to DDoS attacks lasting 90 minutes during a peak time preventing players from placing their stakes online, resulting in a high loss of revenue with no guilty party being recognized [16]. On October 21, 2016 Dyn one of the biggest DNS companies and now subsidiary of Oracle was heavily impacted by the attacked performed by the Mirai botnet. During the attack on their networks several prominent entities of the Internet where taken offline. Amongst them are the names of Netflix, Amazon, Spotify, GitHub, CNN, Indeed, PlayStation Network, Xbox, Twitter, Etsy, and many others. The attacks compromised the DNS hosts of Dyn which resulted in the loss of access of these and many other web sites utilizing Dys Host [9-10, 17-18].

According to a two-year collaborative effort by SDCS's (San Diego Supercomputer Center) CAIDA (Cooperative Association for Internet Data Analysis) team it was revealed that between March 2015 and February 2017 more than 20 million DoS attacks targeted approximately 2.2 million Internet addresses. This number is roughly one third of 6.5 million active connections on the Internet. Based on their data nearly 30,000 attacks per day occurred during the two-year study [19-20]. According to the quarterly DDoS reports from Kaspersky Labs, the most common form of DDoS attack in 2018 are TCP SYN Flooding followed by UDP Flood, HTTP based attacks, TCP, and ICMP based attacks respectively [21-23].

In order to mitigate the potential damage of these cyber-threats, it is prudent to have several layers of defense rather than a single point. A single line of defense upon failure would compromise the flow of traffic. Multilayer security systems involving Intrusion Detection Systems (IDS), Intrusion Prevention System (IPS), Firewalls, routers and switches are all that

stands between the endpoint (servers, desktops, and IoT devices) and the Internet, however they are not often available to the general public and a certain training and expertise is required in order to gain full advantage from them. Therefore, the focus of this thesis will be on the security already built into commonly used server environments. The evaluation will be based on the Operating System (OS) performance under duress from the attack while all security updates are installed, and the built-in firewall is enabled. This will allow for a performance comparison which will determine which environment has higher resiliency and efficiency during the attack without the aid of any external security systems.

The operating systems in this evaluation are deployed on Apple's Mac Pro Server (Mac Pro Mid2010) which can run Microsoft's Windows Server 2012 R2 Datacenter, Windows Server 2016 Datacenter, and Apple's macOS 10.13.6 High sierra. Each operating system will run its own web server to measure network connectivity. Apple's Mac Pro Server (Mac Pro Mid2010) was chosen to maintain consistency on hardware specifications, to ensure the validity of the results upon comparison. The attacks performed for this thesis are processor intensive and are carried out by three distinct simulated botnet sizes as follows: 254 simulated bots, 65 thousand simulated bots and 16 million simulated bots. This number of bots was chosen to replicate they idea of a three different network sizes commonly used those being network with Class C, B, and A IP addressing. The operating systems will be attacked by TCP SYN Flood, UDP Flood, ICMP Echo request flood (Ping Flooding attack), and ICMP Echo reply flood (Smurf attack).

1.2 Problem Statement

Due to impending threat of Denial of Service (DoS) attacks, and the constant development of botnets, computer security cannot be guaranteed across the Internet all the time. Botnets are generally constituted by varying number of compromised systems ranging from the

hundreds to the millions, these systems grant an attacker with a high amount of distributed computing and storage power. Proper resource allocation is one of the main goals of the operating system, this is done to promote efficiency in computational tasks. When a server is targeted by a botnet with a cyber-attack resources will begin to deplete due to exploits in network protocols that have existed since the early stages of the Internet. As the operating system attempts to defend itself through built in security mechanism it will also attempt to satisfy user demands. Therefore, it is of utmost importance to understand the impact of these cyber-attacks and the methods used to mitigate them. In this thesis Microsoft's operating systems Windows Server 2012 R2 Datacenter, Windows Server 2016 Datacenter, and Apple's macOS 10.13.6 High Sierra are tested against four different types of commonly used Distributed Denial of Service (DDoS) attacks while utilizing three separate simulated botnets. The experiments conducted will help understand the impact on performance of a server under attack conditions and will permit a comparison between the revealed platforms.

1.3 Thesis Outline

In this thesis, the security and performance of three popularly deployed operating systems used for web servers is consider while under the effect of four popular forms of cyber-attacks inside of a control environment which utilizes three simulated botnet sizes replicating compromised Class A, B, and C IP networks. The Distributed Denial of Service (DDoS) attacks will be targeted against Apple's Mac Pro Server (Mac Pro Mid2010) while the following operating systems are deployed: Microsoft's Windows 2012 R2 Datacenter, Windows 2016 Datacenter, and Apple's macOS 10.13.6 High Sierra. The impact of DDoS attacks ICMP Ping Flooding attack, ICMP Smurf attack, TCP SYN Flooding, and UDP Flooding is studied when applied to the hardware and software mentioned above.

The cyber-attacks simulated in this thesis are tested at different transmission rates with a starting line rate of 100 Mbps and increasing by 100 Mbps until the 1 Gbps mark is reached, where each iteration of the attack will last 5 minutes and is applied using all three botnet configurations mentioned before separately. Several tools ranging from terminal commands, performance and temperatures monitors were utilized to evaluate the performance and effectiveness of the operating systems while under the effects of the cyber-attack.

The organization for this thesis is as follows: Chapter I is an introduction dedicated to the influence the rapid growth Internet of Things (IoT) devices, has on cyber security threats DDoS attacks, botnets and how they affect everyday life. Chapter II provides for a broad understanding of the DDoS attacks utilized in this thesis and the background of botnets. Chapter III and IV describe the experimental setup used in this thesis and the results collected in the experiments performed in the closed environment at the Network Research Lab. More precisely, Chapter III pertains to the evaluation and comparison of Microsoft's Windows 2012 R2 Datacenter and Windows 2016 Datacenter, and their impact on the Internet Information Service (IIS) web server while under the influence of the DDoS attacks specified in this thesis. Furthermore, Chapter IV parallels the former with Apple's macOS 10.13.6 High Sierra, and its impact on the Apache 2.0 web server while under the influence of the DDoS attacks specified in this thesis. Chapter V links the data collected from experiments in all software platforms for one concluding comparison. Chapter VI concludes with a synopsis of all work performed.

CHAPTER II

DISTRIBUTED DENIAL OF SERVICE ATTACKS

A Denial of Service (DoS) attack aims to disrupt a system (server and/or network) by exhausting the victim's resources such as CPU power, Memory available, and network bandwidth, thus degrading the quality and/or availability of a service provided by the system. Furthermore, several other data structures such as file handles, transmission control blocks, process slots, etc. may be exhausted while under attack. This is achieved by transmitting a single attack traffic message, or an attack message stream resulting in the overloading of a system causing interruption of services, slow operational speeds, system crashes, and none to limited connectivity with intended users in a relative short amount of time. Unlike DoS attacks which utilize a single system to be carried out, a Distributed Denial of Service (DDoS) attack is coordinated active cyber-attack performed by multiple compromised Internet capable computing systems with the same end goal as a DoS attack. The resource consumption rate is heavily dependent on network traffic type, volume of attack traffic, and the victim's processing power.

To perform an DDoS attack, the perpetrator will utilize sophisticated tools and exploits in order to collect data on a network to find vulnerable systems to enlist into a botnet (A series of compromised computing systems); alternatively, a botnet could be hired or purchased. Once the perpetrator has control of a botnet, a command and control center will organize the bots and instruct them to send attack traffic into the victim without the knowledge of the end user of the compromised systems [3, 24-36].

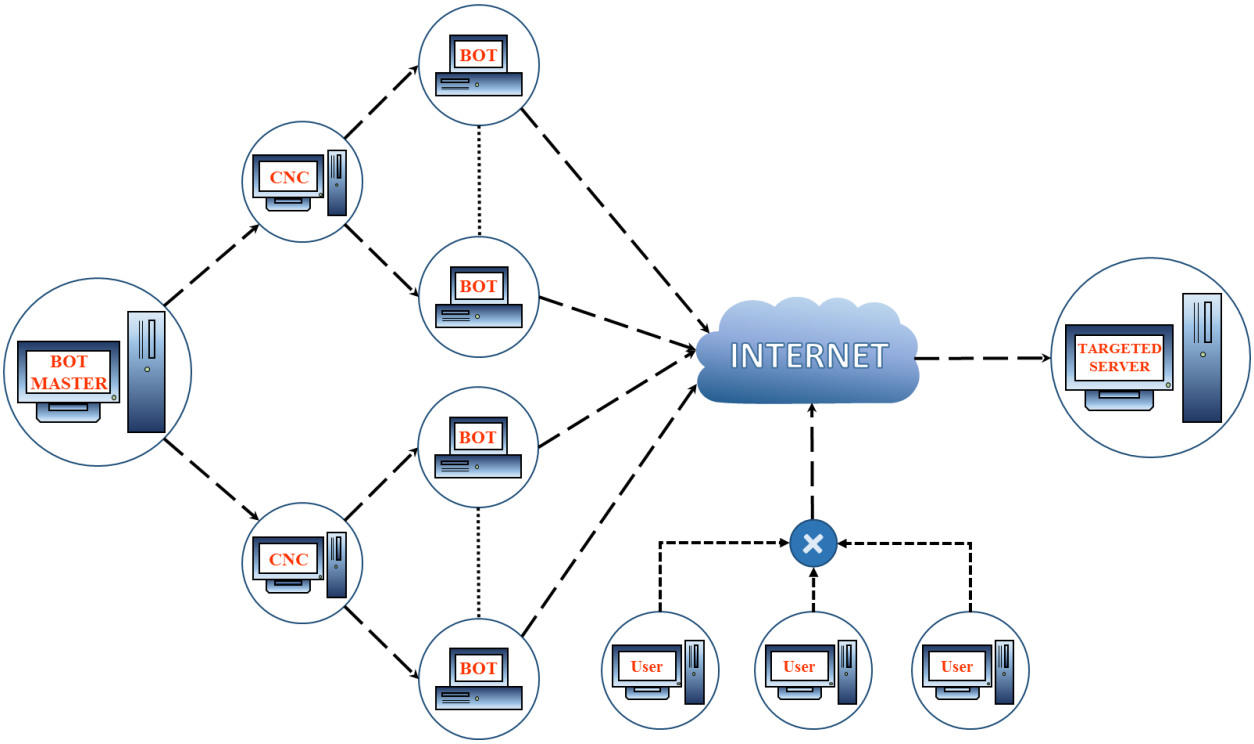


Figure 2.1 DDoS Attack Layout

Innovations on the Internet have connected millions across the globe, but with a higher degree of interconnectedness, a high degree of risk can be experienced online. Botnets and Distributed Denial of Service (DDoS) attacks, have evolved in parallel with the advancements in network technology. One of the earliest Denial of Service (DoS) attacks to make headlines was perpetrated as an accident at the Massachusetts Institute of Technology (MIT) by Robert Tapan Morris on November 1988 a year before the release of the World Wide Web; roughly only 60,000 computers had access to the Internet at this time. Morris's stated goal was to develop a worm to map the Internet, the worm would propagate thru systems on the Internet and report back from each system infected. The worm would travel to a system, check if a version of itself existed, if it did not exist the worm would replicate itself onto the system and continue searching for more. However, one fatal flaw in Morris's code permitted the indiscriminate replication of

the worm. Morris was concerned with defensive mechanism blocking the worm, to bypass this one out seven times the worm would install itself regardless of another copy of the worm existing or not. Over time several thousands of instances of this worm where active at the same time consequently slowing down several systems across the Internet. To undo the excessive damage all computers involved had to be quarantine and re-initialized effectively taking down the Internet [37, 38]. Decades after the events of the Morris Worm on September 2016, a great portion of the US East Coast loses connectivity to the Internet due to cyber-attacks on DNS servers by the Mirai botnet which at its core utilized the new unsecured technologies in IoT devices to gain access to them and attacked Dyn DNS servers as discussed in Chapter I [8-10]. These two cases exemplify how despite years of technological advances, DDoS attacks remain as a prominent threat to organizations and the end-users of the Internet. Several studies have been performed to understand the impact of these attacks [24-36].

2.1 Botnets

Illegitimate botnets are one of the more prominent threats surrounding the Internet and anyone accessing it. There are several reasons why an individual or a group may decide to use an illegal botnet often hacktivist will attempt to disrupt a group with a different view on an issue. Botnet is a term derived from the words robot and network, fundamentally botnets are a group of Internet capable devices such as simple IoT devices, (sensors, cameras, lights, etc.) complex servers, and desktop systems whose security has been compromised through a malware infection thus permitting an unidentified agent to gain control of the targeted systems where each compromised system on the botnet is known as a bot. Botnets are also available for hire or purchase and can be designed to perform criminal and malicious activities including bot

proliferation, click fraud, data theft, sending ransomware, spam, malware, and Distributed Denial of Service (DDoS) attacks [39-43].

According to Kaspersky Lab's DDoS Intelligence Report for the third quarter of 2017, a botnet with several hundred thousand bots at its prime known as WireX, was extinguished. WireX utilized Google Play applications to replicate undercover on Android devices [44]. There are several other methods for the propagation of bots such as exploits in website vulnerabilities, Trojan horses, cracking into systems with weak or no authentication similarly to the case described in Chapter I with the Mirai botnet from 2016 [8-10, 39, 40]. With the methods stated previously access is granted to the targeted systems resulting in the installation of malware to obtain control and thus enlisting a new bot into the botnet. Once the system has been enlisted there is a chance it will attempt to enlist more system in its network [40].

2.1.1 Botnet Topologies

The structure of a Botnet can variate depending on its intended purpose. The bot master (or bot herder) must decide which topology would benefit the botnet more. A topology is chosen based on commercial defenses, legal shutdowns, hijacking attempts, and each topologies strengths and weaknesses [40-41]. Botnet topologies can be classified in two categories: command and control center (CnC) based and peer-to-peer (P2P) based.

2.1.1.1 Command and Control Center based topologies

The command and control center topologies replicate the method of communication found on client/server models where a bot communicates directly or indirectly to a Command and Control center (CnC) for instructions, in the same fashion a remote workstation communicates with central server (or servers) for data. Typically, a bot will be preconfigured to

signal a specified CnC, where it will register itself as a new member of the botnet and await further commands. It is possible for more than one CnC to issue out instructions to the bots, however under this type of topology a communication channel is maintained among CnCs. The CnC may be owned by the attacker, or it may be an infected device the botnet master utilizes to perpetrate a cyber-attack. CnC based topologies encountered where often found to match one of the following types: Star, Multi-server, Hierarchical [40,41].

Botnets employing star topologies are defined by their use of a single CnC. The single CnC is responsible for communicating instructions to all bot agents. The nature of star topologies allows for direct communication between the CnC and all bot agents thus permitting the rapid transfer of instructions and/or stolen data. Star topologies have a single point of failure, if the CnC is blocked, or disabled the entire botnet goes offline [41].

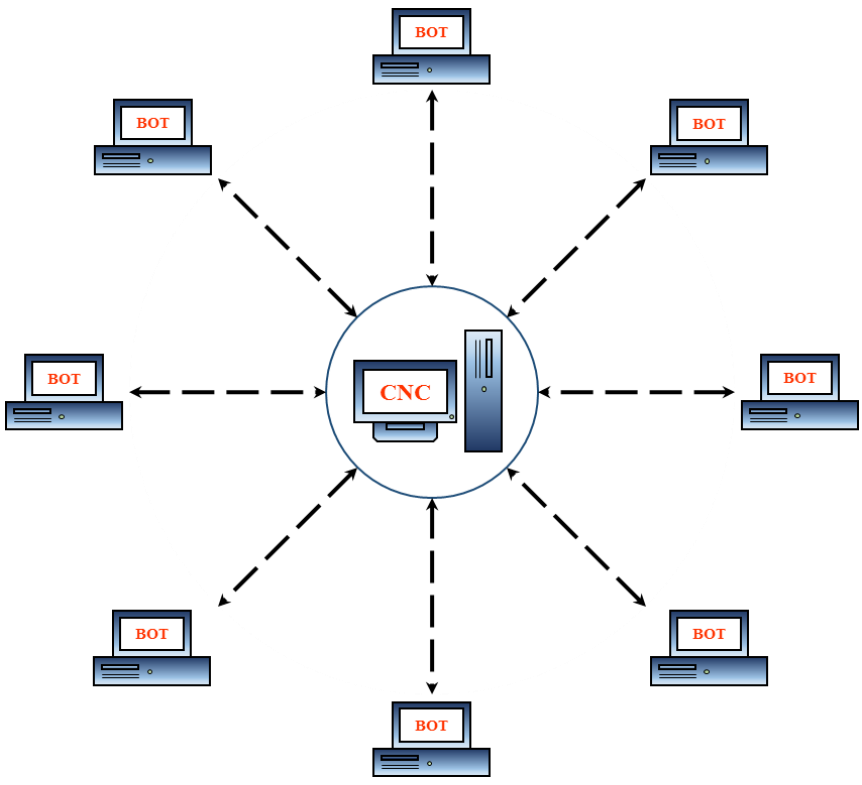


Figure 2.2 Star Topology Botnet

Multi-server topologies develop on top of existing star topologies. To construct the infrastructure for a multi-server botnet additional planning and effort is required. Botnets utilizing a multi-server topology are managed thru a collection of CnCs, which must maintain communication among themselves. Typically, multi-server botnet topologies feature two or more CnCs thus having an advantage over star topologies by lacking a single point of failure. When an individual CnC is blocked or disabled, the remaining CnCs will issue instructions and act on behalf of the missing CnC. Furthermore, multi-server topologies permit for the distribution of CnCs across the globe, resulting in a stronger, and more robust network featuring higher resilience to legal shutdown requests and the possibility for geographical optimization thus speeding up communication between remote bot agents and the botnet [41].

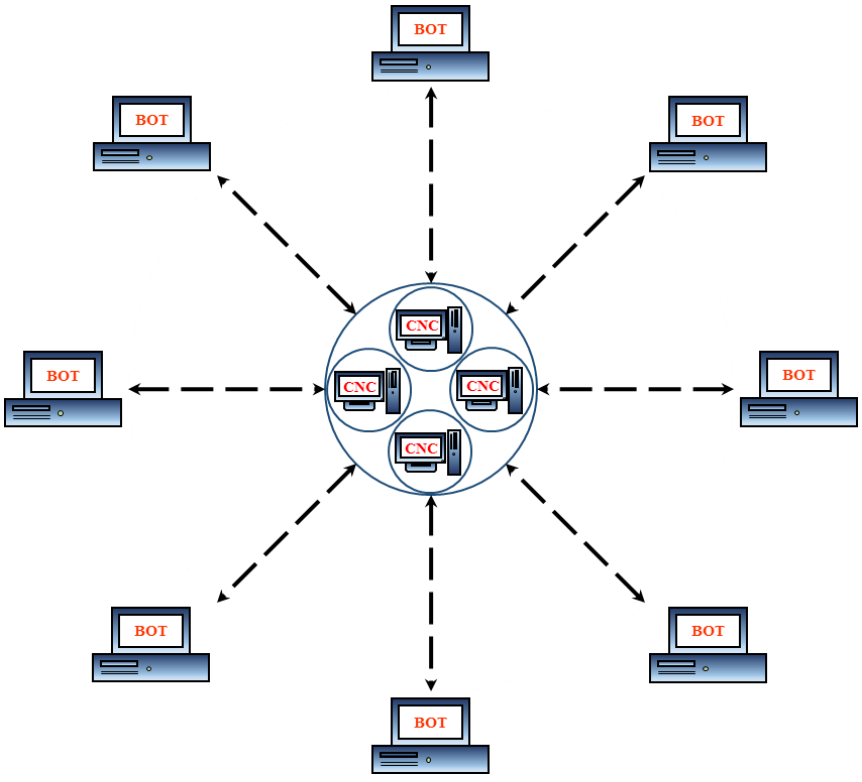


Figure 2.3 Multi-server topology botnet

Hierarchical topologies rely on the ability of bot agents to delegate new CnC instructions to bot agents at lower hierarchical levels. Hierarchical topologies prevent researchers from enumerating the location of all member within a botnet since the location of the entire botnet is not known by a single bot agent. The hierarchical distribution of CnC and bot agents allows for the division of a large botnet into several sub-botnets which can be rented or sold. When compared to star and multi-server topologies, hierarchical topologies feature increased security against exposure and hijack attempts since no single member knows the location of all botnet agents, however a high degree of latency may be experience by the botnet master. The higher latency is attributed to the multiple branches within the botnet a command must go thru before it reaches the final members in the hierarchy. A delay in the instructions issued can complicate certain cyber-attacks [41].

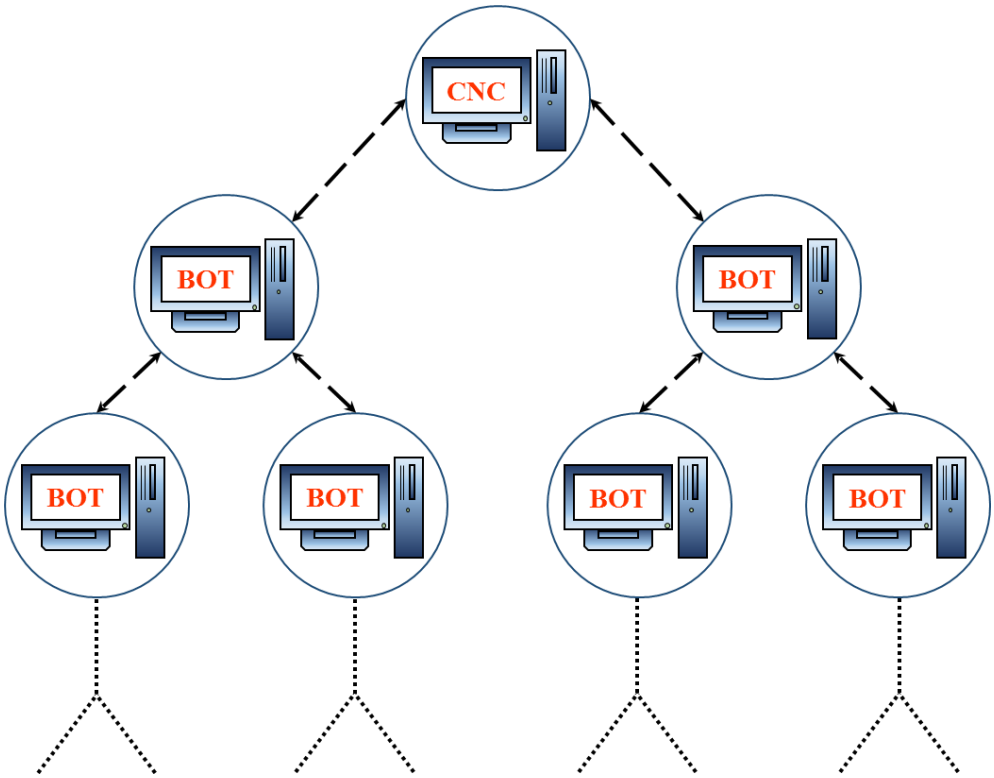


Figure 2.4 Hierarchical topology botnet

2.1.1.2 Peer-to-peer/Random topologies

Botnets with a Peer-to-peer (P2P) or random topologies lack traditional CnCs, instead each bot agent may act as both an agent and a CnC. Communication between the bot master and the botnet is achieved by sending signed commands from a bot agent to other agents in neighboring nodes (list of trusted systems) where the next set of bot agents will replicate the procedure and communicate the instructions to its own set neighboring nodes. Typically, a botnet with a random topology will implement several cryptography tools to sign instructions and protect against a loss of control.

Due to the lack of a centralized CnC infrastructure, this type of botnet is more resilient against shutdown attempts, however identifying the members of the botnet is simpler by observing a single member of the botnet and analyzing the systems it communicates with. Additionally, botnets with random topologies experience a high degree of latency (less when compared to Hierarchical topologies) due to the nature by which each bot agent is reached by the previous one [40,41].

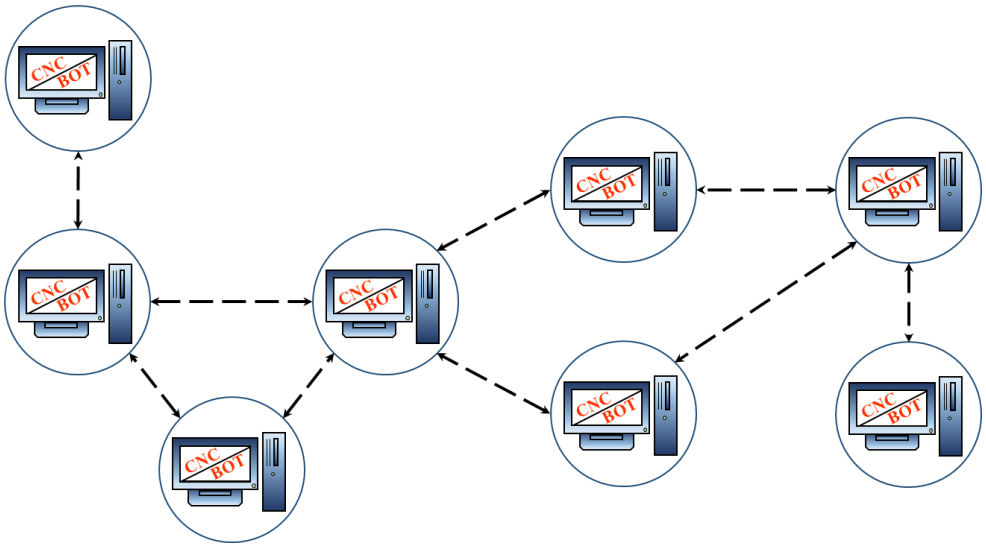


Figure 2.5 Peer-to-peer (P2P)/Random topology botnet

2.1.1.3 Defensive techniques to avoid becoming part of a botnet

Becoming part of a botnet comes with a set of unintended consequences, some risks include: High Internet bills (if on a data plan), sluggish and unstable system performance, potential legal implications, and stolen personal data which could result in identity theft. To avoid becoming part of a botnet it is important to adhere by the best-known practices in network security. Exercise caution when downloading files from the Internet, regularly update Internet security software, allow only trusted execution of third-party code, maintain secure passwords, perform periodic system wipe/restores, implement good ingress and egress filtering practices [40, 43, 45].

To reduce exposure, it is of utmost importance to create and maintain secure passwords on all Internet capable devices, often several IoT devices are released with a preset administrator account and password that are often left as default and create a risk as they are easy to discover as it was the case with the 2016 Mirai botnet discussed in chapter I. According to recommendations by the United States Department of Homeland Security (US DHS) and the National Institute of Standards and Technology (NIST) a password should not use any common phrases, quote, lyrics, personal information or simple words (avoid dictionary attack), a password should use if possible the longest length available, a strong combination of uppercase, lowercase letters, number and if possible special characters (avoid brute force cracking). Additionally, it is wise to not reuse password on several accounts, for only one is required to become compromised for all to become compromised. In the even a system becomes part of a botnet performing a system wipe/restores will dispose of all malware related to the botnet in the infect system, other systems may require a wipe/restore to ensure the local area network is safe [40, 45, 46].

2.1.2 Benevolent Botnets

Inherently botnets are associated with malicious and criminal activity, however botnets could also be used for benevolent purposes and thus be legitimate. A legitimate botnet commonly known as distributed computing systems aim to spread computing power across several systems to expedite the completion of tasks in an efficient manner, and since all active systems participate voluntarily a reduction in cost is present as well. For instance, at the University of California, Berkeley an ongoing scientific experiment named SETI@home utilizes distributed computing to analyze radio telescope data in the Search for Extraterrestrial Intelligence (SETI) [47]. In the same way, at the Universitat Pompeu Fabra in Barcelona, Spain a volunteer distributed computing project for Biomedical Research named GPUGRID.net utilizes the power of several graphics processing units (GPU) to achieve high-performance all-atom biomolecular simulations. These molecular simulations are among some of the most computational demanding processes and would usually require a supercomputer to be performed. The use of a GPU grid allows for a cost-effective solution to high intensity simulations in several research topics such as cancer, HIV/AIDS, neural disorders, etc. [48]. Both benevolent botnets at their core utilize a software module known as BOINC which was developed by the University of California, Berkeley to permits the harvesting off all essential computational power. Several other projects utilizing this software module for cost-effective advancement in research [49].

2.2 Background study on Computer Networks

Distributed Denial of Service attacks remain a prominent threat to the Internet, commonly DDoS attacks are launched from a botnet targeting a weakness in the Internet protocol suite which is currently and generally referred to as the TCP/IP protocol suite. To

understand more about how these attacks work, it is important to understand the origin and methodology of the TCP/IP protocol suite.

In 1984 the International Organization for Standardization (ISO) published the Open Systems Interconnection (OSI) model which has defined the most elementary concepts of computer networking. The OSI model is a theoretical model which defines and standardizes the communication protocols employed by computing and telecommunication systems. The objective of the design for the OSI model offered a framework for the standardization of network design, thus permitting the communication of systems developed by several distinct manufactures across the world. The design of the OSI model sees it divided by seven extensible hierarchical layers with their own header and function, where each layer encapsulates the one after it. Figure 2.6 highlights all seven layers and their basic function [50-56].

Layer:7 Application	Access to OSI environment for users. High-level APIs, resource sharing, remote file access, directory services, network management.
Layer:6 Presentation	Conversion of data between a networking service and an application. character encoding, data conversion, compression and encryption/decryption
Layer:5 Session	Controls communication sessions between applications. Initiates, manages, and concludes sessions among cooperating applications
Layer:4 Transport	Reliable transparent transmission of data segments between endpoints, including error recovery, message acknowledgment, segmentation and flow control
Layer:3 Network	Structuring and managing a multi-node network. Routing, Subnet traffic control, frame fragmentation, address mapping
Layer:2 Data Link	Reliable transmission of data frames between two connected nodes across the physical link, adding synchronization, error control, and flow control.
Layer:1 Physical	Transmission and reception of unstructured binary stream over a physical medium.

Figure 2.6 OSI Model

The OSI model did not become the absolute standard used by the Internet today due to several technical (complexity), political, social, and economical aspects present during the conception of the Internet. Furthermore, two of the major reasons the OSI model was not adopted as the standard for Internet are attributed to the late development of the model and the

involvement in the early 1980s of the US Defense Advanced Research Project Agency (DARPA) which along side with the Defense Communications Agency which helped expedite the adoption of the TCP/IP protocol suite. This was achieved by subsidizing research into implementing the TCP/IP protocol suite into popular operating systems and discontinuing support for the ARPANET host protocols, resulting in the forced adoption by many contractors of the TCP/IP protocol suite. These actions gave birth to the Internet on January 1, 1983 [54]. Despite not becoming the standard for the Internet the OSI model remains as a strong reference for understanding how networks operate and how systems can communicate with each other.

The TCP/IP protocol suite was funded and developed by the US DARPA to address the need for computer communications between heterogeneous computers on heterogenous networks. The standards and protocols set forth by the TCP/IP protocol suite are maintained by the Internet Engineering Task Force (IETF), this is an open non-profit community of network engineers, vendors, operators, and researchers concerned with continuous development of the Internet architecture and the efficient operation of the Internet [55]. The TCP/IP protocol suite is the theoretical model and a set of communication protocols used by the Internet and homogenous networks [56-58]. The TCP/IP protocol suite offers end-to-end data transmission detailing how data should be packetized, addressed, transmitted, routed, and received as defined in RFCs 1122 and 1123 [59,60]. The design of the TCP/IP protocol suite model is divided in five hierarchical layers (originally four) with their own header and function. If compared to the OSI Model, the TCP/IP protocol suite operates in a similar way, however it features five layers where layers five thru seven from the OSI model are merged into one, and originally layers one and two were merged and known as the network access layer but has been split into the network and physical layers. Figure 2.7 highlights the functions of all five layers [56-58].

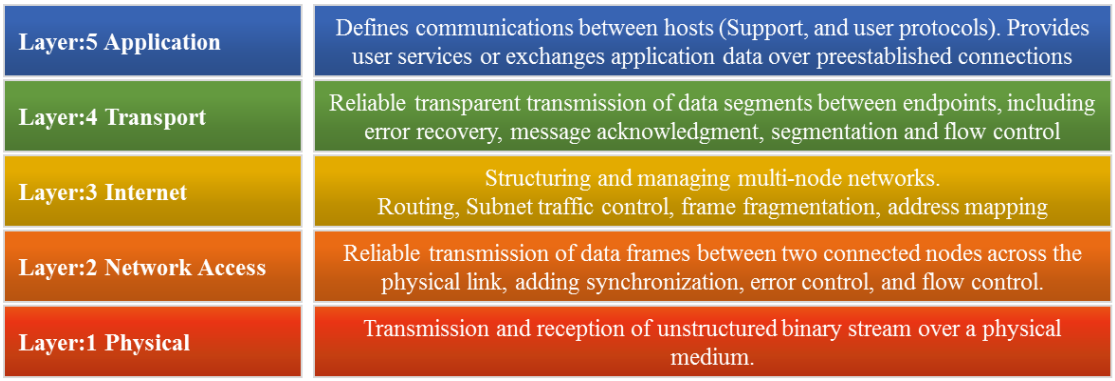


Figure 2.7 TCP/IP protocol Suite Layers

The TCP/IP protocol suite, utilizes encapsulation, for the transmission of data. Encapsulation is the process of constructing protocol data units (PDUs) by generating headers (and trailers occasionally) and adding them to the PDU from the layer above, having at the end the data intended for delivery. The PDUs are comprised by two (sometime three) portions of data including a header, a payload and sometimes a trailer in that respective order. Headers are part of the binary streams used for the transmission of data, within each header several fields are present containing essential instructions for transmission, the trailer if present will contain similar fields. Payloads contain the PDU of the layer above which includes the data intended for transmission [56-60]. The encapsulation method for the TCP/IP protocol suite is exemplified on Figure 2.8

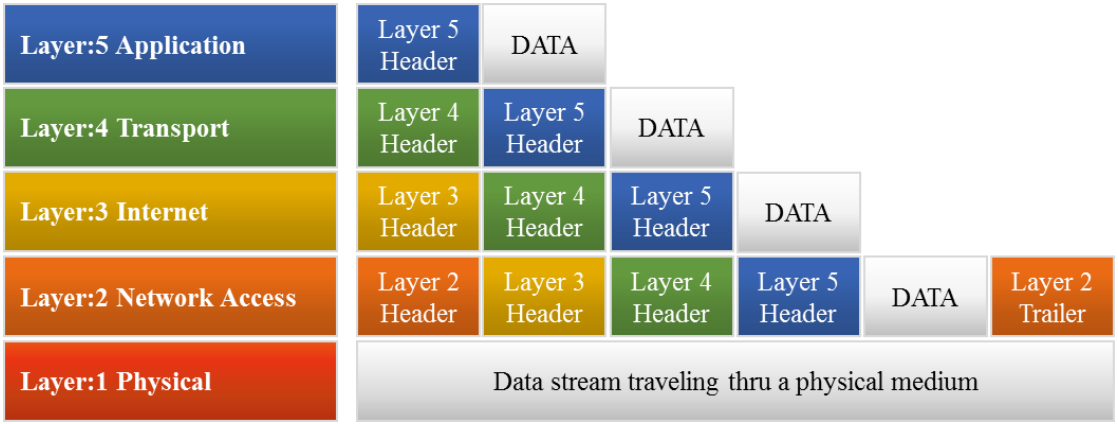


Figure 2.8 TCP/IP protocol Suite encapsulation across abstraction layers

2.3 TCP/IP Suite Layer 3 ICMP Based DDoS Attacks

In computer networking, the Internet Control Message Protocol (ICMP) defined by RFC 792 and updated by RFCs 950, 4884, 6633, and 6918 is a protocol encapsulated within layer 3 of the TCP/IP suite and the OSI layer which is frequently utilized by network administrators to troubleshoot and administrate a network. ICMP as the name implies offers the ability to communicate query messages, and report errors thru compact control messages between end hosts and routers. Query messages provide specific information from a router or host in the network, while error messages report issues within the network. These messages can be used to survey a network and are exceptionally helpful for diagnosticating and troubleshooting a network problem [57, 59, 61-65]. A comprehensive list of ICMP parameters can is found at [66].

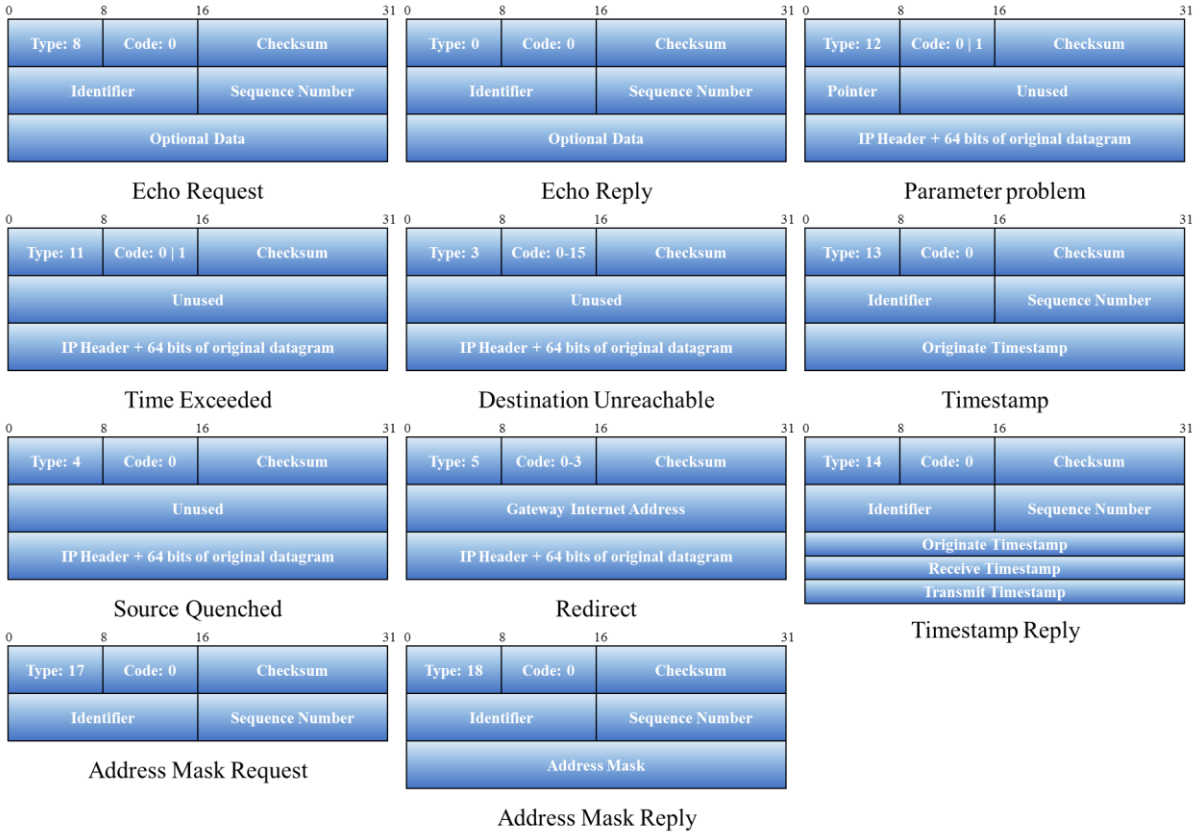


Figure 2.9 Commonly used Internet Control Message Protocol (ICMP) messages

As demonstrated by Figure 2.9 the ICMP message format can differ depending on the type of message sent but generally consists of a 64-bit header and a data section of adjustable size. Regardless of the message type sent out the first 32 bits will always hold the same format, where the first 8 bits pertain to the TYPE of ICMP message that will be sent out, the second belong to the CODE which detail the nature of the message, and the last 16 bits are used for a CHECKSUM which is used in an algorithm used for error detection. However, it is worth nothing that there is no guarantee the ICMP message will be delivered [57,61].

Kaspersky Lab generates DDoS reports for every quarter of the year; according to their data from 2017 and up to the third quarter of 2018 an average 4.82% of all recorded DDoS attacks have been ICMP based [21-23,44,67-69]. The low percentage could be attributed to the ease at which they can still be mitigated. To mitigate ICMP attacks network settings can be set to disable all ICMP messages or limiting the rate at which ICMP messages can be accepted. As a result, the messages utilized by Ping flooding and Smurf attacks will not be processed or transmitted, however with ICMP messages disabled the Ping utility (used in attack), traceroute requests, and other network activities involving ICMP messages will also become unresponsive [70].

ICMP messages provide network administrators with the Ping utility, which is a useful tool for understanding and troubleshooting a computer network via echo request and echo reply messages. The functionality of the Ping utility is demonstrated by Figure 2.10 where a local address is pinged with four echo request messages and returns four echo replies; from the ICMP message and the IP datagram the time required to reach the target system, the round trip time, and the time to live of the packet is displayed to the operator who issued the ping. [57,61].

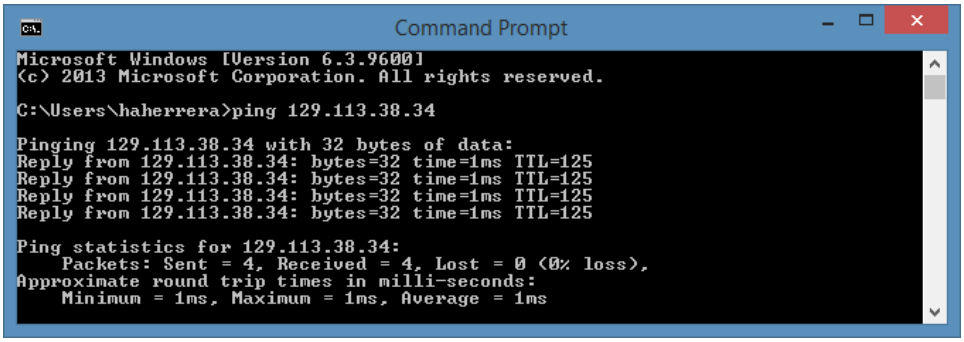


Figure 2.10 Ping Utility under Microsoft Windows

While ICMP messages provide network administrators with a valuable tool, they can also be exploited to trigger devastating Denial of Service attacks. In the following two sections this thesis addresses two popular ICMP based DDoS attacks, namely Ping Flooding, and Smurf attack.

2.3.1 Ping Flood Attack

Ping Flooding is a layer 3 attack which exploits vulnerabilities in the Ping utility by transmitting an extreme amount of ICMP echo request messages to overwhelm and exhaust a system's resources. To launch a Ping flood DDoS attack, the attacker can enlist a botnet to flood the targeted system with echo request messages where the bots may use a false IP address to avoid detection. This technique of using false IP address is known as spoofing [70]. As stated by RFC 792 when a system receives an ICMP echo request message it is obligated to respond with an ICMP echo reply message as seen on Figure 2.11 [61].

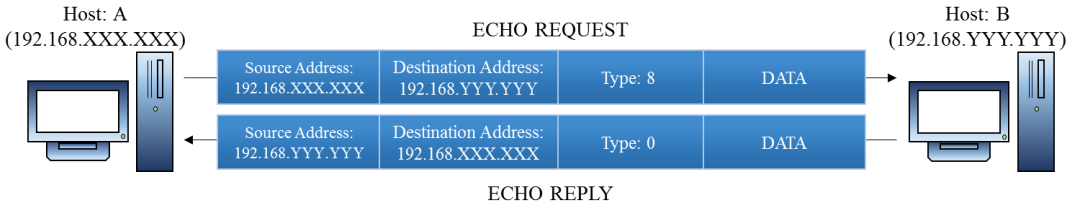


Figure 2.11 Ping Utility

Networks not blocking ICMP messages can be victims of Ping flooding, since attackers can exploit the Ping utility, by generating an excessive amount ICMP echo request messages and transmitting them to one of the systems over the Internet. While the bandwidth of the affected system is diminished it will attempt to answer all ICMP echo request messages with an echo reply message which will consume the processing power of the affected system. Figure 2.12 demonstrates the flow of ICMP echo request and echo replies under a Ping flooding attack [70].

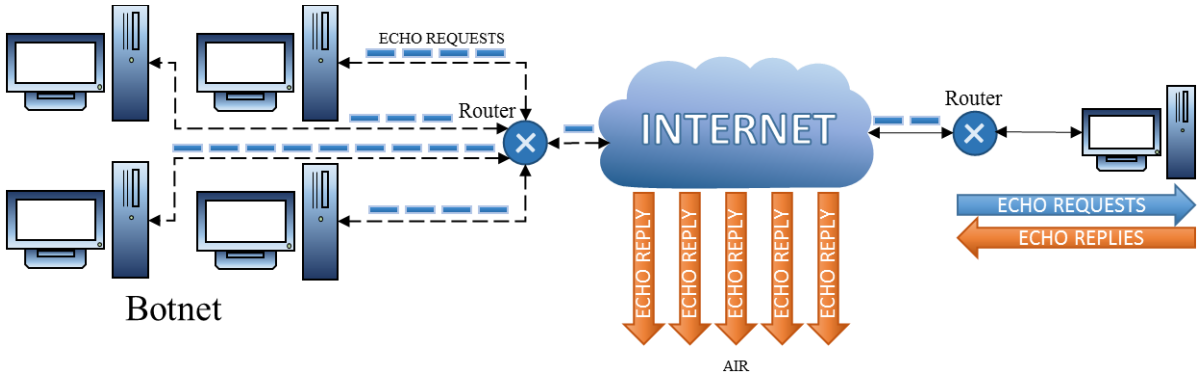


Figure 2.12 Layout of Ping Flooding attack

2.3.2 Smurf Attack

Smurf attacks are layer 3 attacks exploiting vulnerabilities of the Ping utility and the Broadcast address on a network with the intent of flooding the target with ICMP echo request messages resulting in the consumption of network bandwidth, and processing power of the targeted system [71-73]. Broadcast address is defined by RFC 919 and updated by 922. A broadcast address relays data received to every host on the local network it is useful for providing data to a large set of hosts in a timely manner, or when a host needs to find information on the network without knowing specifically what other hosts can supply [74-75]. To perpetrate a Smurf attack, the attacker will use malware to forge packets with the spoofed IP address of the victim and transmit echo requests into the broadcast address of that network, or

alternatively the broadcast address of a botnet, as a result every host that received the request in the broadcasting network will respond to the spoofed IP address with an ICMP Echo reply message as demonstrated by Figure 2.13. The intensity of the attack is determined by the number of hosts responding to the fake ICMP echo request message. As hosts continue to flood the targeted system with echo reply messages, the system can become overwhelmed and lost connectivity from legitimate sources in a rapid manner [71-73]. To mitigate the attack ICMP messages and the IP broadcasting addresses could be deactivated at each network router and firewall [73].

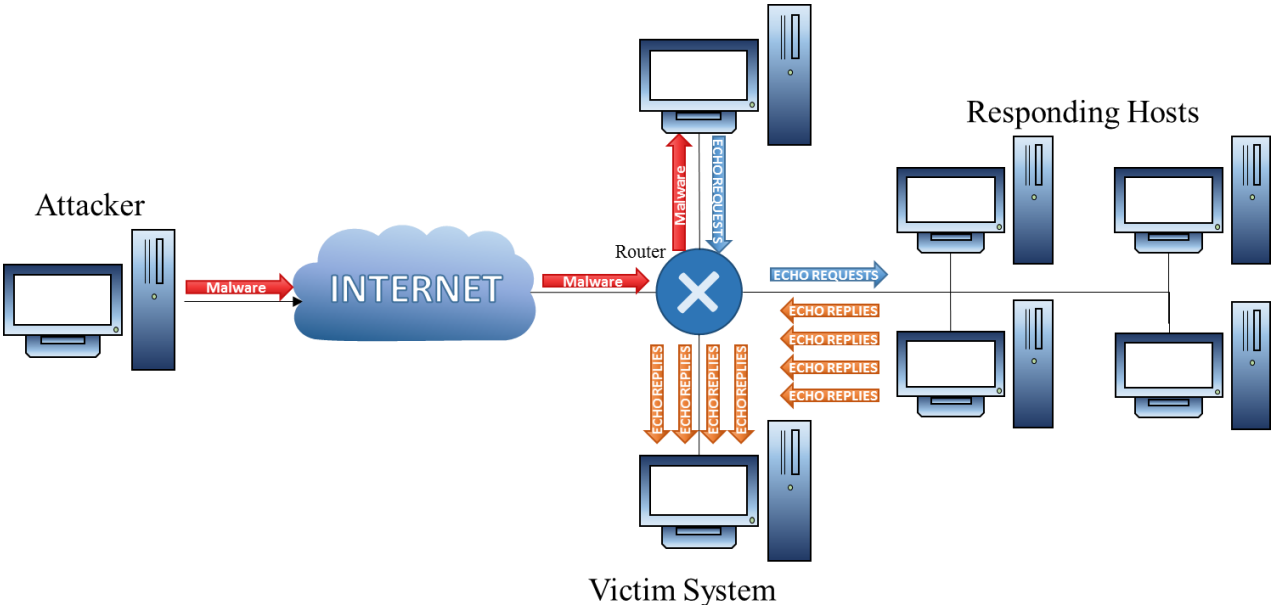


Figure 2.13 Layout of typical a Smurf attack

2.4 TCP/IP Suite Layer 4 Based DDoS Attacks

The Transport layer is comprised by a set of protocols (TCP/UDP) in the fourth layer of both the TCP/IP suite and the OSI model where its encapsulated by the third layer (network/ Internet). The transport layer offers end-to-end data transmission. The transmission layer offers two types of transport service: connectionless and connection-oriented. Connectionless services

utilized the UDP (User Datagram Protocol) and connection-oriented services are fulfilled by the TCP (Transmission Control Protocol). These protocols depending on their service may feature error-checking, segmentation, congestion control, flow control and application addressing (port numbers). The transfer protocols are commonly used by applications available on the Internet such as the World Wide Web (WWW), HTTP, e-mail (SMTP), time synchronization (SNTP), File Transfer Protocol (FTP), Trivial File Transfer Protocol (TFTP), Secure Shell (SSH), peer-to-peer file sharing (P2P), Domain Name System (DNS), TELNET among many more. Core differences between UDP and TCP can be observed on Table 2.1 [57].

TCP vs UDP		
Characteristic	**TCP**	**UDP**
Protocol Connection Nature	Connection oriented transmission	Connection less transmission
Protocol Data Unit	Segment	Datagram
Reliability	Reliable, data sent is acknowledged	Unreliable, delivery by best effort without acknowledgment
Retransmission	Corrupted or lost data is retransmitted	N/A Applications must detect data loss and request retransmission
Flow control	Data flow based on router/network performance	N/A
Transmission Speed	Slower than UDP	Faster than TCP
Header size	Dynamic (20-60 Bytes)	Low (Static 8 bytes)
Checksum function	Obligatory	Optional with IPv4 not optional with IPv6
Error Reporting	Yes	No
Support broadcast	No	Yes
Support multicast	No	Yes
Packet Ordering	Sequential based	No order

Table 2.1 UDP vs TCP

2.4.1 TCP SYN Flood Attack

The Transmission Control Protocol defined by RFC 793 and updated by RFCs, 1122, 3168, 6093, and 6528 is a connection oriented, end-to-end protocol intended to provide reliable data transmission [76]. TCP features error-checking/reporting, segmentation, congestion control, flow control and application addressing (port numbers). Error-checking verifies the integrity of the data transmitted, segmentation is achieved by splitting a larger data streams into smaller segments which is necessary in order to fill the TCP data field over several segments which helps eliminated duplicated packets and offers retransmission in case of packet loss, or corruption, while congestion and flow control prevent in wired connections the loss of packets (loss of packets can still be experienced in wireless connections due to physical interference) by regulating the transmission rate based on the performance of routers on the network path (bandwidth and data buffer). TCP segments are exemplified on figure 2.14 [59, 76-79].

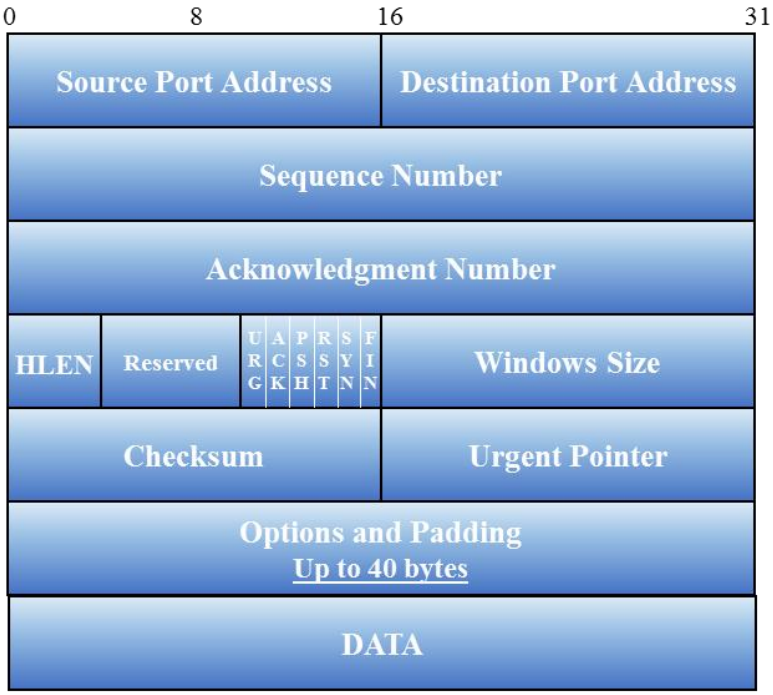


Figure 2.14 TCP Segment

The end-to-end reliability of data transferred under TCP is greatly preserved by the generation of a connection between the sender and the receiver. Per TCP data will only be transferred once the connection has been established. The connection used by TCP is commonly known as the three-way handshake. To initiate a new TCP connection in step one, a host(sender) must transmit a TCP segment containing an active SYN (Synchronization) flag and an initial sequence number chosen at random and the acknowledgment number field empty. On step two upon arrival the receiver will transmit to the sender a new TCP segment containing active SYN and ACK (Acknowledgment) flags along with a new randomly chosen sequence number, and a new acknowledgment number resulting from the single increment to the previous sequence number. Additionally, the receiver upon obtaining the initial segment will allocate buffers for each TCP connection. In the final stage once the initial sender receives the second segment with the SYN and ACK active flags from the receiver it will transmit another segment with an active ACK flag and allocates a buffer for the connection that was just established. The three-way handshake is demonstrated by Figure 2.15 [59,76].

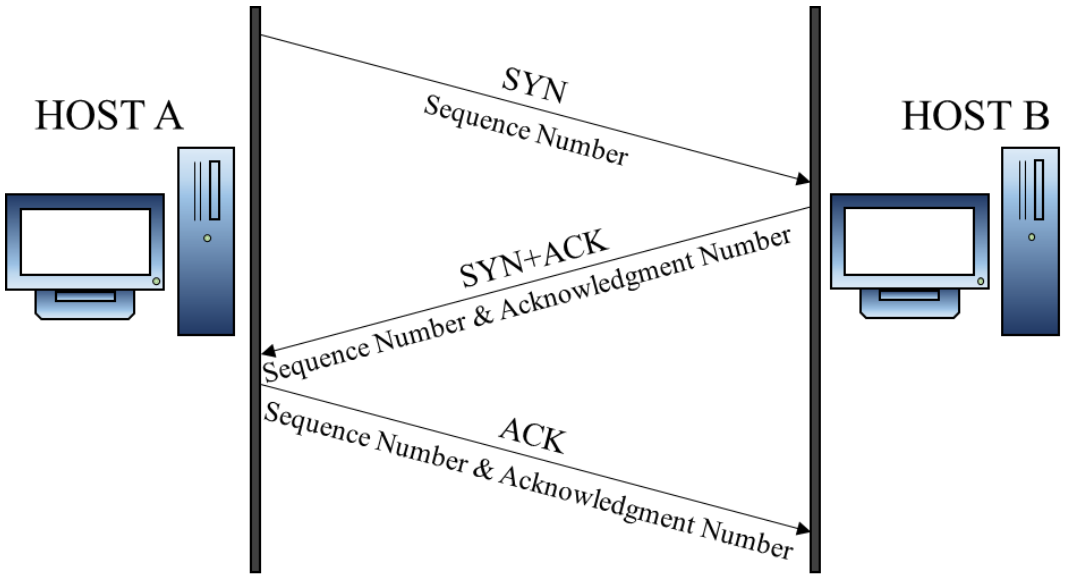


Figure 2.15 Legitimate three-way handshake

Records of TCP connections are kept by a Transmission Control Block (TCB) in a per connection basis. TCB is a data structure holding relevant information of each TCP connection such as connection state, feedback parameters regarding the connection's transmission properties, and its associated process, this includes port numbers and addresses and of the connection endpoints, data transmitted by either end-point (done for retransmission), the latest expected round-trip time, and several TCP statistics regarding the connection [76].

Kaspersky Lab generates DDoS reports for every quarter of the year; according to their data on average 65.58% of all recorded DDoS attacks since the first quarter of 2017 and up to the third quarter of 2018 have been TCP SYN Flood attacks [21-23,44,67-69]. Kaspersky reported in the second and third quarters of 2018 an increase in TCP SYN Flooding attacks has been experienced [22,23].

SYN Floods are carried out by an attacker who may use a botnet or a set of spoofed IP addresses to exploit the three-way handshake in TCP resulting in a devastating DDoS attacked. SYN Flooding is achieved by not completing the final step in the three-way handshake of a TCP connection. To initialize a SYN Flood that attacker will transmit connection requests (TCP SYN segments), to the victim at a high volume. The victim per TCP will create half-active connections in the TCB as it attempts to respond to all incoming SYN requests, while expecting a final ACK TCP segment that will never arrive. If the final ACK segment never arrives half-active connections will be dropped after a preconfigured amount of time however, overtime half-active connections will have already begun to saturate the TCB (dependent on attack traffic rate) while resources such as processing power and bandwidth are consumed rendering the victim unable to process legitimate traffic [80-83]. Traffic flow under TCP SYN Flooding is demonstrated by Figure 2.16.

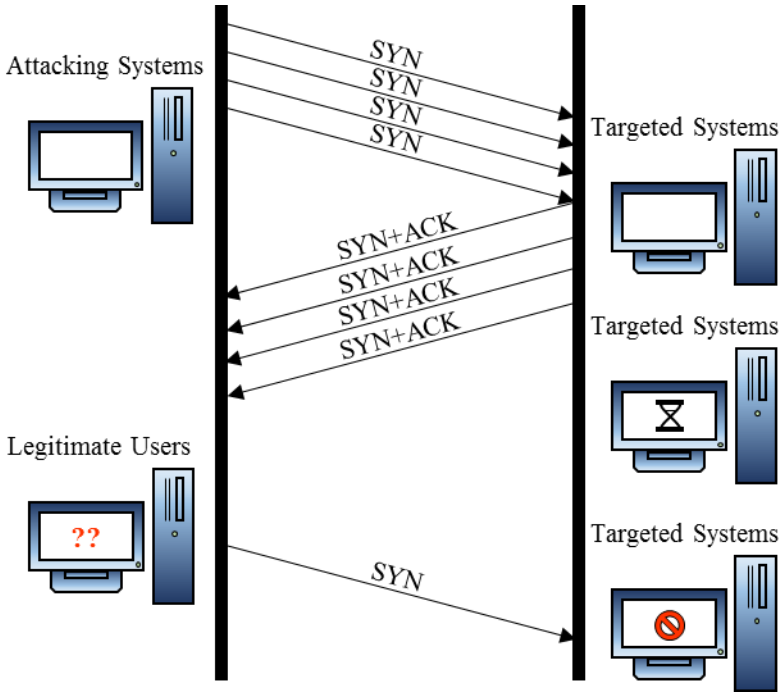


Figure 2.16 TCP SYN Flooding

TCP SYN Flooding have been an active treat of the Internet for a long time. Therefore, several tactics have been proposed and implemented to mitigate the effects of TCP SYN Flooding attacks. RFC 4987 describes common defense techniques which include packet filtering based on IP addresses, increasing backlog, reducing SYN-received timer, recycling the oldest half-open TCB, SYN cache, SYN cookies, hybrids approaches from the former techniques, firewalls, and proxies [80].

2.4.2 UDP Flood Attack

The User Datagram Protocol defined by RFC 768 with best practices defined by RFC 8085 is a connectionless communication layer 4 protocol, intended to provide simple yet unreliable data transmission for transaction-oriented services while lacking delivery and duplication protection [84,85]. UDP's unreliability originates from the lack of a connection-

based model which address packet ordering, delivery, and duplication protection. As such UDP performs in a best effort basis, and has a lower overhead than TCP, offering lower latency which may be preferred in some applications where loss of data can be tolerated or where latency might be a concern (Real time applications such as: VoIP, TFTP, online gaming as examples). UDP also provides checksum for the verification of data integrity (optional for IPv4), but it is up to the application to request a retransmission if necessary. UDP datagrams are exemplified on figure 2.17 [84].

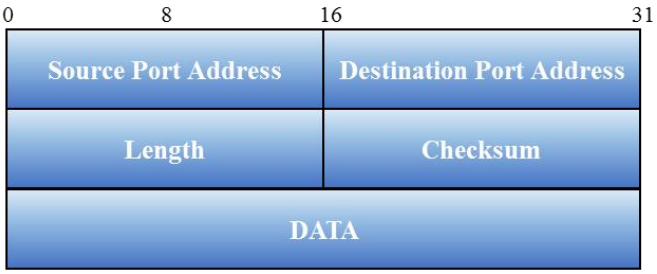


Figure 2.17 UDP Datagram

When a system receives a UDP datagram it will inspect if any active application is currently listening for requests at the specified destination port address. If port number specified by the received datagram is currently being listened to by an application, the datagram is forwarded and the application will address the data in the UDP datagram however, if the port is closed or if no applications are listening to the destination port, the system responds with an ICMP packet stating the destination was unreachable [59,84] as seen on Figure 2.18.

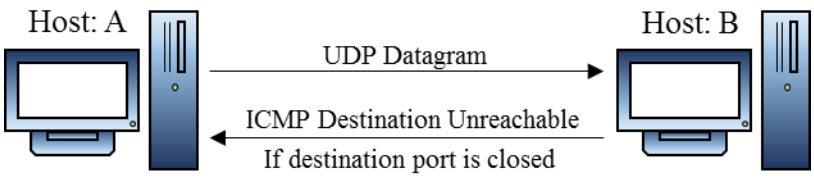


Figure 2.18 UDP Response on closed ports

Attackers can exploit this utility by flooding the victim with a high volume of spoofed UDP datagrams while targeting random ports. Consequently, the system will utilize its resources to check for applications listening to the ports, and upon finding no application listening will be forced to reply with an ICMP Destination Unreachable packet to every spoofed source thus consuming bandwidth and processing power, consequently preventing the system from being reached by other clients [84,86,87] as demonstrated by Figure 2.19.

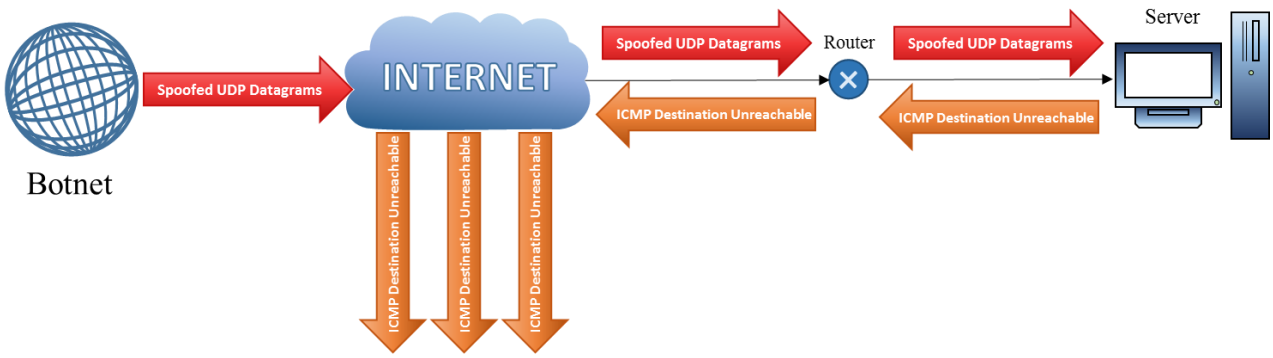


Figure 2.19 UDP Flooding

To mitigate UDP Flood attacks, network administrators can utilize proxy mechanisms, and disable ICMP responses, thus lowering the amount of processing power consumed, additionally disabling and filtering unused UDP services will help mitigate the attack [86-87].

Kaspersky Lab generates DDoS reports for every quarter of the year; according to their data on average 11.68% of all recorded DDoS attacks since the first quarter of 2017 and up to the third quarter of 2018 have been UDP Flooding attacks [21-23,44,67-69].

2.5 Chapter Summary

This chapter discussed and reviewed the concepts behind Botnets, and Distributed Denial of Service Attacks. An overview of common cyber-attacks ICMP Ping Flood, Smurf, TCP SYN

Flooding and UDP Flooding was presented along with a brief description of the protocols exploited by them and techniques available for their mitigation. These four attacks were performed in a controlled environment with simulated botnet sizes for the experiments in this thesis.

The cyber-attacks defined by this chapter will help in the evaluation and comparison of the built-in security features of several popular operating systems used for webservers.

CHAPTER III

COMPARATIVE EVALUATION OF WINDOWS SERVER 2012 R2 AND WINDOWS 2016 DATACENTER UNDER THE EFFECT OF DISTRIBUTED DENIAL OF SERVICE ATTACKS FROM BOTNETS

In this chapter the security features of Microsoft's Windows Server 2012 R2 and Windows Server 2016 are assessed thru a comparative evaluation where the performance of each operating system is compromised thru DDoS attacks executed by simulated botnets replicating compromised Class A, B, and C IP networks. Similar works have been performed by participants at the Network Research Lab (NRL) of the University of Texas Rio Grande Valley (Before the school merger the lab and works were listed under the University of Texas - Pan American) [24-36] nevertheless, this study will introduce newer operating systems, and the effects of larger simulated botnet sizes.

Windows 2012 R2 is the sixth release of Microsoft's Windows Server family, it preceded Windows Server 2012 and is succeeded by Windows 2016 which became available on October 15, 2016. The lifecycle of Windows 2012 R2 started on November 25, 2013; mainstream support concluded on October 9, 2018 and extended support expires on October 10, 2023 [88]. Comparatively mainstream support for Windows 2016 will end on January 11, 2022 while extended support expires on January 11, 2027 [89].

This study is performed to ascertain if any modifications or improvements have been achieved by Microsoft in terms of performance and survivability against common DDoS attacks.

The Datacenter version with GUI of both operating systems were implemented on Apple's Mac Pro Server (Mac Pro Mid2010). Applying identical hardware characteristics for both server versions will provide clarity in these experiments as difference in hardware will not have a role on system performance. Furthermore, all operating systems used in this study feature all security updates released up to December 2018.

Section 3.1 of this chapter introduces the experimental setup, hardware and software specifications used under the experimental trials for these attacks, and the methodology thru which they are carried out. Parameters of performance evaluation are defined in section 3.2 within this chapter. These parameters will be used to compare the performance provided by each operating system. Section 3.3 will demonstrate and discuss the data obtain for each parameter of evaluation described. This will permit the understanding of the effects created by these attacks.

3.1 Experimental Setup

Operating system evaluations of Microsoft's Windows 2012 R2 and Windows 2016 Datacenter versions were conducted within a controlled environment established at the Network Research Lab of the University of Texas Rio Grande Valley. The security features and performance of both operating systems are evaluated against common IP layer ICMP, TCP, and UDP based attack traffic originating from simulated botnets emulating compromised Class A, B, and C IP networks. Clean installations of both operating systems were deployed on Apple's Mac Pro Server (Mac Pro Mid2010) to properly evaluate the security features both server operating systems. Additionally, operating system updates were installed up to releases from December 2018, and firewall settings were left on by default while high performance features were enabled.

For these experiments attack traffic was transmitted as simulated legitimate traffic attempted to reach the web server within the platform. Microsoft's Internet Information Services (IIS) is an extensible web server and is utilized by both operating systems to launch and manage a simple Hypertext Markup Language (HTML) website named Index.html and is seen on Figure 3.1 [90].

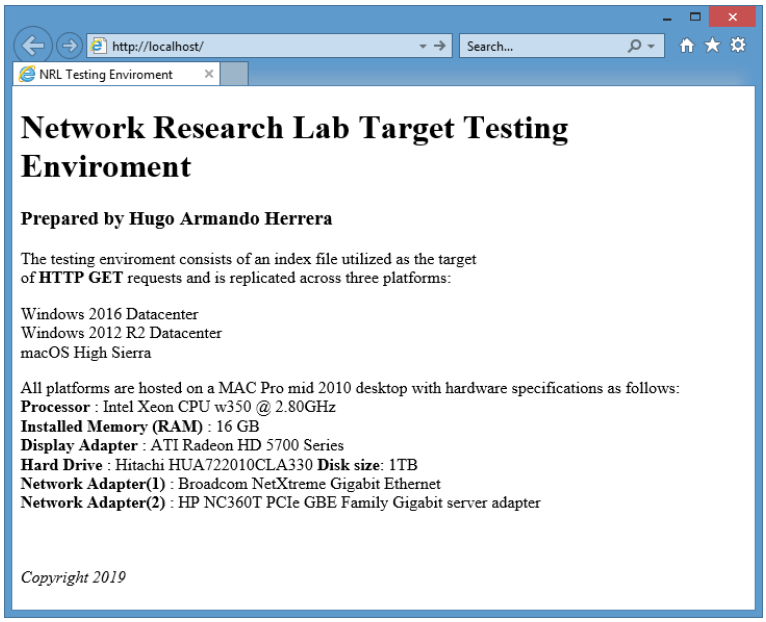
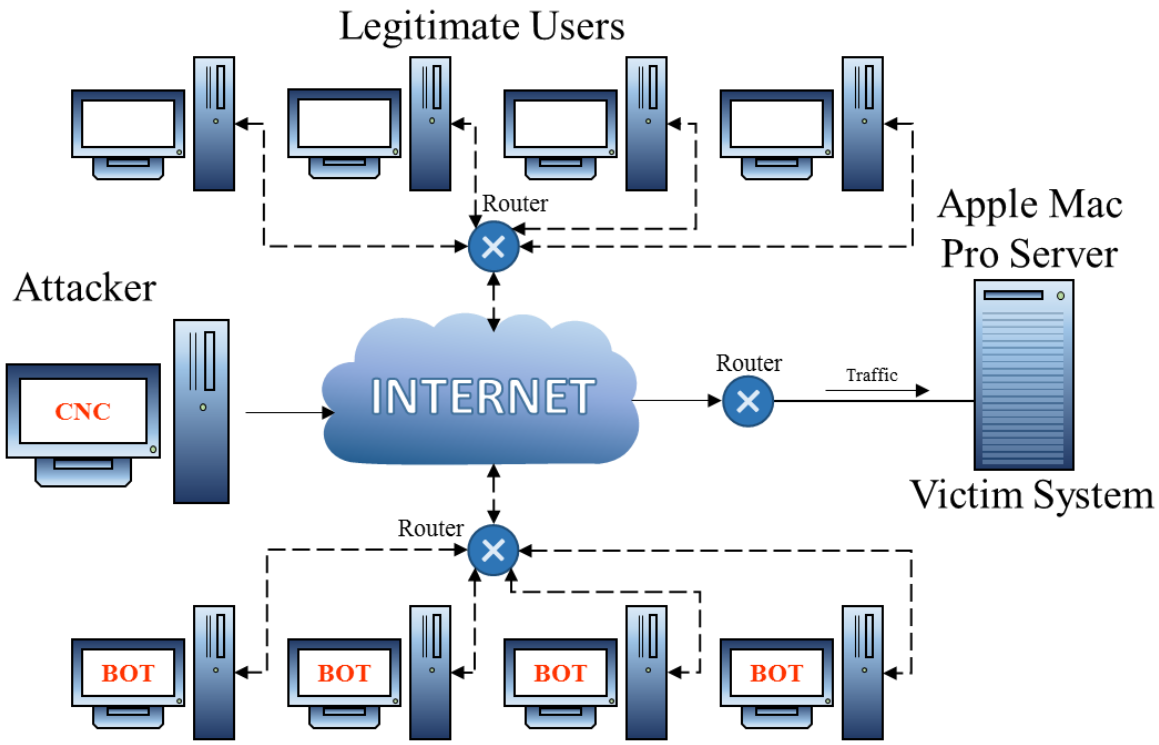


Figure 3.1 Targeted Website on Windows 2012 R2 and 2016 Datacenter Versions

The availability of this website was determined by the amount of consistent Hypertext Transfer Protocol (HTTP) Transaction per second achieved by the legitimate clients. HTTP defined by RFC 7230 is a stateless request/response application layer protocol that operates by exchanging request and response messages thru a reliable connection [91]. For these experiments HTTP GET messages will be used to request the Index.html website file from the webserver at a rate of 3000 HTTP Transaction per second for 55 minutes. Under this condition no HTTP requests were denied.

Attack traffic is transmitted at different line rates (attack loads), up to a maximum of 1 Gbps. During an experiment attack traffic is initially transmitted at a line rate of 100Mbps for five minutes, subsequent loads increase the line rate of the previous by 100Mbps up to 1Gbps while the parameters of performance are recorded. Legitimate traffic was transmitted five minutes prior to the attack traffic, this was done to provide data for both idle(regular) performance and under-attack performance.

The experimental setup for the comparative evaluation was designed to simulate the network described in Figure 3.2 where the Internet and routers can be visualized as a single router and a switch.



	Host IP Range	Mask	# of Hosts
Class C	192.168.5.1 - 192.168.5.254	255.255.255.0	254
Class B	172.25.0.1 - 172.25.255.254	255.255.0.0	65534
Class A	10.0.0.1 - 10.255.255.255	255.0.0.0	16777214

Figure 3.2 Experimental Setup

3.1.1 Hardware

This subsection will provide hardware specifications of the network and system under attack.

Server Platform

The targeted server system used was Apple's Mac Pro Server (Mac Pro Mid2010) equipped with one 2.8GHz Quad-Core Intel Xeon W3530 "Nehalem" processor (featuring Hyper-Threading technology for up to 8 virtual cores), 16 GB of RAM,[92] and a Broadcom NetXtreme Gigabit Ethernet adapter (BCM5719) [93]. The built-in network interface card (NIC) provided by Apple did not feature any compatible drivers for Windows 2016, or 2012 platforms, therefore the Broadcom's BCM5719 network interface card was chosen as a replacement. Documentation for the NIC can be found at [93]. The Mac Pro Server has the capabilities of running both Microsoft's Windows 2012 R2, and Windows 2016 Datacenter versions.

Server specifications are as follows:

Operating Systems: Microsoft's Windows 2012 R2 Datacenter and Microsoft's Windows 2016 Datacenter

CPU: 2.8GHz Quad-Core Intel Xeon W3530 "Nehalem" processor [94]

Number of Processors: 1

Number of Cores: 4 physical cores. 8 virtual cores thru Hyper-Threading

Random Access Memory (RAM): 16 Giga Bytes

Network-Interface-Card (NIC): BCM5719 - QUAD-PORT 1GBASE-T

High-Performance, Feature-Rich NetXtreme® BCM5719 Quad-Port 1GBASE-T PCIe 2.1 Ethernet Controller

Graphics Processing Unit: ATI Radeon HD 5770 with 1GB of GDDR5 memory, PCI Express 2.0

Switch

For these experiments Cisco's SRW2024 24-port gigabit switch was utilized. This switch features 24 high-speed ports optimized for bandwidth-intensive applications, with data transfer rates of 10/100/1000Mbps and is compliant with IEEE standards 802.3, 802.3u, 802.3ab, 802.3x, 802.1p, 802.1q. No additional devices were connected at the time of experiments were performed. Documentation for this switch can be found at [95].

3.1.2 Software

To collect and process data regarding the performance parameters several tools were utilized and are documented as follows:

Microsoft Performance Monitor is built-in on both Windows versions. This monitoring tool measured the consumption of system resources while recording relevant network traffic data. This was achieved with the creation of a user defined data collector set [96]. Moreover, CoreTemp was used to record CPU temperature and observe CPU frequency [97]. Both tools recorded at a sampling rate of one sample per second and stored all data on comma separated value (.csv) files. Execution of both tools has a low impact on server performance as CPU utilization is below 1% and random access memory (RAM) consumption is 13 megabytes (MB)

(10.4 MB from the performance monitoring tool, and 2.6MB from CoreTemp) out of 16 gigabytes (GB).

To process the data recorded a set of short scripts were programmed in MATLAB [98]. The scripts extract relevant data from .csv files and output averages for each traffic load. Microsoft Excel was chosen as the plotting tool for the graphs in the results section [99].

3.2 Parameters of Performance Evaluation

In this experiment, the parameters that are being used in the comparative evaluation of performance were processor utilization, processor temperature, random access memory (RAM) consumption, HTTP transactions per second, number of Echo request and replies received per second, number of Echo replies sent per second, number of ICMP Destination Unreachable messages sent, and UDP datagrams received per second. Some parameters regarding network traffic may be omitted depending on their relevance to the attack implemented. Performance parameters are recorded as the targeted system experiences both legitimate and attack traffic. Performance parameters are more clearly defined in the following pages:

Processor Utilization (Usage of CPU in %) – The central processing unit (CPU) is one of the most crucial hardware components in a computing system as it interprets and executes a wide majority of the commands/instructions relayed by other hardware and software components. For a CPU processing generally involves fetching data, decoding, and executing instructions. Software applications are passive collections of instructions, once the software is executed a process is created containing the program code and its activity. Process are comprised by threads of execution. Threads are short sequences of instructions managed autonomously by the operating system thru a scheduler. Instructions typically involve basic arithmetic, relocating data

on the memory or data comparison. Originally a processor had a single core and only a single thread could be executed at a time. Multithreading was made possible by switching between multiple software threads at a high rate creating the illusion of parallel execution. Modern processors feature multiple cores which has allowed for multiple threads to be executed in parallel thus expediting the run time of several applications. The server used in these experiments has a 2.8GHz Quad-Core Intel Xeon W3530 "Nehalem" processor [94]. This processor features Hyperthreading which allows the operating system to recognize the four physical cores as eight virtual or logical cores thus allowing multicore processors twice as many threads to be processed in parallel which increases the amount of independent instructions in the pipeline. [100] It is worth nothing that not all applications take advantage of hyperthreading, typically applications with high resource consumption such as digital media applications, and gaming titles which make use of 3D rendering, asset streaming, terrain generation, physics effects, video encoding, virus scanning etc. are designed to use hyperthreading [101]. To measure the performance of a processor the Processor: %Processor Time from Microsoft's performance monitor is taken into consideration. Processor: % Processor time as it appears on the performance monitor is the percentage of time transpired by a processor before it executes a non-Idle thread. Clock cycles in a processor are always being utilized, if a non-Idle thread is not currently being executed an Idle thread is. The calculation for %Processor time is performed by measuring the percentage of time consumed by the processor executing the idle thread and subtracting the result from 100%. Calculations are performed at an internal sampling rate of 10ms. This performance counter demonstrates in percentages the activity of the processor [102]. Certain attacks can be extremely detrimental to processor performance and be classified as processor intensive. A computing system may slow down and become unstable if the processor

saturation is achieved by attempting to process all incoming attack traffic. Collected data for this parameter will permit the assertion of the impact botnet sizes and types of attacks can have on the system.

Processor Temperature (CPU temperature in °C) – Processor core temperatures are taken into consideration since CPU's core temperatures are directly proportional to processor consumption (Computationally intensive tasks draw more energy and more energy yields more heat). If a processor temperature breaches it's physical limits, the system may become unresponsive and crash. Additionally, physical damage from the heat may be present, prolonged exposure can diminish a processors effectiveness permanently. Generally, CPU core temperatures are measured at transistor junctions (Tjunction) within each core. This is the hottest area of each core. Intel defines Tjunction as a synonym for core temperatures, they are calculated by subtracting the output from the Digital Thermal Sensor (DTS) from an established Temperature Junction Max (TjMax) also known as the throttle temperature. To prevent thermal damage to the processor core speeds and voltage are reduced (throttle) when a processor's core reaches TjMax. The processor used in these experiments as well as most processors throttle at 100 °C [103]. Alternatively, a measurement for the entire CPU can be obtained from temperature sensor diode at the die. Access to temperatures values can be dependent on both the operating system and the processor itself.

Memory Consumption (RAM consumption in MB) – Random access memory (RAM) is a form of computer data storage which retains short-term machine code and data based on process currently running on a computing system. RAM determines the amount of processes a system can perform. If RAM became fully exhausted, the operating system will resort to use hard drive space as virtual memory. Virtual memory has slower read/write speed compared to RAM, as a

result when RAM is exhausted the performance of the system degrades and system stability is lost. The increase of virtual memory used will also be detrimental to system performance.

Echo Request Messages Received per Second (Received Echo Request/Sec) – Echo request messages are an integral part of the ping utility. Echo requests messages received per second will be recorded by the victim system to ascertain if any limits have been imposed by the firewall or built-in security mechanism. As the amount of these requests increases, the victim server will have to process more messages which can have a negative impact on the victim server. This parameter will be observed for Ping attacks.

Echo Reply Messages Received per Second (Received Echo Reply/Sec) – Echo reply messages are an integral part of the ping utility and are only transmitted after an echo request has been received. Echo reply messages received per second will be recorded by the victim system to determine if any security mechanism mitigate Smurf attacks. As the amount of these requests increases, the victim server will have to process more messages which can have a negative impact on the victim server. This parameter will be observed for Smurf attacks.

Echo Reply Messages Sent per Second (Sent Echo Reply/Sec) – Echo reply messages are an integral part of the ping utility and are only transmitted after an echo request has been received. Echo reply messages sent per second will be recorded by the victim system. Per RFC 792 when a system receives an ICMP echo request message it is required to respond with an ICMP echo reply message [61]. Under ping flooding conditions the victim server must process incoming ICMP echo request packets while attempting to reply to as many as possible (Firewall settings can prevent this). This parameter will be observed for Ping attacks.

ICMP Destination Unreachable Messages Sent per Second (Sent Destination Unreachable/Sec) – ICMP destination unreachable messages are sent when a frame is discarded due to not being able to reach its destination. Per RFC 1122, a host should transmit an ICMP destination unreachable message with code 3 (Port Unreachable) when the designated transport protocol is unable to demultiplex the datagram but lacks a method to notify the sender [59]. This type of messages may be transmitted by the victim server at an alarming rate under UDP Flooding attacks when UDP datagrams reach ports not being listened to by the system. This parameter will be observed for UDP Flooding attacks.

UDP Datagrams Received per Second (Received UDP Datagrams/Sec) – UDP Datagrams received per second will be recorded by the victim system to determine if any limits have been imposed by the firewall or other built-in security mechanism. This parameter will be observed for UDP Flooding attacks.

HTTP Transactions per second (HTTP Transaction/sec) – Successful HTTP transactions per second will be recorded by an external client transmitting HTTP requests (HTTP GET) to the victim server. Successful HTTP transactions involve the transmission of a request thru the HTTP GET command and receiving a response. The number of successful transactions compared with the baseline will indicate what sort of impact DDoS attacks can have on the server's network connectivity.

3.3 Results and Discussion

Experimental results will be reviewed in this section. In these experiments the performance and security strategies of two commonly deployed operating systems are evaluated under legitimate and attack traffic i.e. Microsoft's Windows 2012 R2 Datacenter and Windows

Server 2016 Datacenter. Both operating systems were deployed on the same hardware platform but were not active at the same time.

The parameters of evaluation discussed in section 3.2. will be utilized in the following subsections where the endurance of both operating system against ICMP Ping flooding, Smurf attacks, TCP SYN flooding, and UDP flooding is compared. To simulate botnet conditions the IP address of the attack traffic will be spoofed to replicate fully compromised Class A, B, and C networks. Networks in these classifications can hold up to 16777214 hosts, 65534 hosts, and 254 hosts respectively. Spoofed IP address were chosen at random, this is done to replicate a practical scenario where a bot master may choose the same action or may send attack traffic from different sources at random depending on each bot's bandwidth availability. As legitimate traffic attempts to reach the webserver, parameters of evaluation will be recorded as each attack traffic is transmitted using a line rate in the range of 0 to 1 Gbps with a step size of 100 Mbps. Attack traffic rate is demonstrated in Table 3.1. Baseline performance will be accounted for as legitimate traffics reaches the webserver and attack traffic is not present. Baseline performance will dictate normal operating conditions which will be compared with each attack bandwidth implemented. Graphs and tables will provide a visual representation of data acquired.

Attack traffic		
Line Rate (Mbps)	Packets/sec (pps)	Bit Rate(Mbps)
0	0	0
100	148,809.52	76.11
200	297,619.05	152.38
300	446,428.57	258.37
400	595,238.10	304.76
500	744,047.62	380.95
600	892,857.14	457.14
700	1,041,667.00	533.33
800	1,201,923.10	615.39
900	1,358,695.70	695.65
1000	1,488,095.20	761.90

Table 3.1 Attack Traffic

3.3.1 Layer 3 ICMP Ping Attack

During the ICMP Ping flooding attacks it was observed that the botnet size did not have an impact on the average of Echo Requests received by both Windows Server operating systems. As demonstrated in Figure 3.3 regardless of the simulated botnet network employed the average amount of ICMP Echo Request messages received per second was kept within the same range for both server operating systems where only the line rate of attack traffic would impact this number.

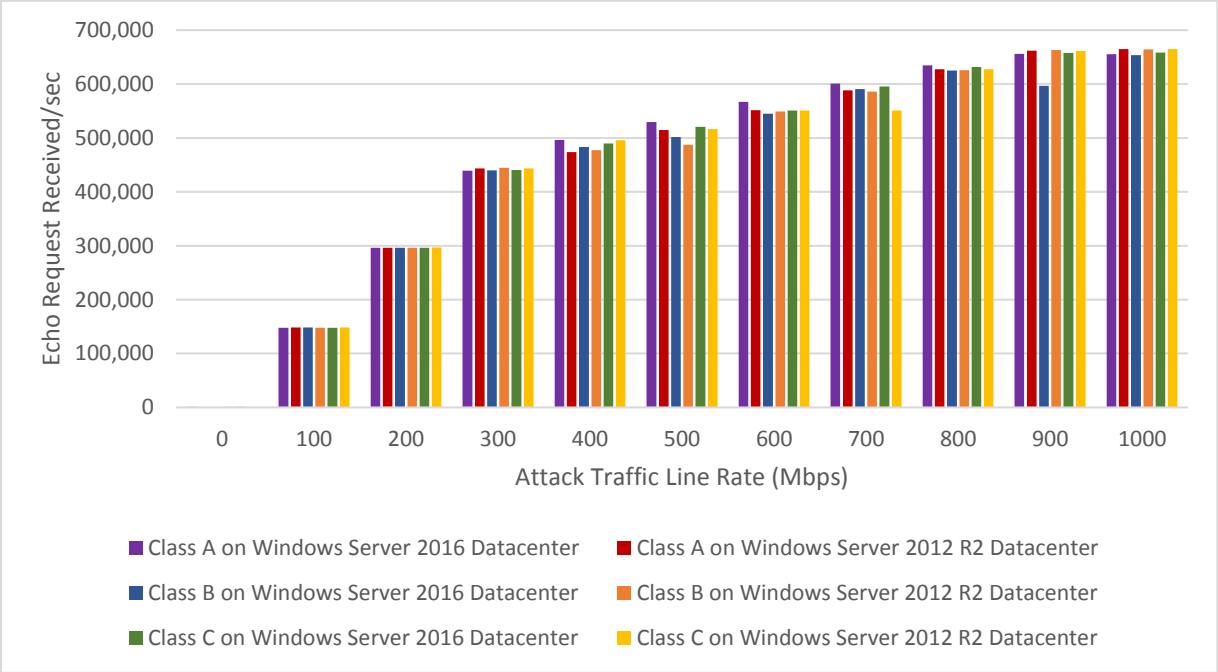


Figure 3.3 Echo Request Received per second under Ping Attacks on Windows Servers

Despite the number of ICMP Echo requests message received during the attack not a single ICMP Echo reply message was transmitted back to any of the spoofed IP addresses. This was not expected since RFC 792 states that for every ICMP Echo Request message received by a system will trigger it to transmit back to the source an ICMP Echo Reply message [61]. The behavior depicted by RFC 792 is present when a small amount of ICMP Echo request messages

are transmitted via the Ping Utility on the command prompt towards the targeted server. This would suggest that a security feature within both Windows Server versions will prevent ICMP Echo replies from being generated if a high amount of ICMP Echo request messages are received thus conserving some of its processing power.

While the simulated botnet networks did not have an impact on the amount of ICMP Echo requests received, they did have an impact in the connectivity of Windows Server 2012 R2. As Ping attack traffic was introduced the baseline performance of 3000 HTTP transactions/sec was maintained by Windows Server 2012 R2 up to 200 Mbps of attack traffic while under the effects of three simulated botnets. However, the operating system began to become unstable at an attack traffic line rate of 300 Mbps and subsequently experienced a considerable decline in connectivity at 400 Mbps where Class A, B, and C botnets denied an average of 2758.27, 2353.84, and 1928.36 HTTP transactions per second out of 3000 respectively. HTTP transactions/sec under a Class A botnet experienced a small recovery of 271.6 HTTP transactions/sec at 500 Mbps. However, attack traffic loads of 500 Mbps and higher yielded a gradual decline in HTTP transactions/sec for all simulated botnets. At 900 Mbps nearly 95% of HTTP transactions/sec under Class A and B botnets were denied while 84.25% were denied under a Class C botnet. Lastly, at line rate of 1 Gbps of attack traffic roughly 98% off all HTTP transactions/sec were lost. The network connectivity of Windows Server 2012 R2 under Ping attack is demonstrated in Figure 3.4.

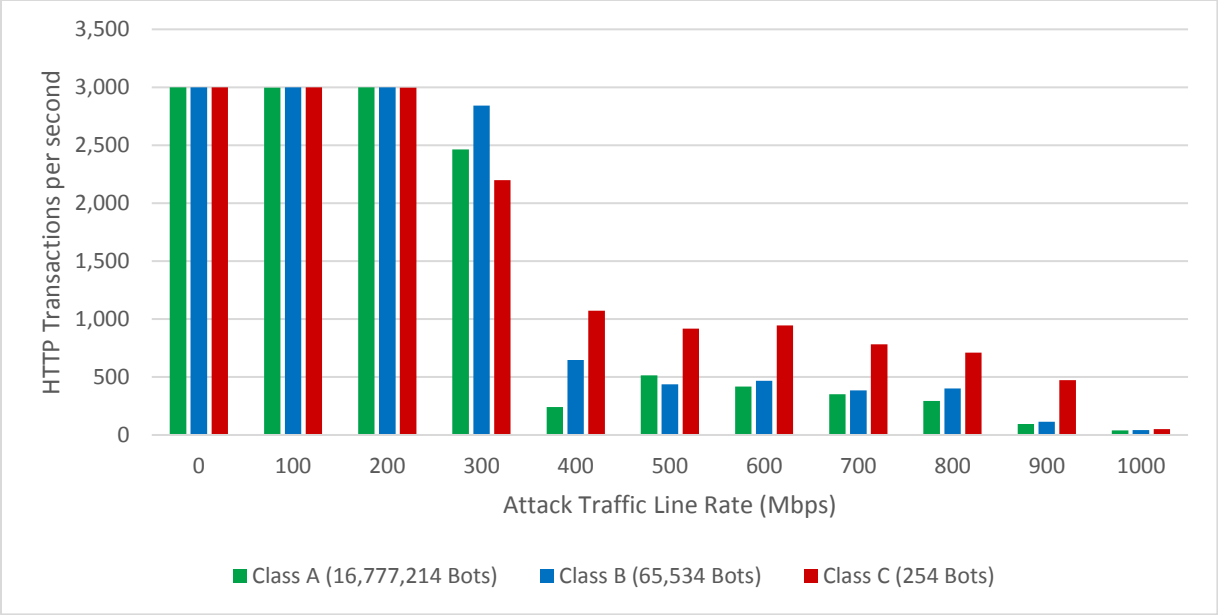


Figure 3.4 HTTP Transactions/sec under Ping Attacks on Windows Server 2012 R2

Windows Server 2016 experienced similar connectivity performance when compared to Windows Server 2012 R2, however the effectiveness of the Ping attack was not increased by the botnet size, HTTP Transaction/sec were similar at each load regardless of botnet size. As Ping attack traffic was introduced the baseline performance of 3000 HTTP transactions/sec was maintained by Windows Server 2016 up to 200 Mbps of attack traffic while under the effects of three simulated botnets in the same form as Windows Server 2012 R2. However, Windows Server 2016 began to become unstable at an attack traffic line rate of 300 Mbps with slightly better performance than its former release. Furthermore, the operating system experienced a considerable decline in connectivity at 400 Mbps where Class A, B, and C botnets denied an average of 2060.48, 2326.91, and 2284.77 HTTP transactions per second out of 3000 respectively. Traffic loads in the range of 400 Mbps to 800 Mbps saw a gradual decline in HTTP Transactions per second while at 900 Mbps nearly 95% of HTTP transactions/sec were denied under all three botnet sizes. Lastly, at line rate of 1 Gbps of attack traffic roughly 98% off all

HTTP transactions/sec were lost. The network connectivity of Windows Server 2016 under Ping attack is demonstrated in Figure 3.5.

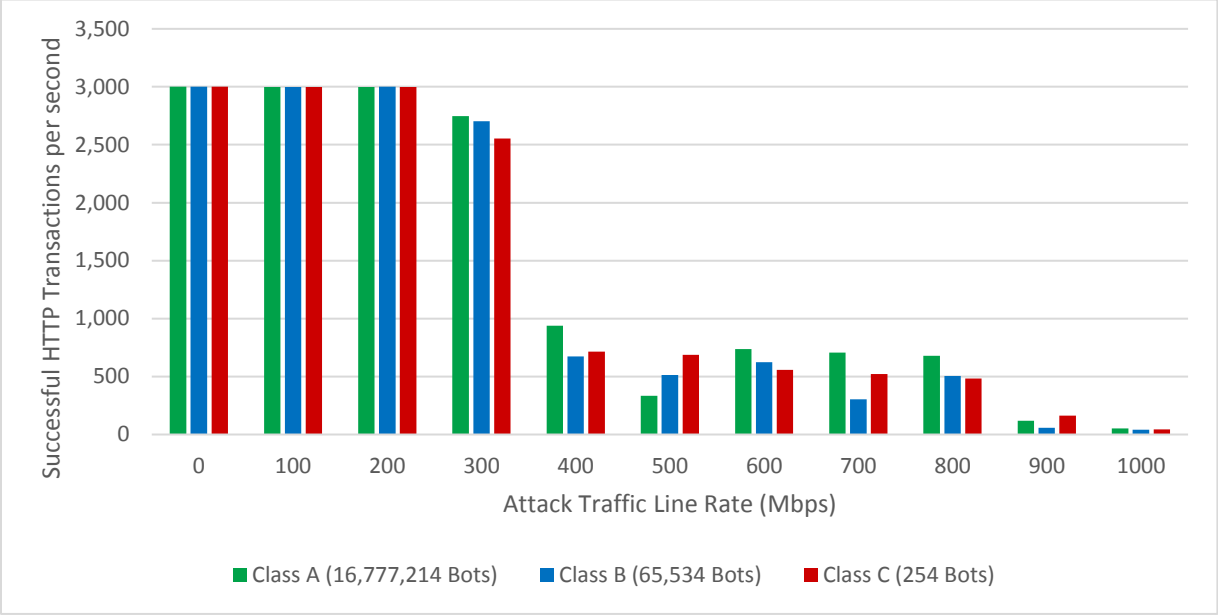


Figure 3.5 HTTP Transactions/sec per second under Ping Attacks on Windows Server 2016

In this experiments Ping attacks were found to have low impact on the consumption of random access memory (RAM). These attacks are not memory intensive, the difference in memory consumed by the Ping attack regardless of botnet size is of 40.03 MB for Windows Server 2012 R2 and 85.02 MB for Windows Server 2016. The server hardware used for these attacks features 16 GB of RAM meanwhile most modern systems are deployed with RAM in the ranges of 4 GB, to 32 GB therefore the impact of Ping attacks on memory is negligible for modern systems. Nevertheless, attack traffic loads exacerbated the rate at which RAM is utilized for both server operating systems. The overall memory consumption of both operating systems under Ping attack can be seen in Figures 3.6, 3.7. While under attack Windows Server 2012 R2 exhibited better performance than Windows Server 2016 in terms of memory consumption.

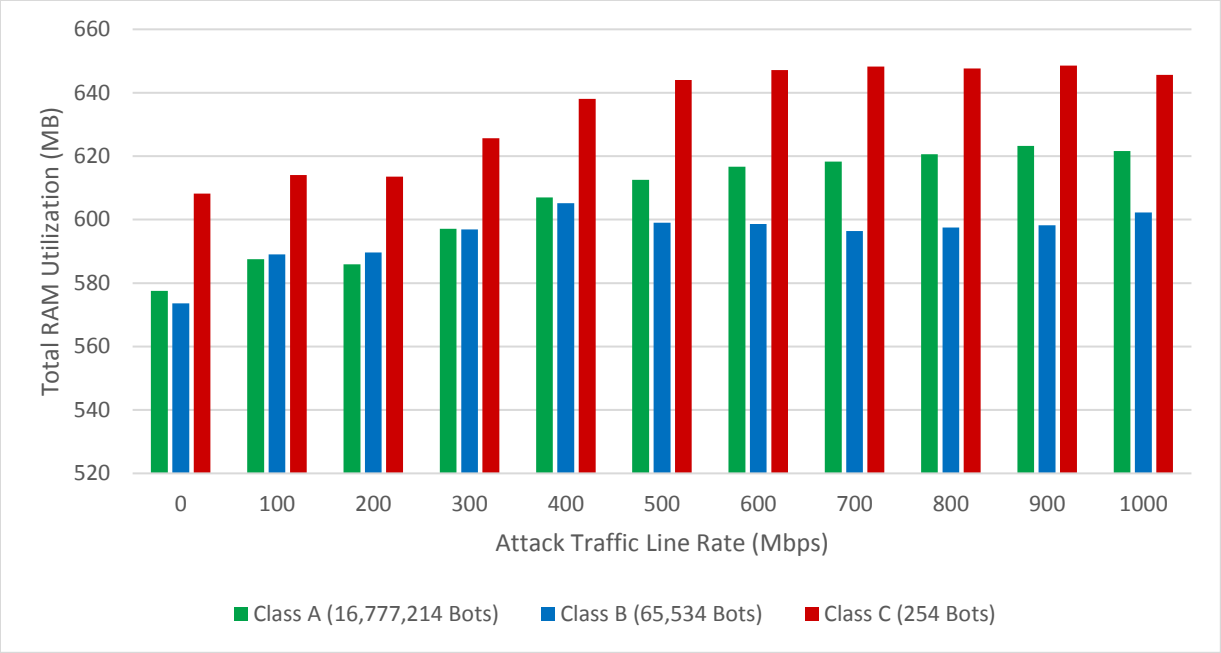


Figure 3.6 Total Memory Consumption under Ping Attacks on Windows Server 2012 R2

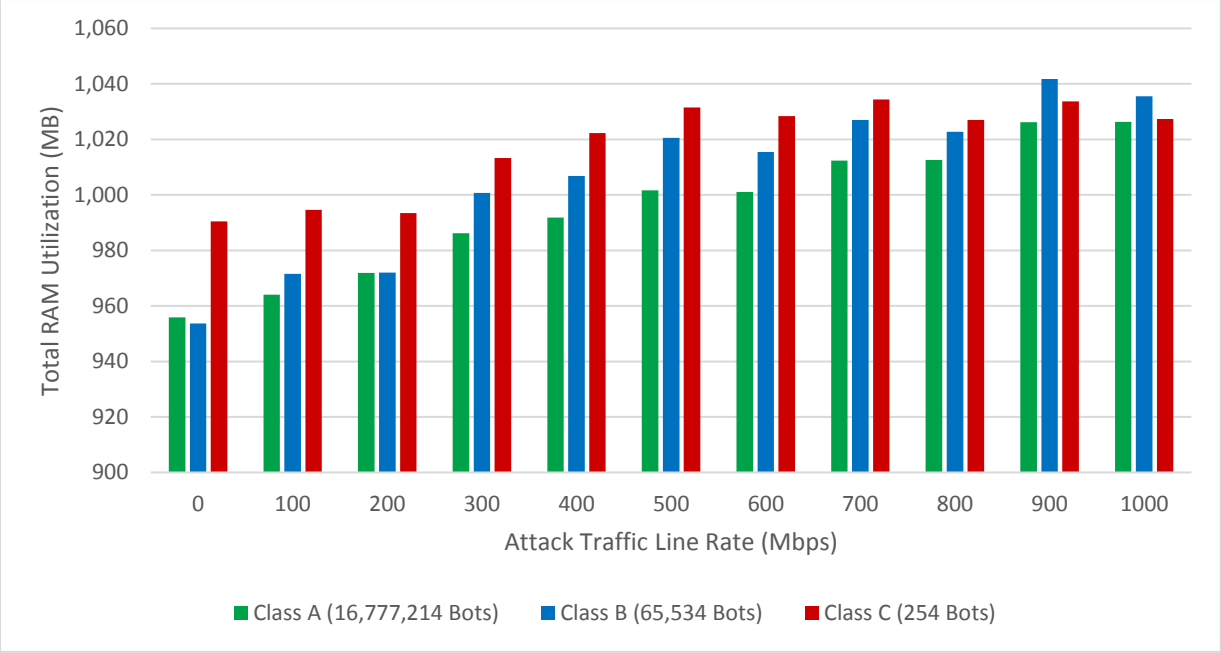


Figure 3.7 Total Memory Consumption under Ping Attacks on Windows Server 2016

Windows Server 2016 was found to overall consume nearly twice the amount of memory Windows Server 2012 R2 does. This can be a direct result of new features implemented in the

newer version which would require additional resources to maintain. Interestingly, during experimental trials for Class C Ping attacks a higher RAM consumption was demonstrated for both server operating systems. These trends are not a result of the introduction of attack traffic as this were preexisting conditions. It is unclear if background processes required a higher amount of memory to run at the time of the experimental trials as the difference in memory consumed is rather small. In Figures 3.8 and 3.9 the memory consumed during the baseline performance is subtracted from that of each attack load, this is done to accurate visualize the impact of the Ping attacks on memory and how botnet size and attack traffic load affect it.

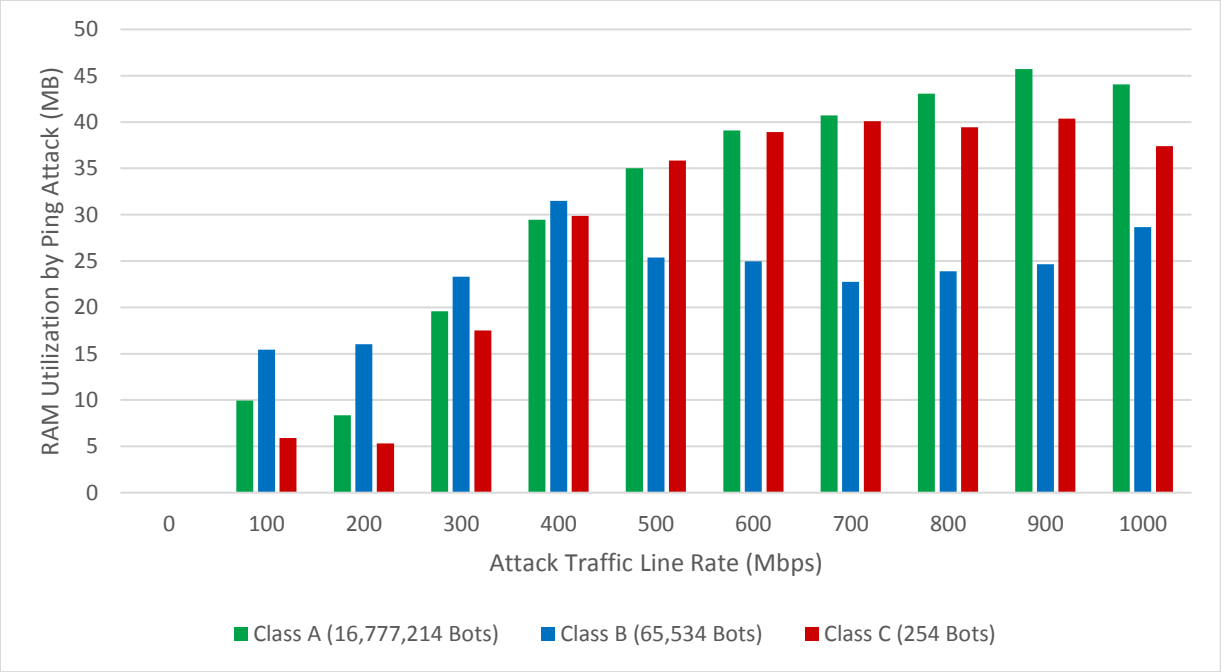


Figure 3.8 Memory Consumed on Windows Server 2012 R2 by Ping Attacks

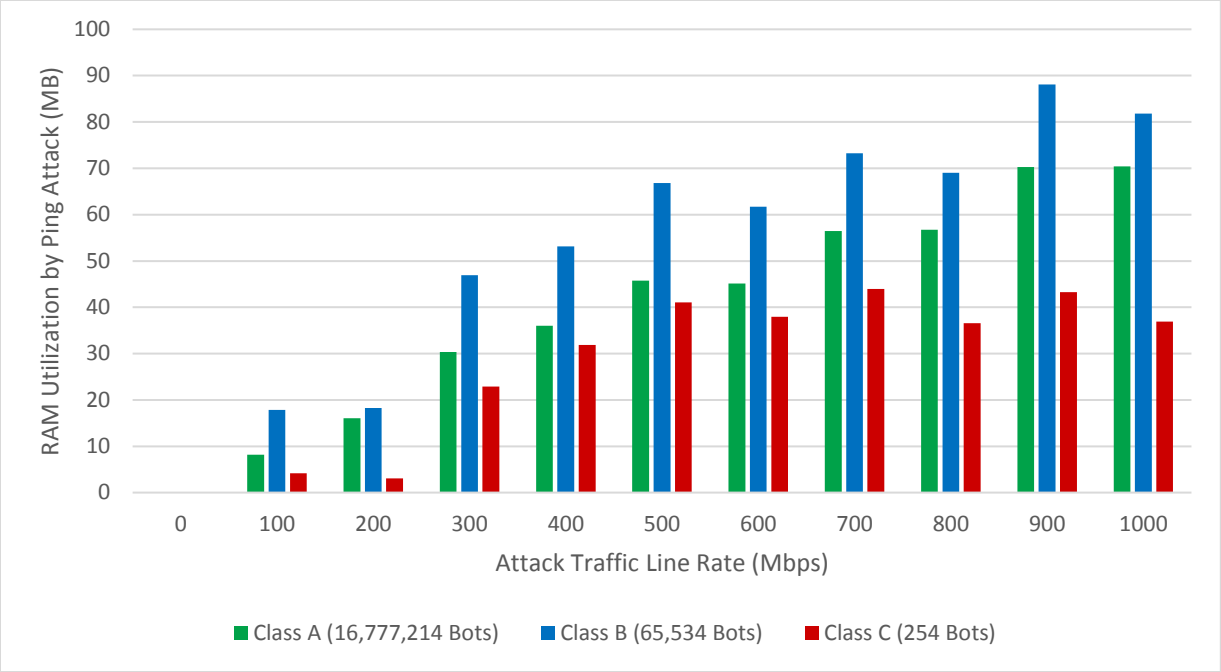


Figure 3.9 Memory Consumed on Windows Server 2016 by Ping Attacks

The memory consumed by Ping attacks on both server operating systems do not suggest an impact on memory consumption was achieved by the botnet size but rather the attack traffic load at which the echo request messages were being transmitted. Memory consumed by Ping attacks for both server operating systems was similar at lower loads in the range of 100 to 200 Mbps however starting at 300 Mbps the consumption of memory begins to gradually rise in small increments with each attack load until 900 Mbps with slight decline at 1 Gbps where nearly no HTTP connections can reach the server explaining the decreases in memory used. In these attacks it was observed that the memory consumptions caused by Ping attacks on Windows Server 2016 was higher than Windows Server 2012 R2, where at the higher loads the consumption is nearly twice than that of its predecessor.

Botnet magnitude did not have an impact on the consumption of processing time for either operating system. As seen earlier in this subsection, the number of ICMP echo requests

transmitted to the targeted server was reported to be similar for all botnet sizes, and no echo replies were transmitted back which would indicate the CPU processed the same amount of network traffic regardless of botnet size. The impact of Ping attacks on both platforms had similar results which did not favor either server operating system. In these experiments a higher amount of processing power was observed in iterations of the attack with a lower line rate at 100 Mbps and 200 Mbps which consumed 14 and 18% of processing power respectively, a decline in consumption was experienced at 300 Mbps and 400 Mbps to 11%, iterations of the attack at higher loads saw a consumption in the range of 12% to 14%. Total CPU utilization is demonstrated in Figures 3.10 and 3.11

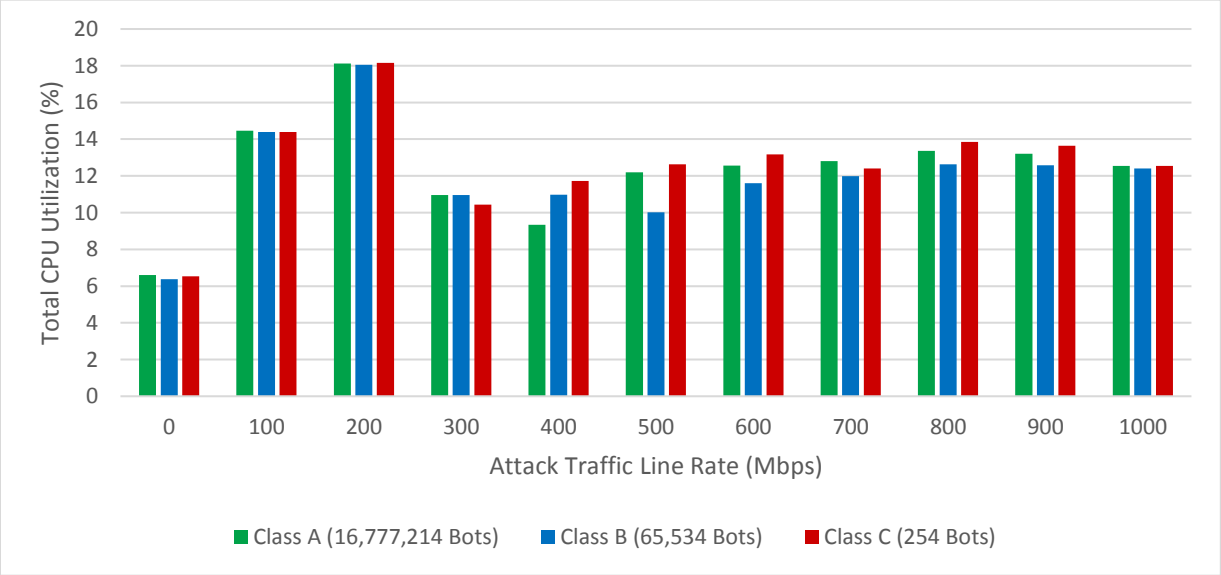


Figure 3.10 CPU Depletion under Ping Attacks on Windows Server 2012 R2

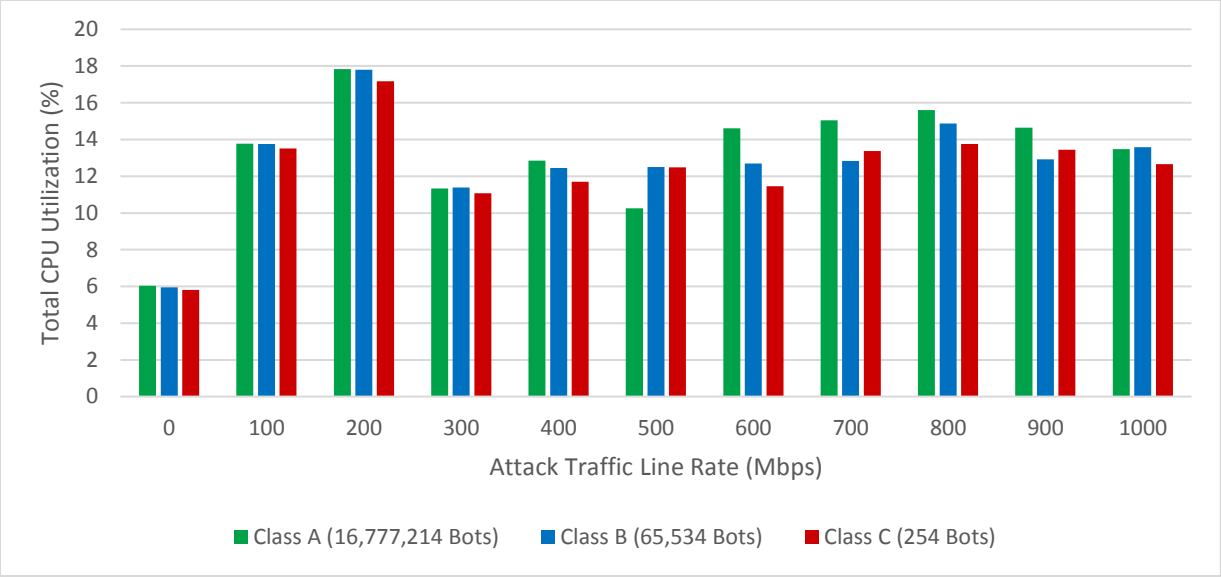


Figure 3.11 CPU Depletion under Ping Attacks on Windows Server 2016

During Ping attacks it was observed that in both server operating systems only half of the virtual cores experienced any consumption of processing time because of the attack. All simulated botnets consumed a similar amount of processing power therefore the average core utilization for Ping attacks is demonstrated thru Figures 3.12 and 3.13.

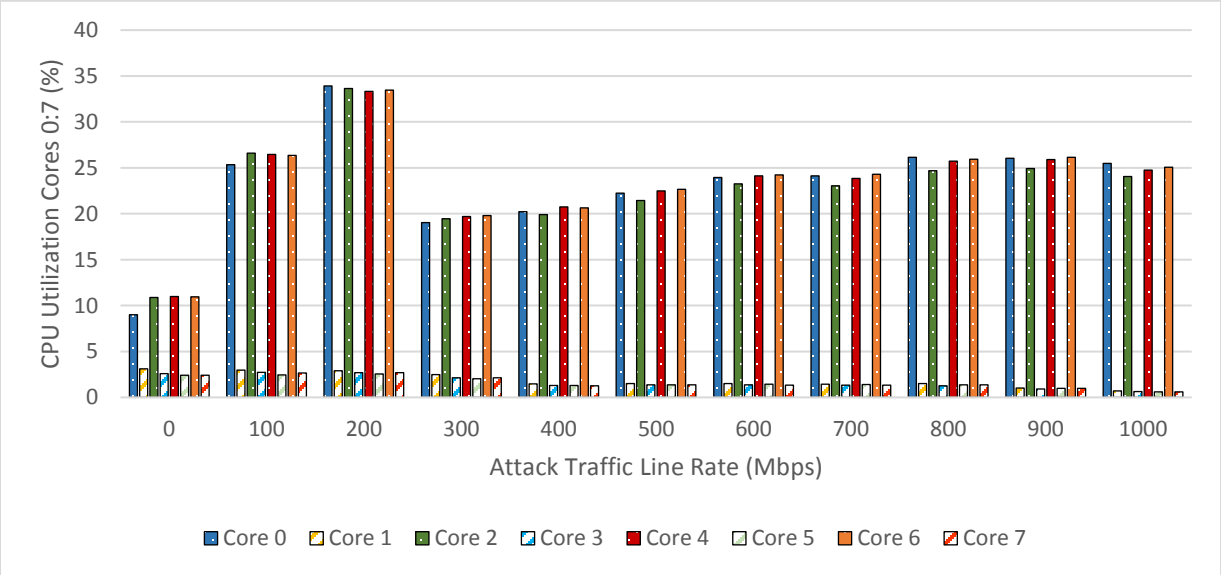


Figure 3.12 CPU Core Utilization under Ping Attacks on Windows Server 2012 R2

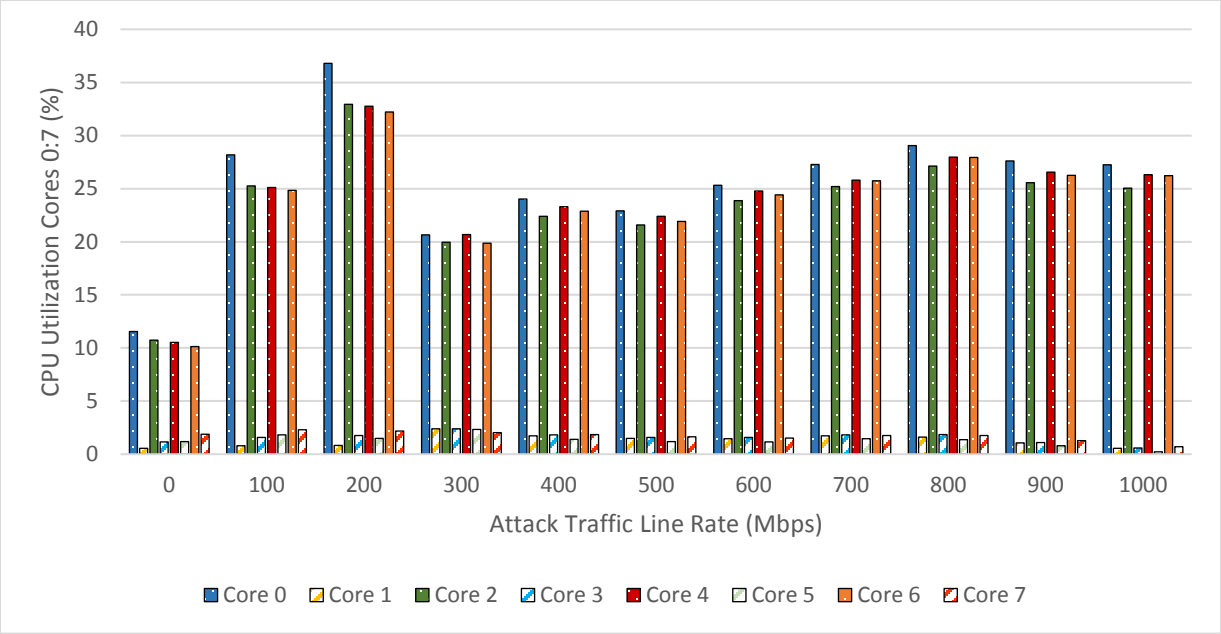


Figure 3.13 CPU Core Utilization under Ping Attacks on Windows Server 2016

Ping attacks did not affect half of the virtual cores, this would imply that the network functions of Windows do not take advantage of Hyperthreading technology which helps mitigate the effect of these attack on the targeted system. Lastly CPU temperatures reach their highest point under attack traffic loads of 100 Mbps and 200 Mbps reaching 61-63 °C and 69-71 °C respectively which is still below TjMax. The temperatures of the CPU reflect the amount of processing power consumed and can be seen in Figures 3.14 and 3.15.

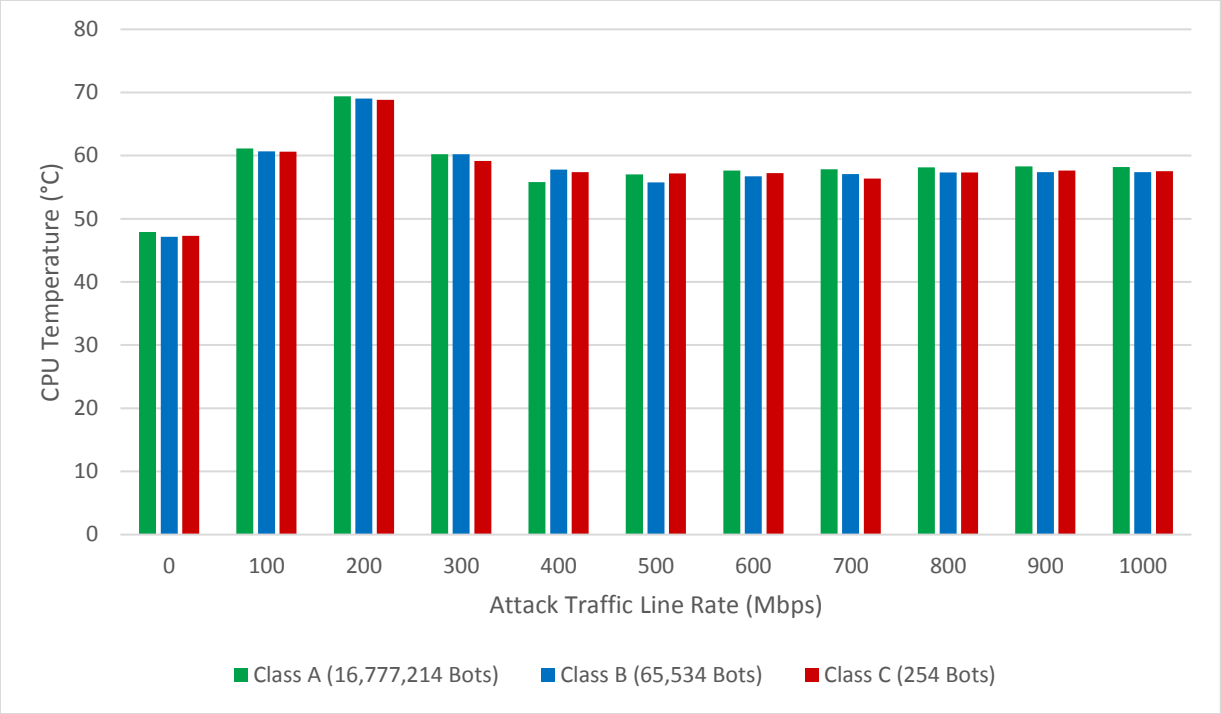


Figure 3.14 CPU Temperature under Ping Attacks on Windows Server 2012 R2

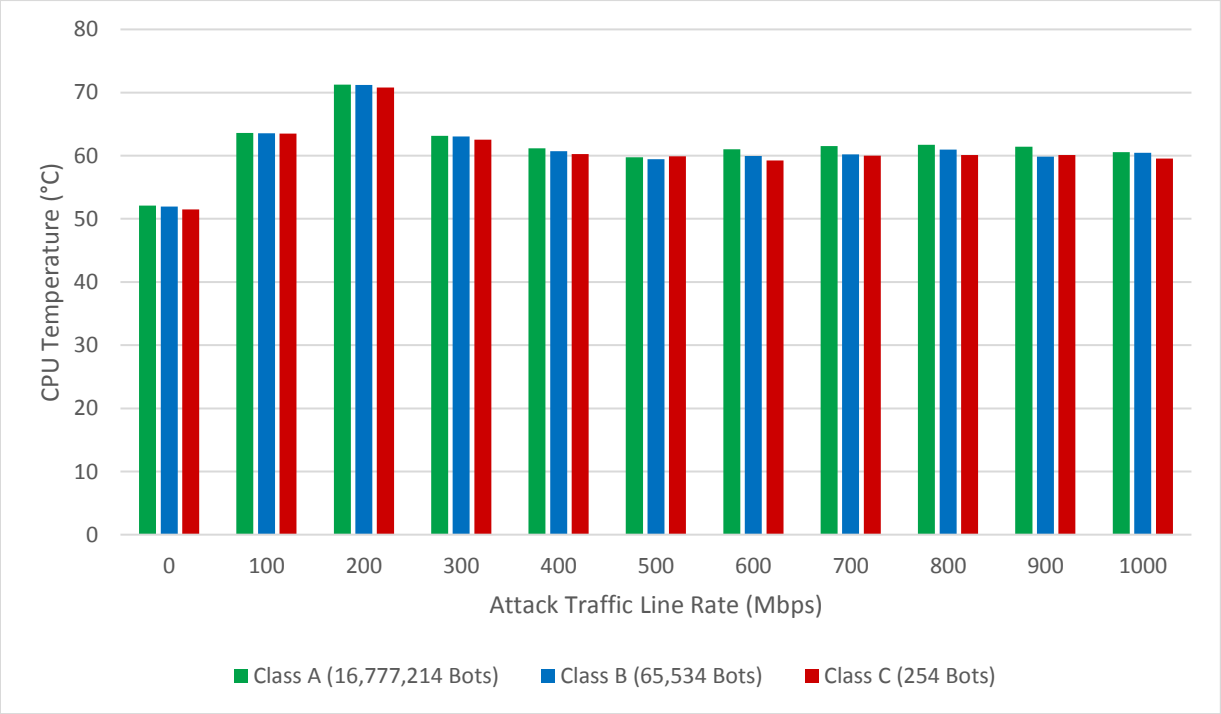


Figure 3.15 CPU Temperature under Ping Attacks on Windows Server 2016

3.3.2 Layer 3 ICMP Smurf Attack

During the Smurf attacks it was observed that the botnet size had little to no impact on the average amount of Echo Replies received by both Windows Server operating systems. As demonstrated in Figure 3.16 regardless of the simulated botnet network employed the average amount of ICMP Echo Request messages received per second was kept within the same range for both server operating systems where only the line rate of attack traffic would impact this number with an exception for Class A Smurf attack on Windows Server 2016 which experienced a higher number or echo requests received, and Class C Smurf attack on Windows Server 2012 R2 which received less Echo replies at higher traffic loads.

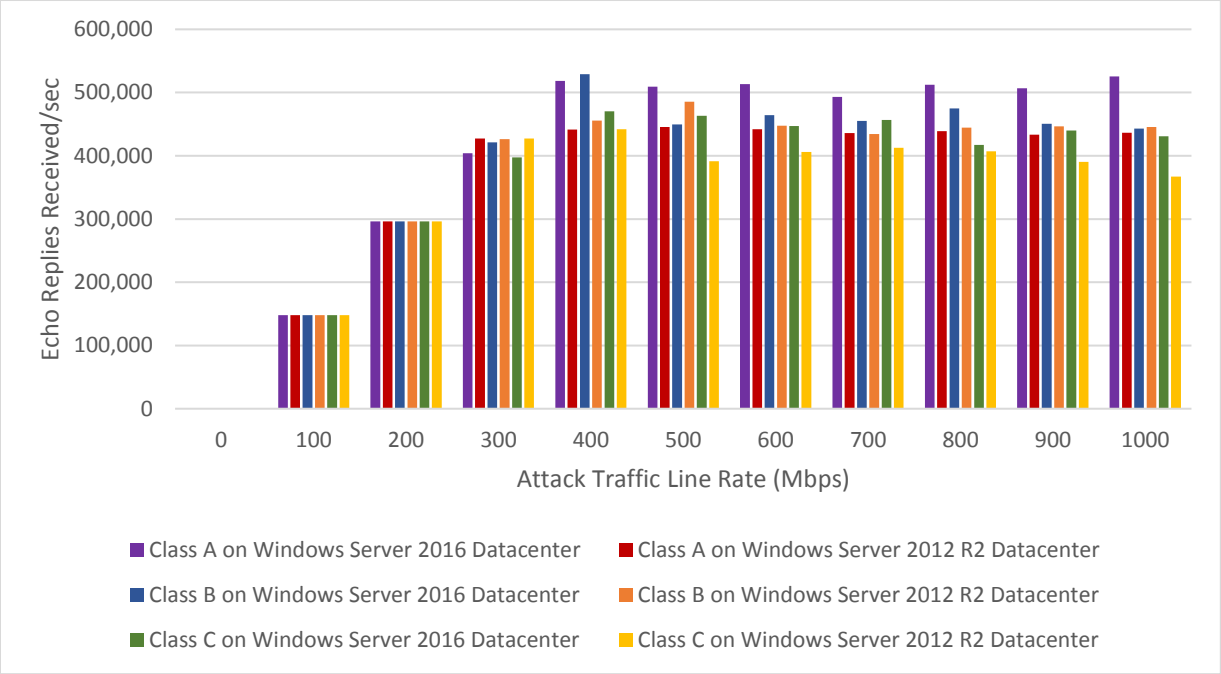


Figure 3.16 Echo Request Received per second under Smurf Attacks on Windows Servers

While the simulated botnet networks had little to no impact on the amount of ICMP Echo replies received, they did have an impact in the connectivity of both Windows Server operating systems. As Smurf attack traffic was introduced the baseline performance of 3000 HTTP

transactions/sec was maintained by Windows Server 2012 R2 up to 200 Mbps of attack traffic while under the effects of three simulated botnets. However, the number of HTTP transactions/sec declined drastically at 300 Mbps where a Class A botnet compromised 90.56% of the HTTP Transaction/sec while a Class B botnet compromised 88.90% and Class C compromised 88.09%. Attack traffic at 400 Mbps diminished HTTP Transaction/sec to nearly 70 transactions/sec while attack loads in the range of 500 Mbps to 1 Gbps extinguished most HTTP connections. The network connectivity of Windows Server 2012 R2 under Smurf attack is demonstrated in Figure 3.17.

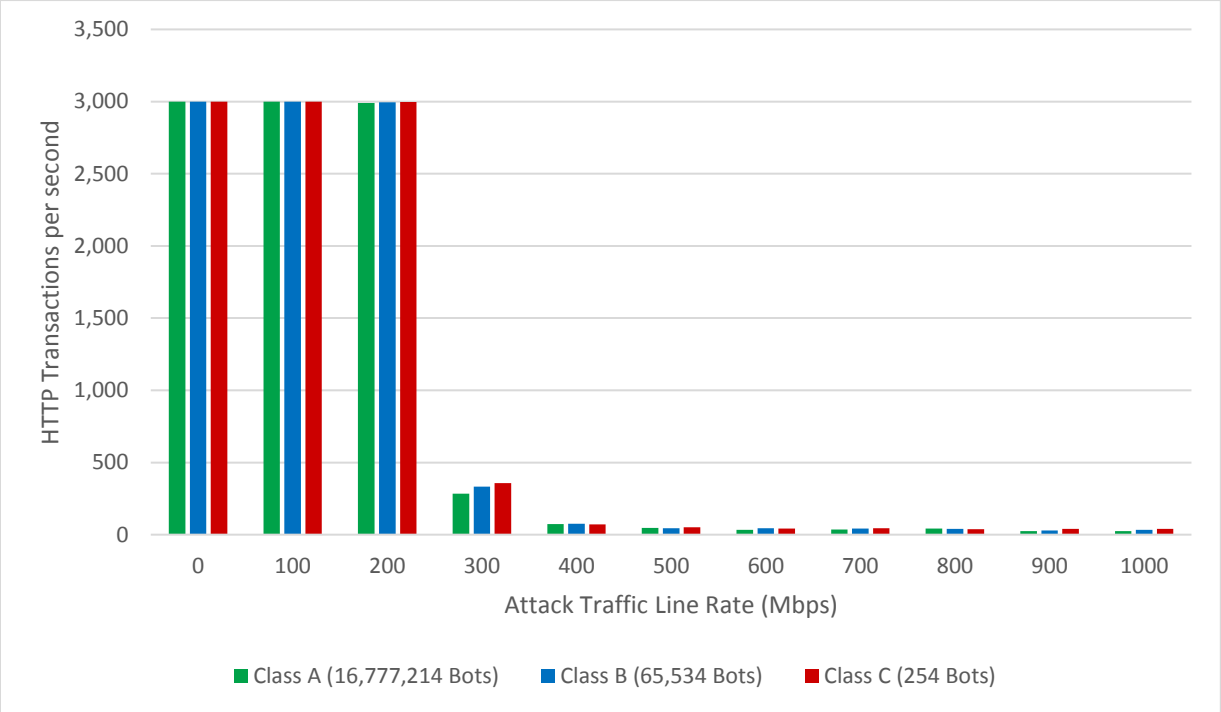


Figure 3.17 HTTP Transactions/sec under Smurf Attacks on Windows Server 2012 R2

The effectiveness of Smurf attacks against the connection rate on Windows 2016 was decreased for Class C botnets when compared with Windows 2012 R2. Windows Server 2016 similarly to Windows Server 2012 R2 was able to maintain the baseline performance of 3000

HTTP transactions/sec up to 200 Mbps of attack traffic while under the effects of three simulated botnets. Class A and B botnets drastically affected the number of HTTP transactions/sec at 300 Mbps where a Class A botnet compromised 84.94% of the HTTP Transaction/sec while a Class B botnet compromised 92.32%. Higher traffic loads for Class A and B botnets in the range of 400 Mbps and 1 Gbps diminished the servers connection rate heavily and most HTTP connections were lost. However, Smurf attacks performed using a simulated Class C botnet allowed HTTP connections to have some survivability. At 300 Mbps Smurf attacks under a Class C botnet compromised 51.17% HTTP transaction/sec. As attack traffic load increased the connection rate gradually was lost until 1Gbps where only an average of 103.38 HTTP transactions/sec were able to be maintain. The network connectivity of Windows Server 2016 under Smurf attacks is demonstrated in Figure 3.18.

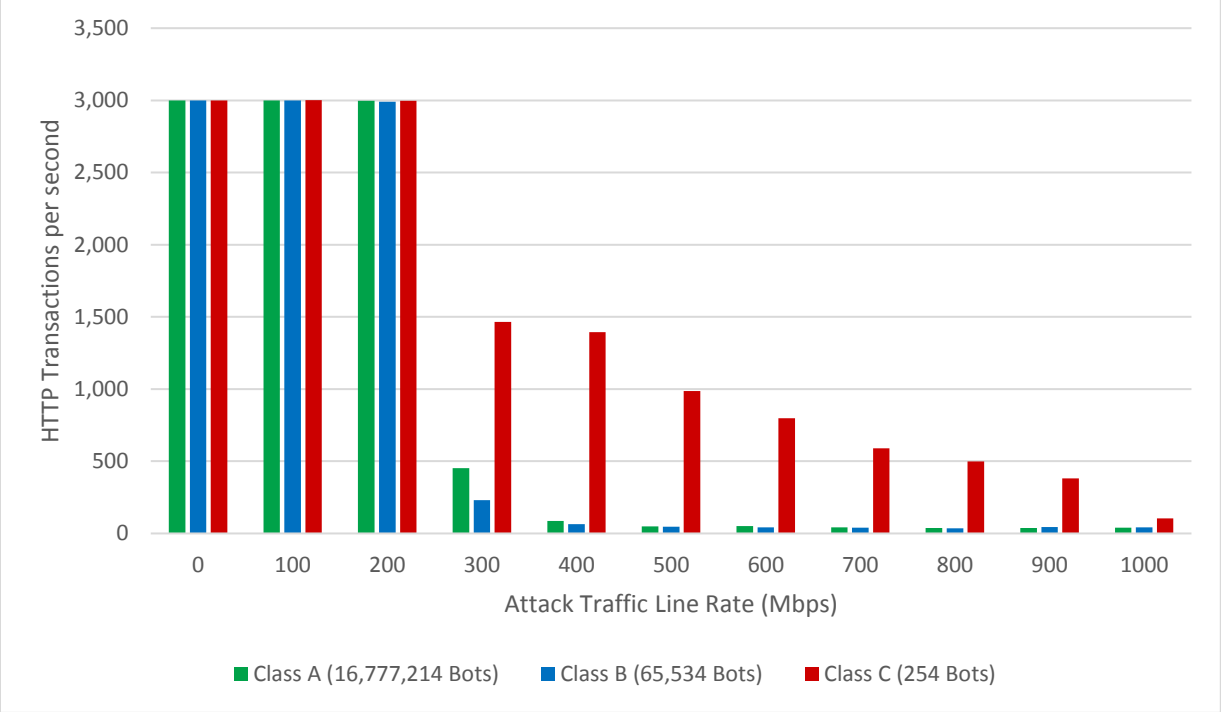


Figure 3.18 HTTP Transactions/sec under Smurf Attacks on Windows Server 2016

The amount of memory consumed during Smurf attacks was relatively low. Similarly, to Ping Attacks, data collected for Smurf attacks would suggest this form of DDoS attack is not memory intensive either. The difference in total RAM consumed regardless of botnet size is of 77.52 MB for Windows Server 2012 R2 and 101.47 MB for Windows Server 2016. The system characteristics of the server under attack feature 16 GB, at such a low memory consumption Smurf attacks can be considered negligible for modern systems in terms of memory. Windows Server 2012 R2 experienced lower memory consumption than Windows Server 2016 under baseline performance Windows Server 2012 R2 consumed memory in range of 573.65 MB to 628.45 MB while Window Server 2016 consumed nearly twice the amounts with a range of 943.94 MB to 1022.14 MB. While botnet sizes did not have a clear impact on memory consumption, higher attack traffic loads intensified the rate at which RAM is utilized for both server operating systems. The overall memory consumption of both operating systems under Smurf attacks can be seen in Figures 3.19, 3.20.

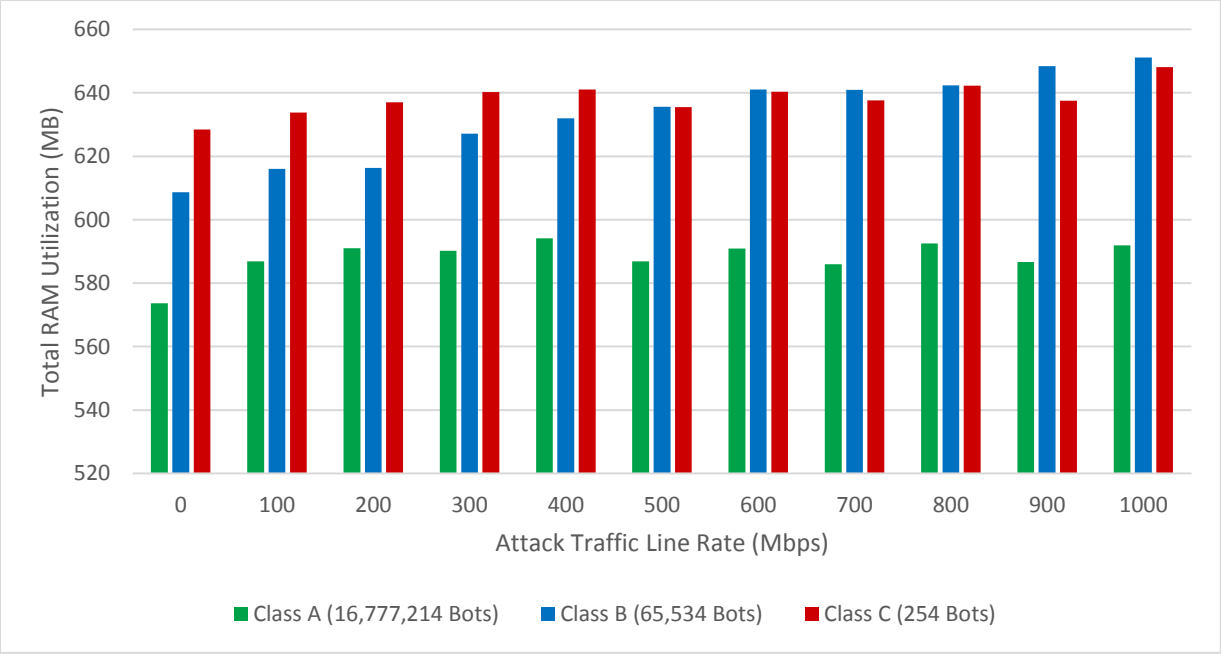


Figure 3.19 Total Memory Consumption under Smurf Attacks on Windows Server 2012 R2

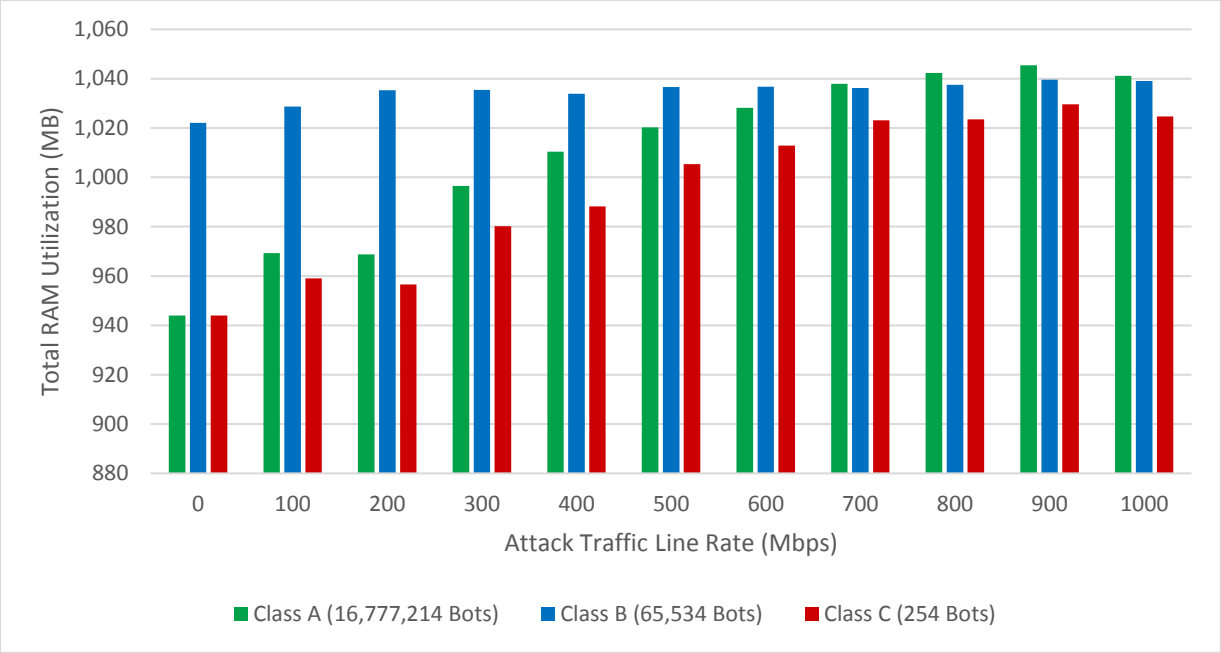


Figure 3.20 Total Memory Consumption under Smurf Attacks on Windows Server 2016

Simulated Class B and C botnets appears to trigger a higher amount of memory consumption than their larger counterpart on Windows Server 2012 R2, while simulated a Class B botnet appears to consume more memory than the larger simulated Class A botnet for Windows Server 2016. These trends are not a result of the introduction of attack traffic as this were preexisting conditions. It is unclear if background processes required a higher amount of memory to run at the time of the experimental trials as the difference in memory consumed is rather small. In Figures 3.21 and 3.22 the memory consumed during the baseline performance is subtracted from that of each attack load, this is done to accurate visualize the impact of the Smurf attacks on memory and how botnet size and attack traffic load affect it.

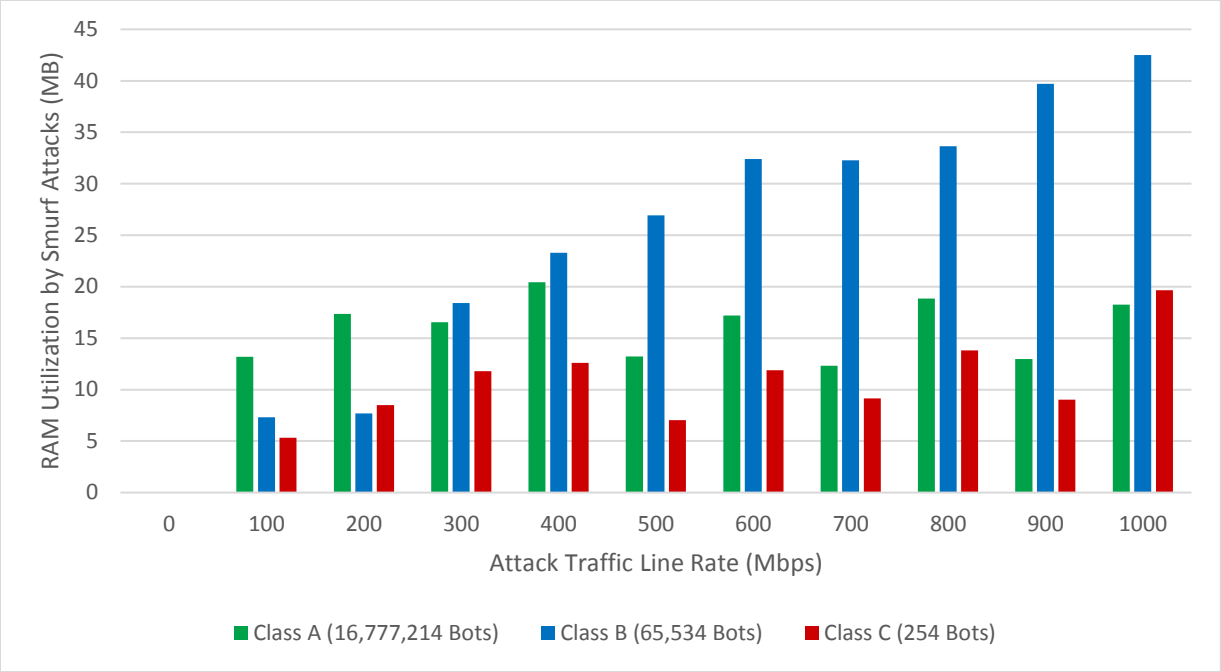


Figure 3.21 Memory Consumed on Windows Server 2012 R2 by Smurf Attacks

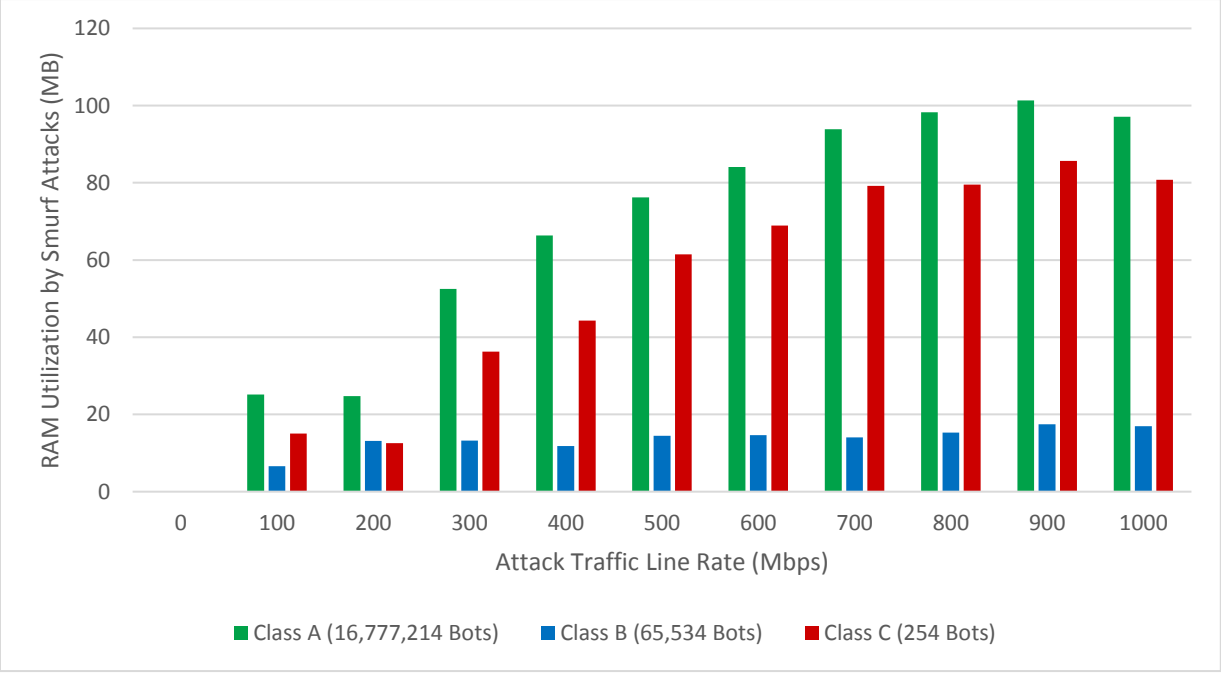


Figure 3.22 Memory Consumed on Windows Server 2016 by Smurf Attacks

Figures 3.21 and 3.22 do not depict a trend where botnet size is related to memory consumption however attack traffic load present a negative impact to memory consumption. Memory consumed by Smurf attacks for both server operating systems was similar at lower loads in the range of 100 to 200 Mbps however starting at 300 Mbps the consumption of memory begins to gradually rise in small increments with each attack load. In these attacks it was observed Windows Server 2016 can consumed approximately 100 MB at high attack traffic loads while the memory consumption of Windows Server 2012 R2 is nearly 40 MB at high attack traffic loads .

In these experiments the botnet size had low to no impact in the effectiveness of Smurf attacks at different attack traffic loads in terms of CPU utilization. Earlier in this subsection it was demonstrated that the number of ICMP echo replies received by the targeted server was reported to be similar for all botnet sizes with a couple of exceptions for Class A botnets where a high amount of echo replies were received. A higher amount of echo replies would explain would explain why CPU utilization was higher for Class A botnets when compared with Class B and C. The impact of Smurf attacks was found to be more detrimental for Windows Server 2016 which on average consumed 6% more processing power than Windows Server 2012 R2 thus favoring its predecessor. All three botnet sizes had similar effectiveness on Windows Server 2012 R2. At baseline performance Windows Server 2012 R2 CPU utilization was kept at approximately 6.82%. The introduction of Smurf attack traffic saw CPU consumption rise to nearly 25% at 100 Mbps, as the traffic load increased beyond 100 Mbps CPU utilization was nearly 40% where Class A botnets would consume roughly 1.5% more CPU power than lesser botnet sizes. Meanwhile CPU consumption on Windows Server 2016 had a CPU utilization of approximately 6.28%. The introduction of Smurf attack traffic saw CPU consumption rise to

nearly 29% at 100 Mbps, as the traffic load increased beyond 100 Mbps CPU utilization was in the range of 40% to 50% for most traffic load. Total CPU utilization is demonstrated in Figures 3.23 and 3.24

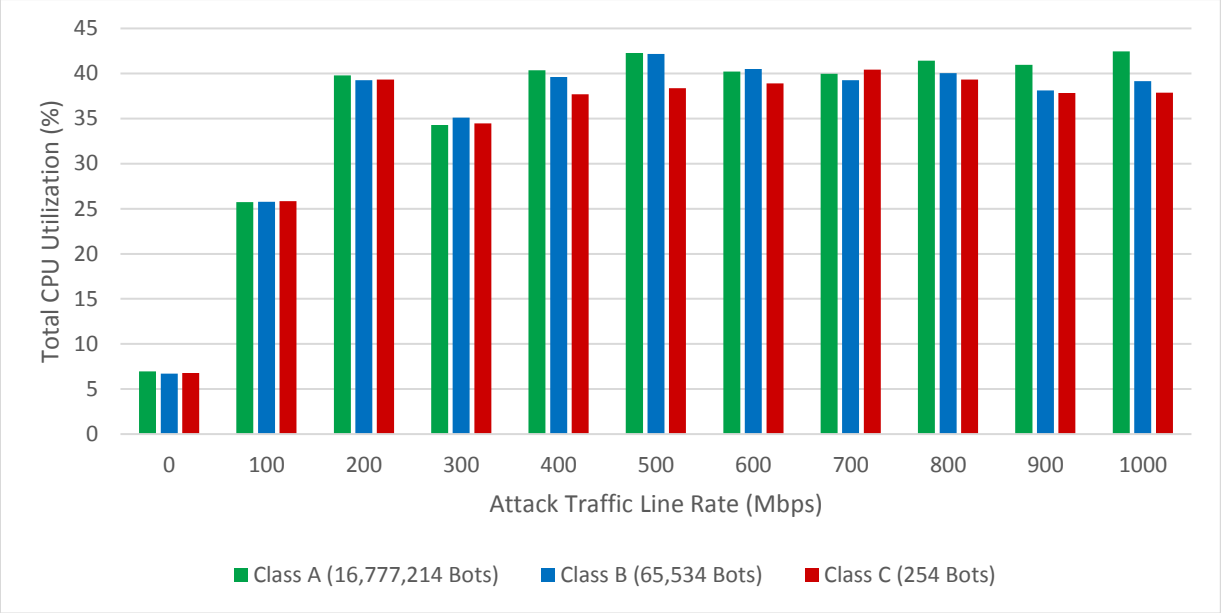


Figure 3.23 CPU Depletion under Smurf Attacks on Windows Server 2012 R2

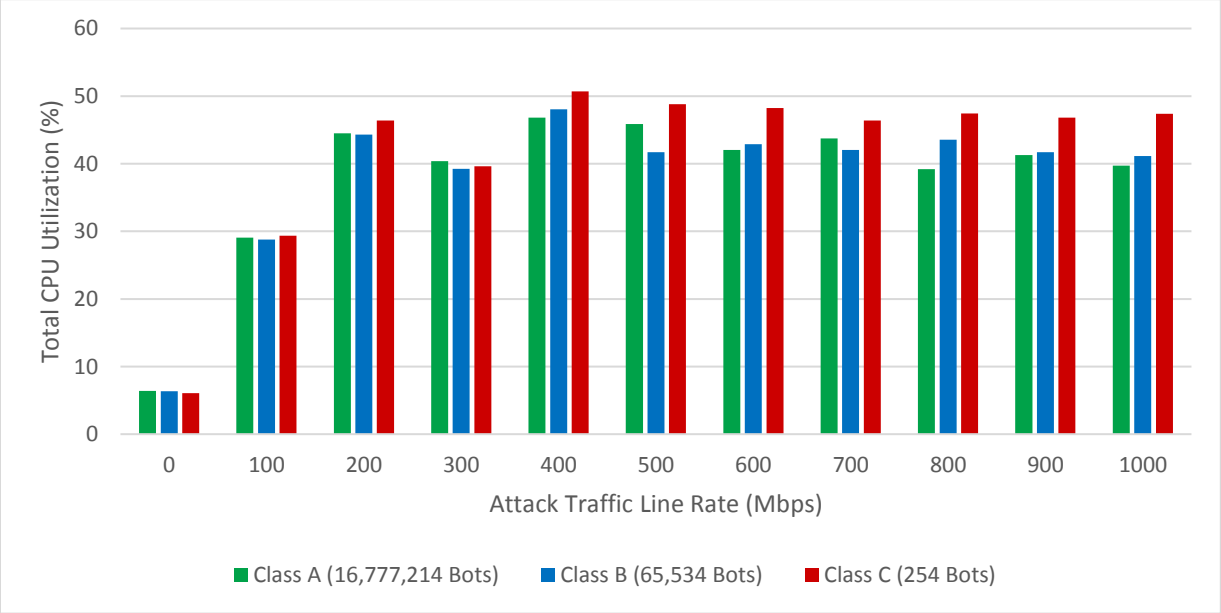


Figure 3.24 CPU Depletion under Smurf Attacks on Windows Server 2016

During Smurf attacks it was observed that in both server operating systems only half of the virtual cores experienced any consumption of processing time as a result of the attack. All simulated botnets consumed a similar amount of processing power therefore the average core utilization for Smurf attacks is demonstrated thru Figures 3.25 and 3.26.

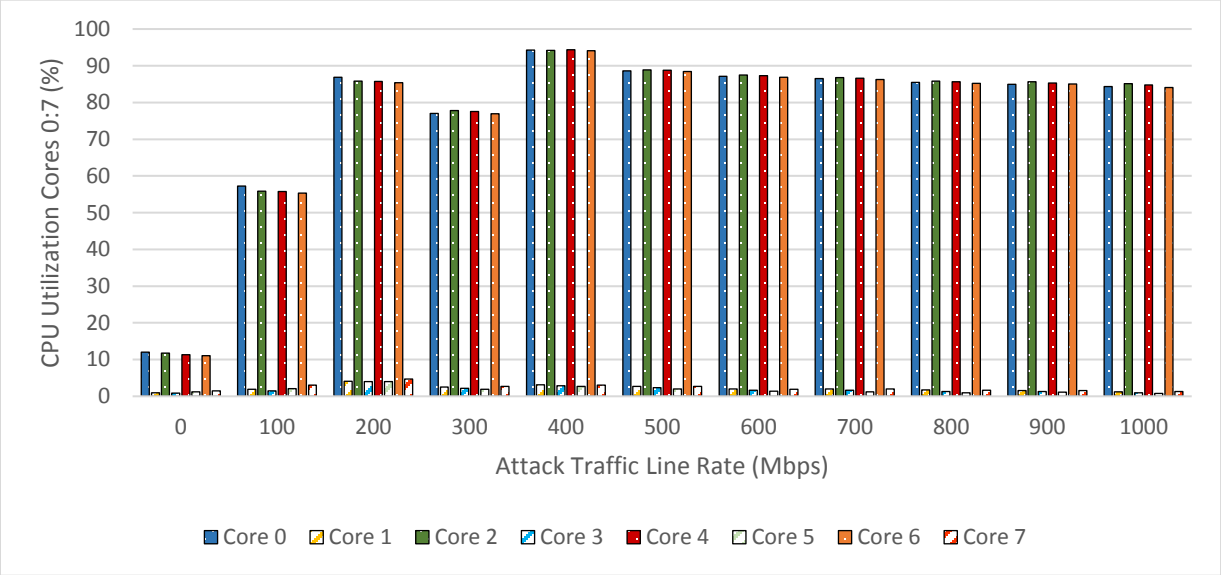


Figure 3.25 CPU Core Utilization under Smurf Attacks on Windows Server 2012 R2

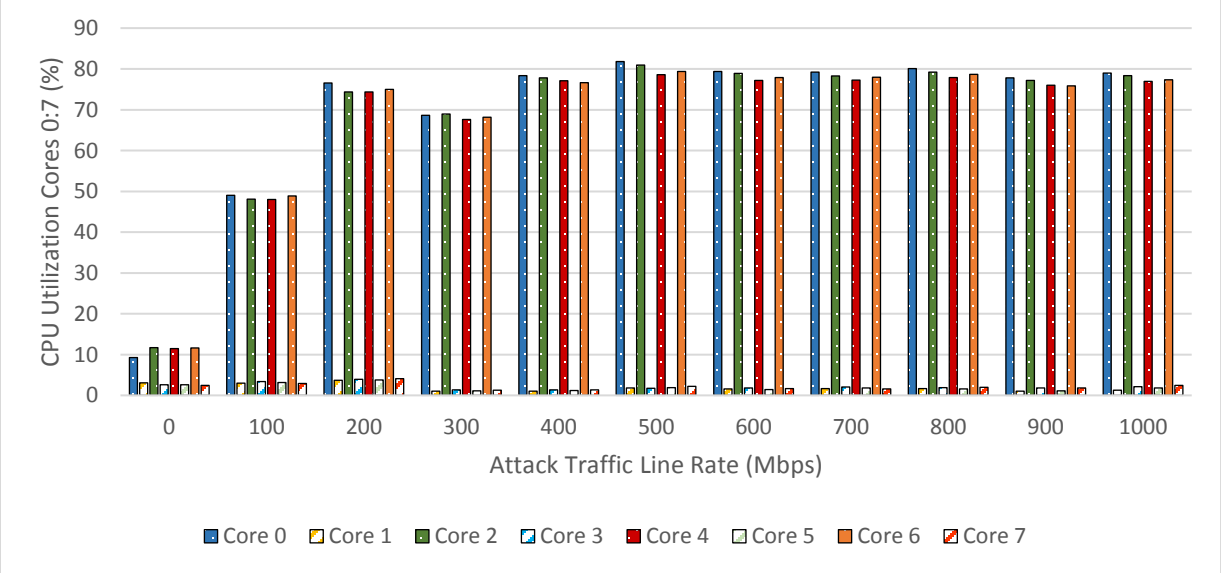


Figure 3.26 CPU Core Utilization under Smurf Attacks on Windows Server 2016

Smurf attacks did not affect half of the virtual cores, this would imply that the network functions of Windows do not take advantage of Hyperthreading technology which helps mitigate the effect of these attack on the targeted system. Lastly CPU temperatures on Windows Server 2012 R2 reached their highest point under attack traffic loads in the range of 500 Mbps and 1 Gbps reaching nearly 90 °C, while temperatures on Windows Server 2016 reached their highest point under attack traffic loads in the range of 400 Mbps and 1 Gbps reaching nearly 90 °C as well, which is still below TjMax but are not considered safe. The temperatures of the CPU reflect the amount of processing power consumed and can be seen in Figures 3.27 and 3.28.

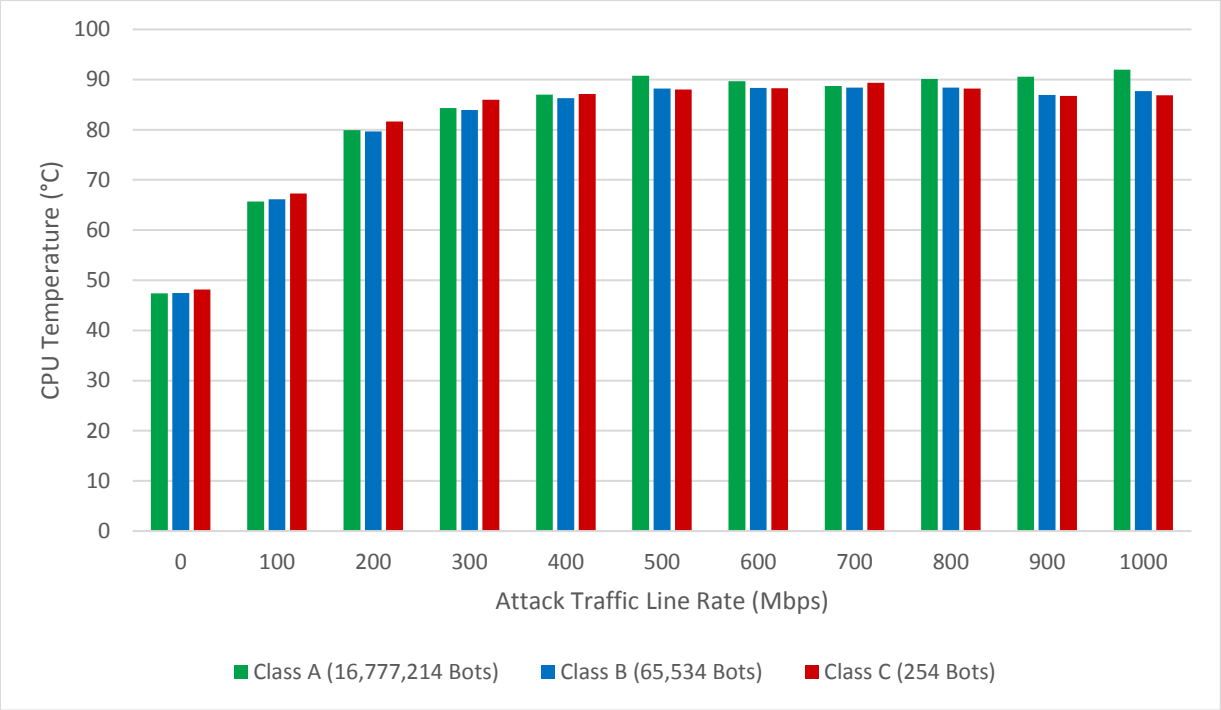


Figure 3.27 CPU Temperature under Smurf Attacks on Windows Server 2012 R2

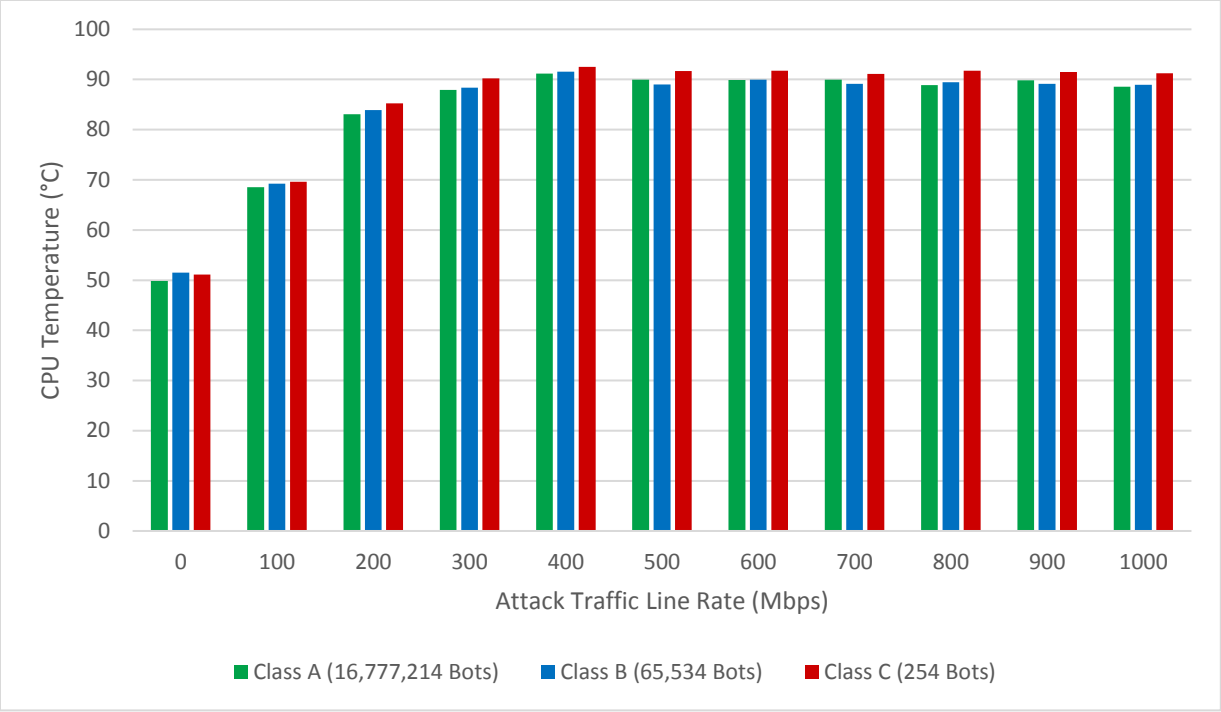


Figure 3.28 CPU Temperature under Smurf Attacks on Windows Server 2016

3.3.3 Layer 4 TCP SYN Flood Attack

The effectiveness of TCP SYN Flooding attacks was slightly increased with a higher botnet size particularly on Windows Server 2012 R2. As TCP SYN attack traffic was introduced the baseline performance of 3000 HTTP transactions/sec was maintained by Windows Server 2012 R2 up to 200 Mbps of attack traffic while under the effects of three simulated botnets. However, the number of HTTP transactions/sec declined drastically at 300 Mbps where a Class A botnet compromised 86.7% of the HTTP Transaction/sec while a Class B botnet compromised 70.23% and Class C compromised 61.58%. Attack traffic loads in the range of 400 Mbps to 1 Gbps saw a gradual decline from 478.78 HTTP connections to nearly none. However, it was observed through all experimental trials that a small portion of HTTP Transactions was recovered at 600 Mbps. It is unclear why this recovery is present considering CPU utilization is

higher at this load. Further experimentation and insight into defense mechanism and background processes implemented by Microsoft might address this anomaly. The network connectivity of Windows Server 2012 R2 under TCP SYN Flooding attacks is demonstrated in Figure 3.29.

Furthermore, Windows Server 2016 experienced less fluctuation on network connectivity based on the impact of botnet sizes. As connection TCP connection requests began to flood the targeted server the baseline performance of 3000 HTTP transactions/sec was maintained by Windows Server 2016 up to 200 Mbps of attack traffic while under the effects of three simulated botnets. However, a decrease in the number of HTTP transactions/sec was experienced at 300 Mbps where all three botnets compromised 65% of the HTTP Transaction/sec. Attack traffic loads in the range of 400 Mbps to 1 Gbps caused most HTTP connections to be lost, where a small recovery was experienced at 500 Mbps under a Class C botnet. The network connectivity of Windows Server 2016 under TCP SYN Flooding attacks is demonstrated in Figure 3.30.

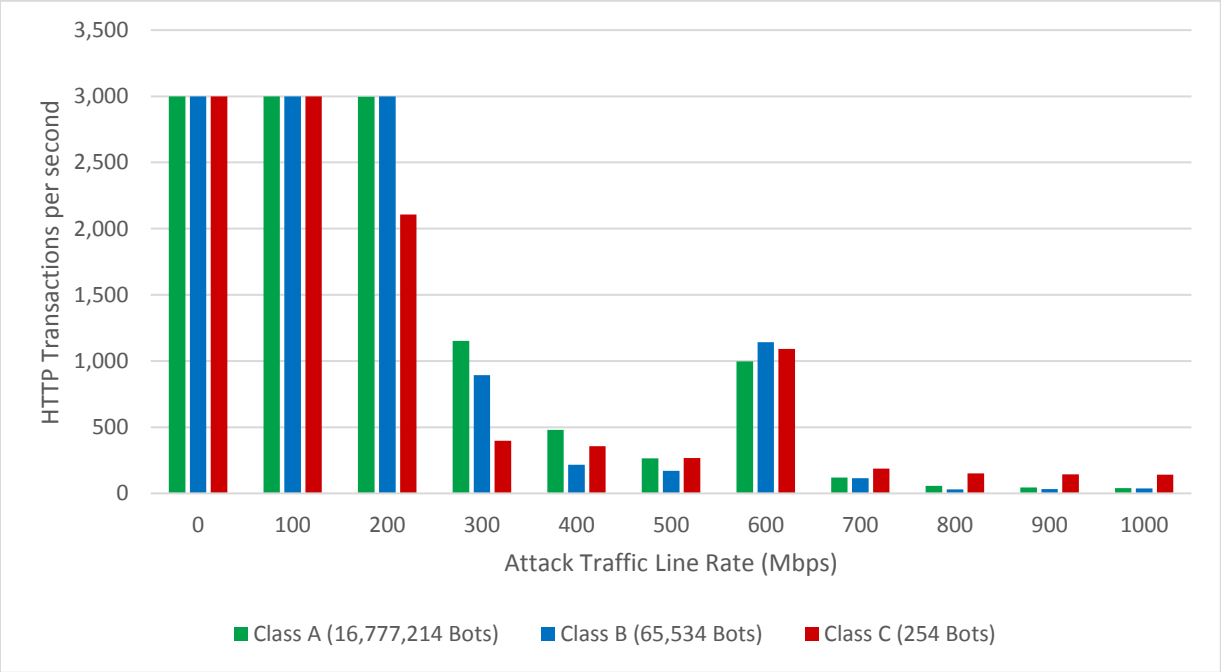


Figure 3.29 HTTP Transactions/sec under TCP SYN Flooding on Windows Server 2012 R2

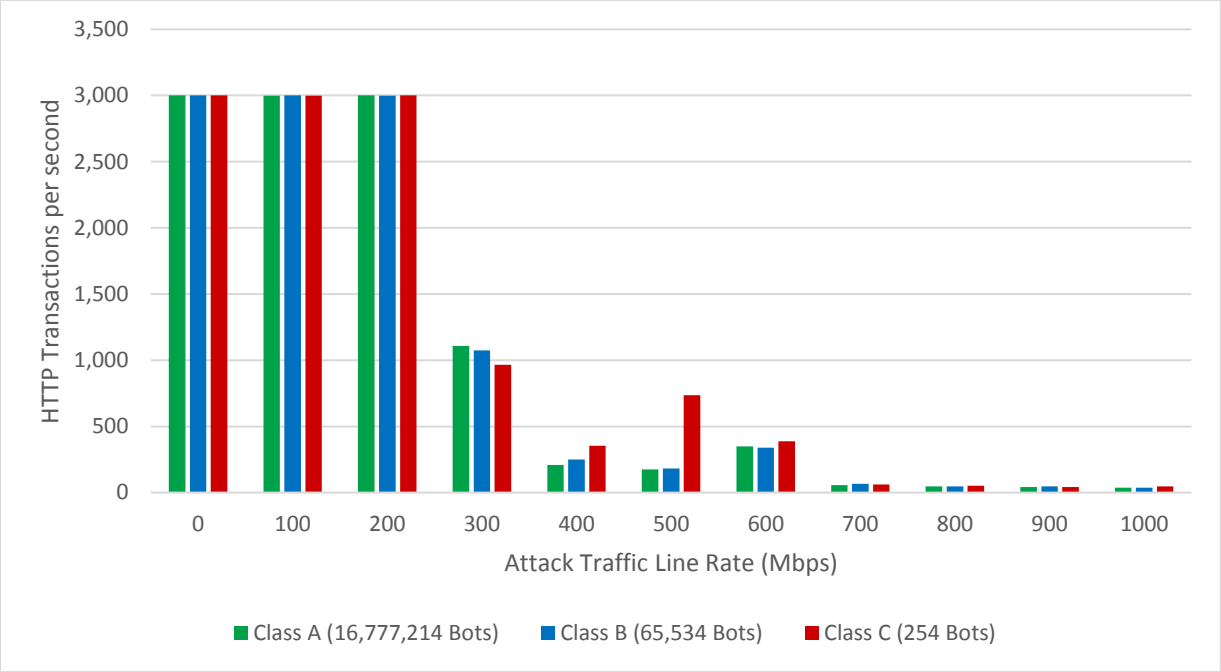


Figure 3.30 HTTP Transactions/sec under TCP SYN Flooding on Windows Server 2016

The amount of memory consumed during TCP SYN flooding attacks was relatively low. Similarly, to ICMP based attacks, data collected for TCP SYN flooding attacks would suggest this form of DDoS attack is not memory intensive either. The difference in total RAM consumed regardless of botnet size is of roughly 50 MB for Windows Server 2012 R2 and 120 MB for Windows Server 2016. The system characteristics of the server under attack feature 16 GB, at such a low memory consumption TCP SYN flooding attacks could be considered negligible for modern systems in terms of memory exhaustion. Windows Server 2012 R2 experienced lower memory consumption than Windows Server 2016 under baseline performance Windows Server 2012 R2 consumed memory in range of 577.48 MB to 585.29 MB while Window Server 2016 consumed nearly twice the amounts with a range of 911.53 MB to 948.65 MB. At low consumptions it is hard to evaluate one botnet size to be more detrimental than others. However, higher attack traffic loads intensified the rate at which RAM is utilized for both server operating

systems. The overall memory consumption of both operating systems under Smurf attacks can be seen in Figures 3.31, 3.32.

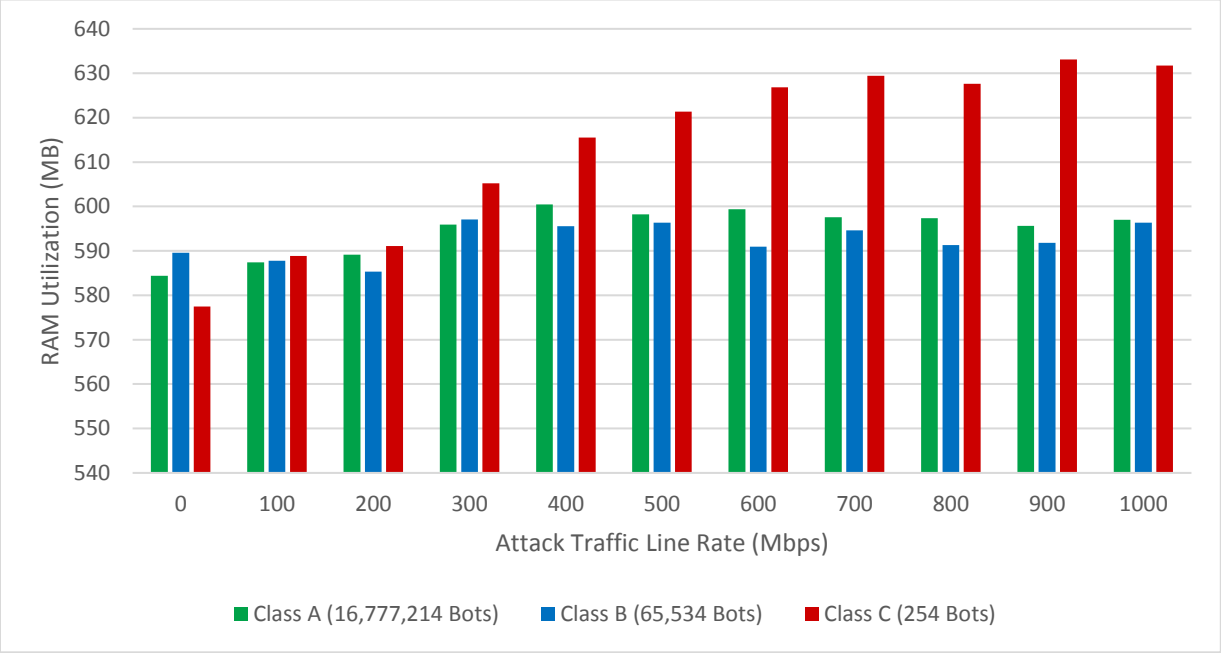


Figure 3.31 Total Memory Depletion under TCP SYN Flooding on Windows Server 2012 R2

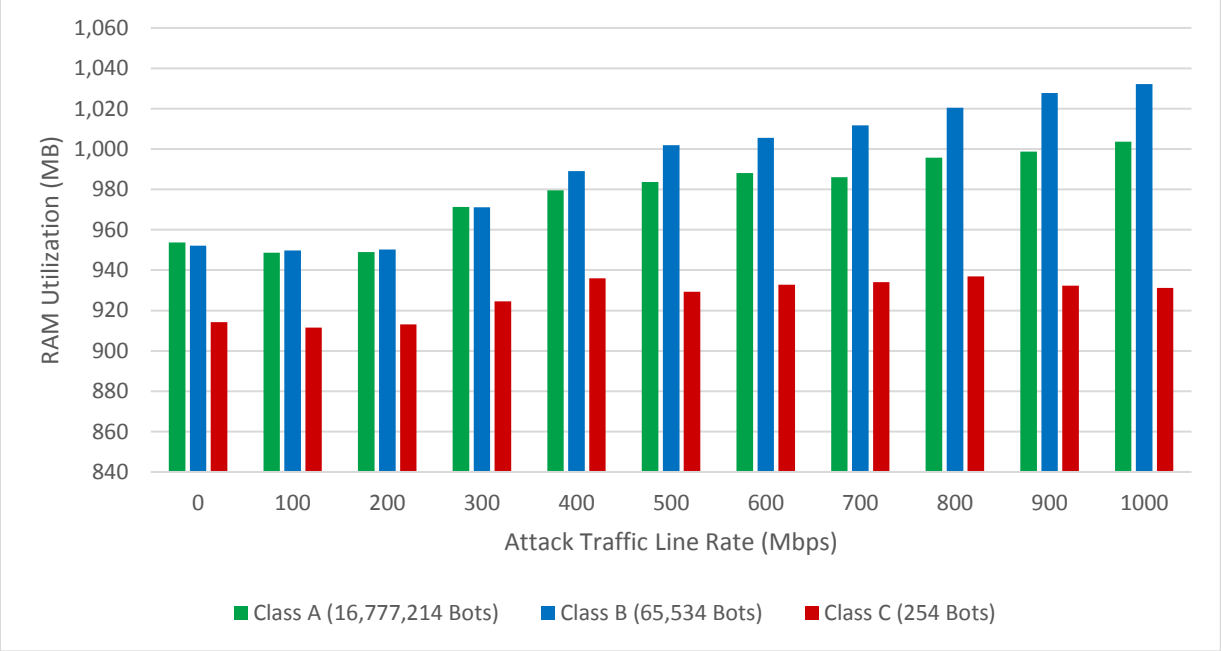


Figure 3.32 Total Memory Depletion under TCP SYN Flooding on Windows Server 2016

A simulated Class C botnet appears to trigger a higher amount of memory consumption than its larger counterparts on Windows Server 2012 R2, while simulated Class A and B botnets appear to consume more memory than simulated Class C botnets on Windows Server 2016. These trends are not a result of the introduction of attack traffic as this were preexisting conditions. It is unclear if background processes required a higher amount of memory to run at the time of the experimental trials as the difference in memory consumed is rather small. In Figures 3.33 and 3.34 the memory consumed during the baseline performance is subtracted from that of each attack load, this is done to accurate visualize the impact of the TCP SYN flooding attacks on memory and how botnet size and attack traffic load affect it.

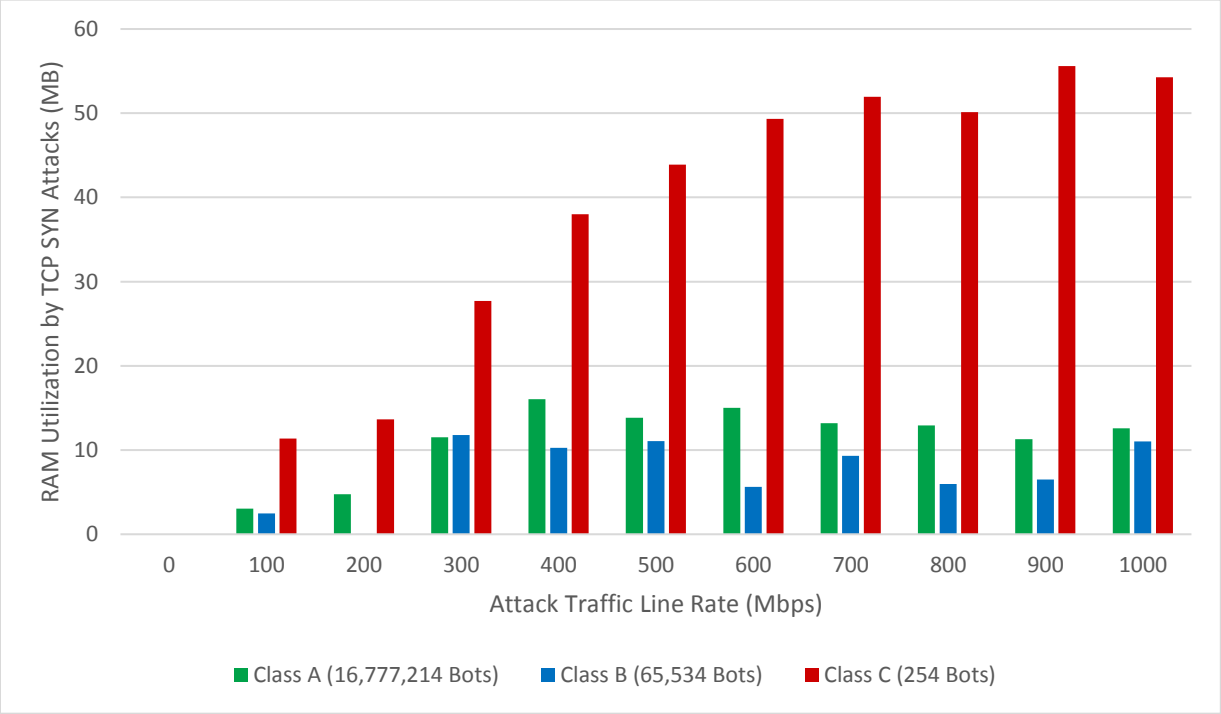


Figure 3.33 Memory Consumed on Windows Server 2012 R2 by TCP SYN Flooding

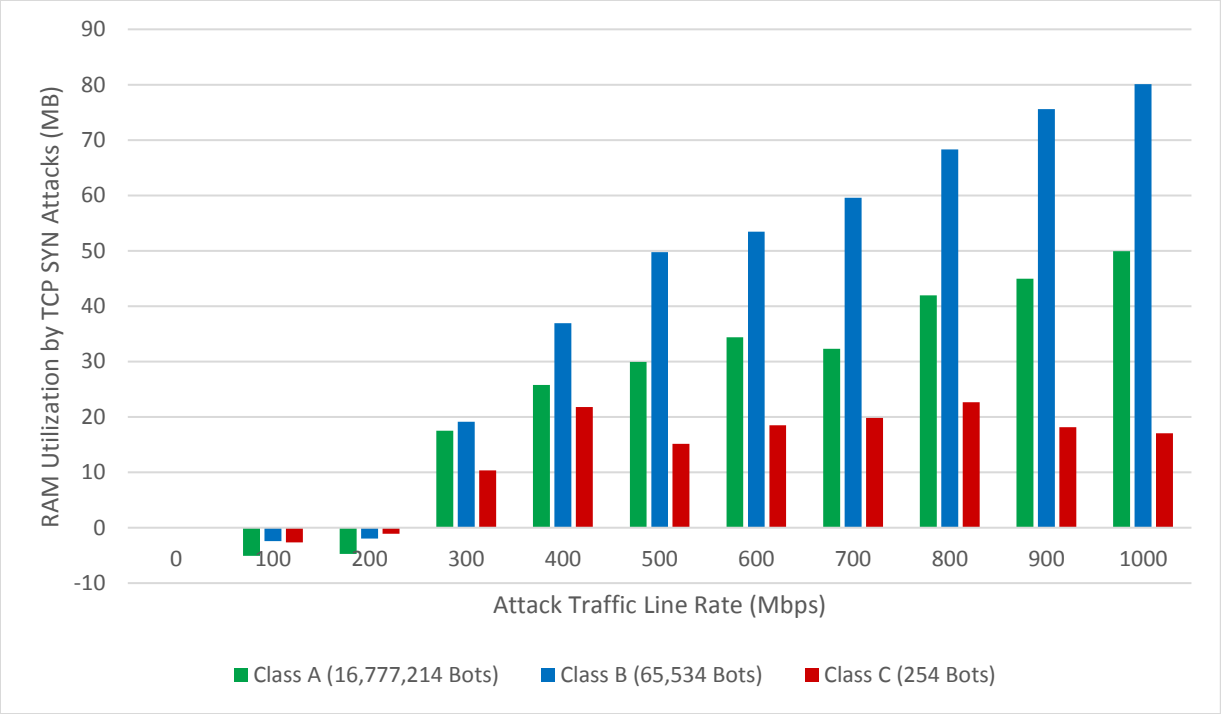


Figure 3.34 Memory Consumed on Windows Server 2016 by TCP SYN Flooding

Figures 3.33 and 3.34 do not depict a trend where botnet size is related to memory consumption however attack traffic load clearly presents a negative impact to memory consumption regardless of botnet size. Memory consumed by TCP SYN flooding attacks is minimal at values lower than 60 MB and 80 MB for Windows Server 2012 R2 Windows Server 2016 respectively. Class A, B and C botnets under Windows 2012 R2 showed low memory consumption with the highest amount of memory consumed by each botnet being 16.04 MB at 400Mbps, 11.79 MB at 300 Mbps and 55.61MB at 900 Mbps respectively. It is worth observing Class A, and B botnets under Windows Server 2012 after 300 Mbps maintain consumption between 16.06 MB and 5.64 MB of memory. Windows Server 2016 on the other hand exhibits a behavior which would imply that the botnet size did not affect it, however attack traffic raised the consumption of memory in for all botnets except Class C which fluctuated in the range of

-2.65 MB and 22.64MB. Meanwhile Class A and B botnets experienced a maximum memory consumption of 49.93 MB and 80.12 MB respectively.

In these experiments the botnet size had low to no impact in the effectiveness of TCP SYN Flooding attacks, while higher attack traffic loads, conversely, had a high impact in CPU exhaustion. All three botnet sizes had similar effectiveness on Windows Server 2012 R2. At baseline performance Windows Server 2012 R2 CPU utilization was kept at approximately 6.7%. The introduction of TCP SYN flooding traffic caused CPU consumption rise to approximately 21% at 100 Mbps, as the traffic load increased beyond 100 Mbps CPU utilization kept gradually increasing. CPU Utilization seemed to reach its limits at 600 Mbps where utilization was nearly 48% and was maintain with higher traffic loads up to 1 Gbps. No advantage was given to TCP SYN Flooding by botnet sizes. Total CPU utilization for Windows Server 2012 R2 is demonstrated in Figure 3.35

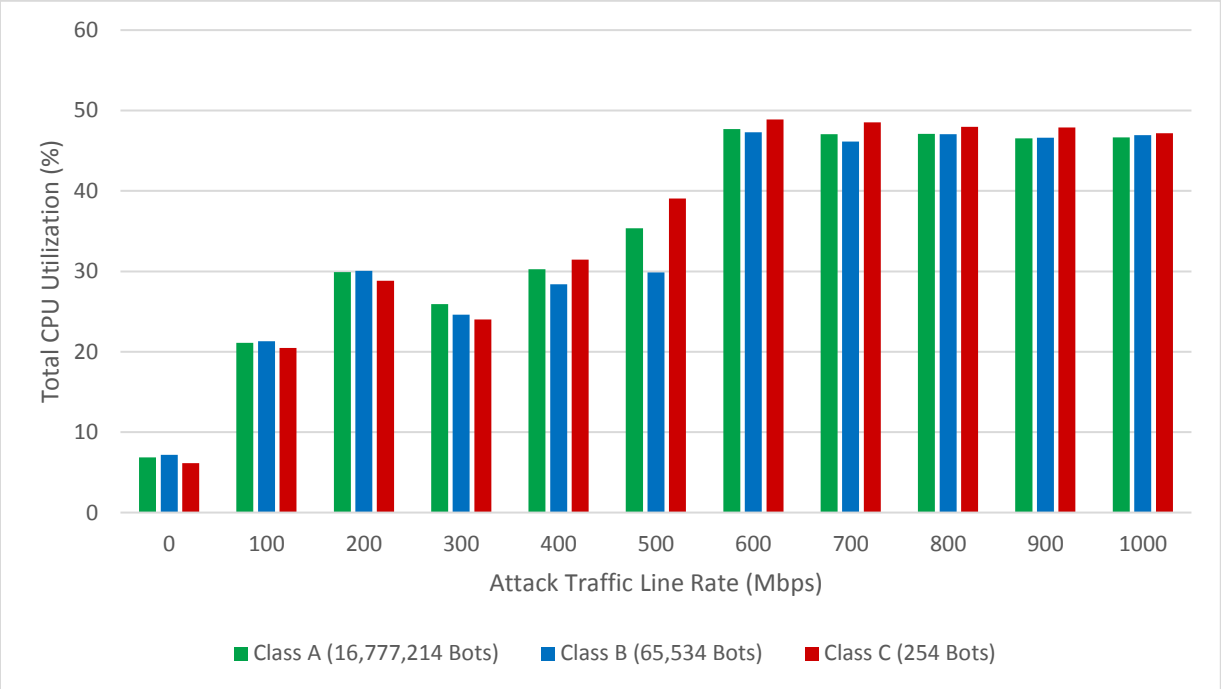


Figure 3.35 CPU Depletion under TCP SYN Flooding on Windows Server 2012 R2

On the other hand, CPU consumption on Windows Server 2016 had a CPU utilization of approximately 6.1%. The introduction of TCP SYN flooding traffic caused CPU consumption rise to approximately 20% at 100 Mbps, as the traffic load increased beyond 100 Mbps CPU utilization kept gradually increasing. CPU Utilization seemed to reach its limits at 600 Mbps where utilization was nearly 46% and was maintain with higher traffic loads up to 1 Gbps. As demonstrated by Figure 3.36 CPU power was not affected directly by the botnet size but rather the attack traffic load intensity on Windows Server 2016.

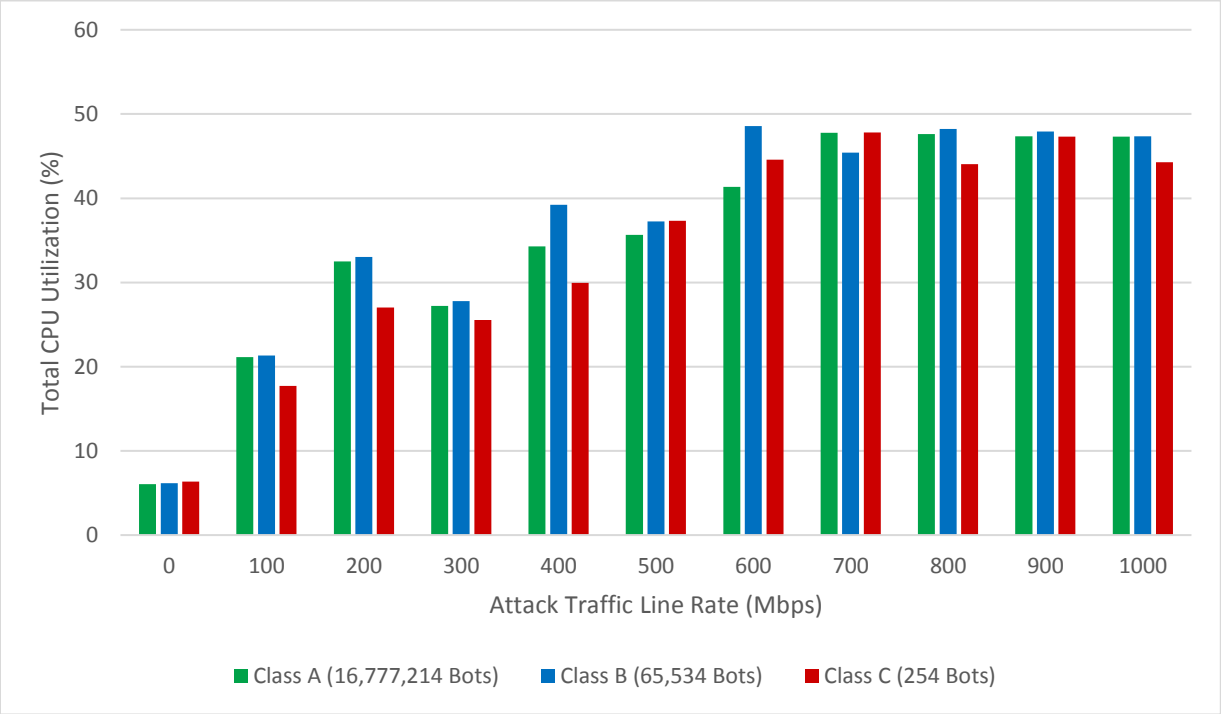


Figure 3.36 CPU Depletion under TCP SYN Flooding on Windows Server 2016

During TCP SYN flooding attacks it was observed that in both server operating systems only half of the virtual cores experienced any consumption of processing time as a result of the attack. All simulated botnets consumed a similar amount of processing power therefore the average core utilization for TCP SYN attacks is demonstrated thru Figures 3.37 and 3.38.

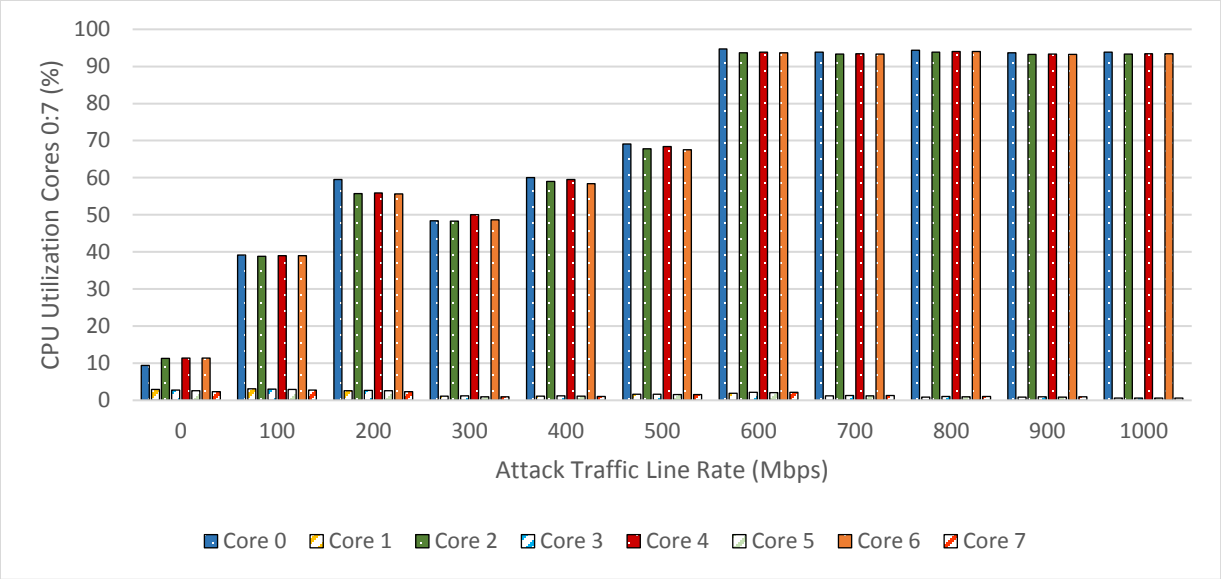


Figure 3.37 CPU Core Utilization under TCP SYN Flooding on Windows Server 2012 R2

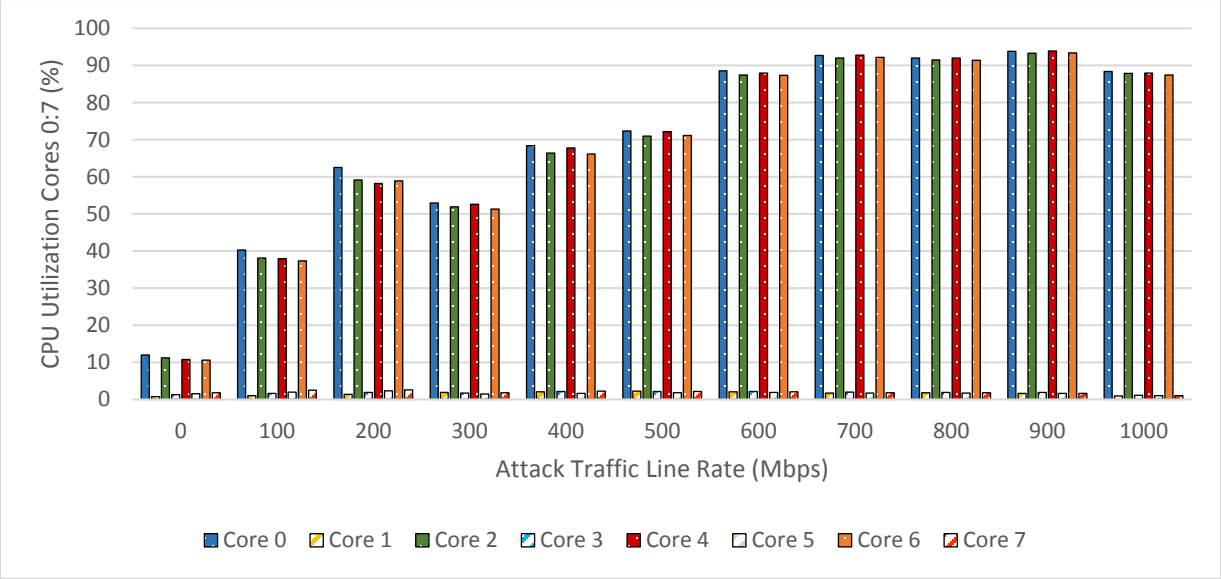


Figure 3.38 CPU Core Utilization under TCP SYN Flooding on Windows Server 2016

TCP SYN flooding attacks did not affect half of the virtual cores, this would imply that the network functions of Windows do not take advantage of Hyperthreading technology which helps mitigate the effect of these attack on the targeted system similarly, to the effects seen on ICMP based attacks. Lastly CPU temperatures on Windows Server 2012 R2 had a baseline of

47.58 °F and reached their highest point under attack traffic loads in the range of 700 Mbps and 1 Gbps reaching nearly 92.59 °C, while temperatures on Windows Server 2016 had a baseline of 51.41 °C and reached their highest point under attack traffic loads in the range of 700 Mbps and 1 Gbps reaching nearly 90.23 °C as well, which is still below TjMax but are not considered safe. The temperatures of the CPU reflect the amount of processing power consumed and can be seen in Figures 3.39 and 3.40.

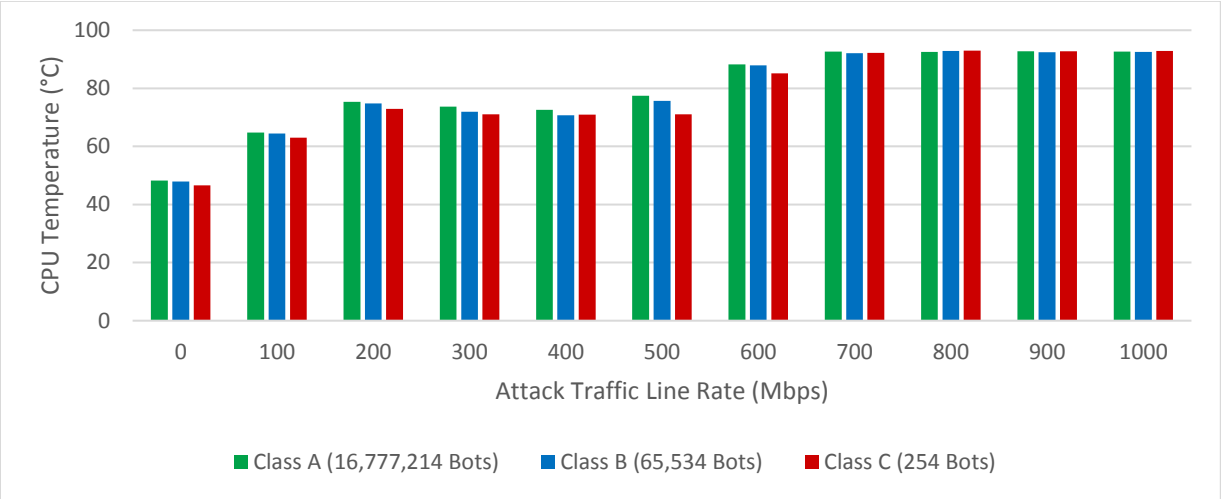


Figure 3.39 CPU Temperature under TCP SYN Flooding on Windows Server 2012R2

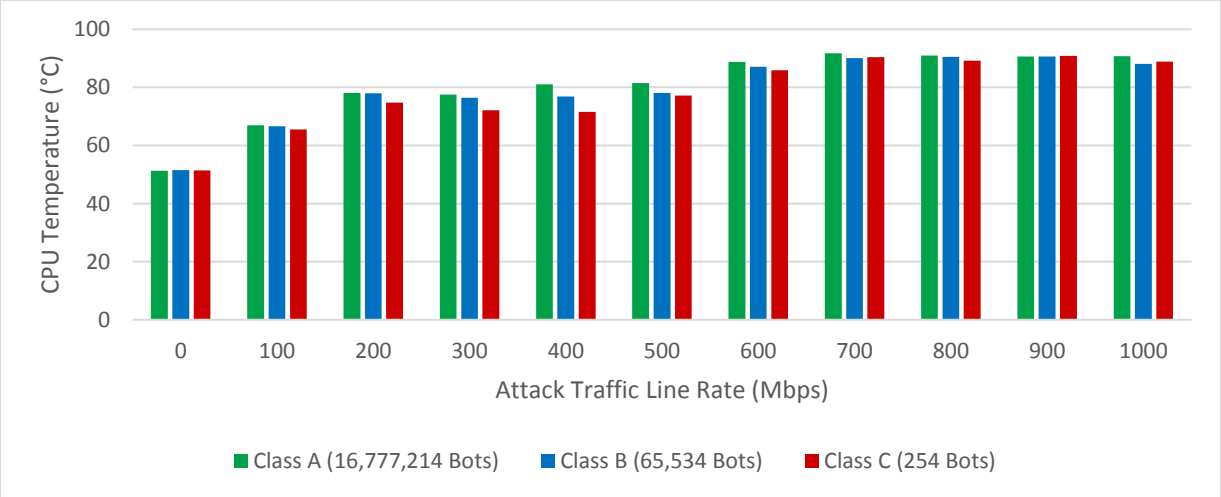


Figure 3.40 CPU Temperature under TCP SYN Flooding on Windows Server 2016

3.3.4 Layer 4 UDP Flood Attack

Botnets sizes had no impact on the number of UDP Datagrams received by both server operating systems under UDP Flooding Attacks. As demonstrated in Figure 3.41 regardless of the simulated botnet network employed the average amount of UDP Datagrams received per second was kept within the same range for both server operating systems where only the line rate of attack traffic would impact this number. UDP datagrams in this attack were targeted at a closed port. Per RFC 1122 in the event a system receives a UDP datagram at a port that is not being listened to by an application will trigger a response in the form of an ICMP packet stating the destination was unreachable. In experiments for both operating systems not a single ICMP Destination Unreachable message was transmitted from the targeted server. By not transmitting these messages the system saves processing power and bandwidth.

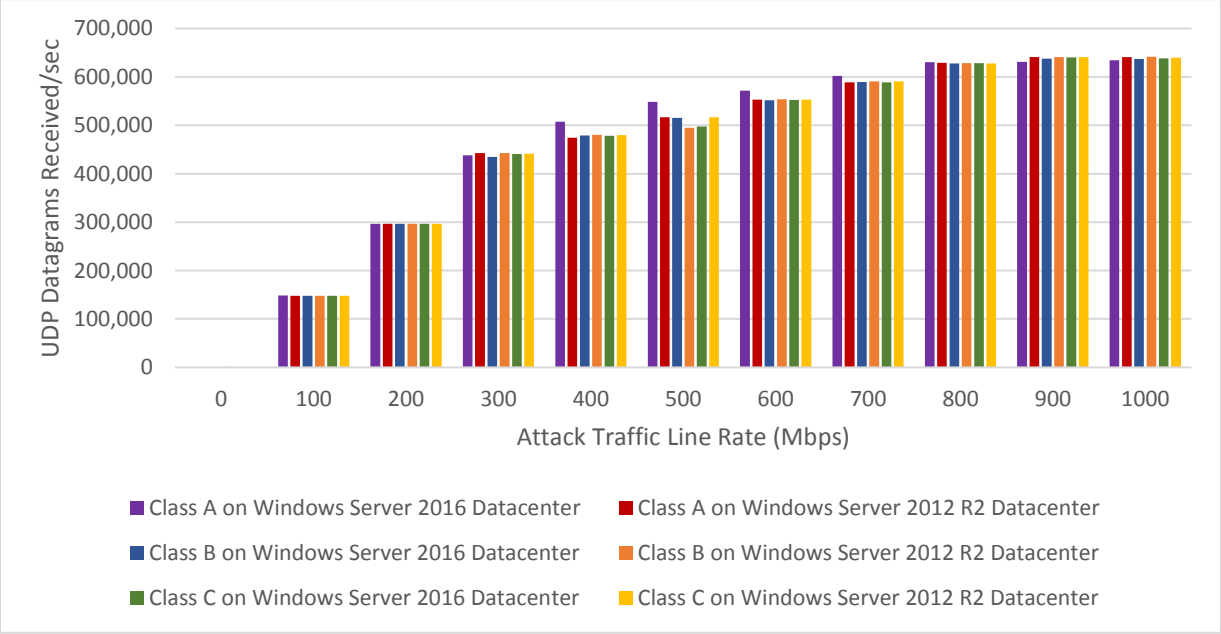


Figure 3.41 UDP Datagrams Received/sec under UDP Flooding on Windows Servers

In these experiments the number of HTTP transactions per second where found to be better maintained when a smaller botnet was employed against Windows Server 2012 R2, but botnet sizes had a lower impact on Windows Server 2016 where the number of HTTP connections was found to fluctuate beyond 200 Mbps of attack traffic regardless of botnet size. As UDP datagrams began to flood Windows Server 2012 R2 the baseline performance of 3000 HTTP transactions/sec was maintained up to 200 Mbps of attack traffic while under the effects of three simulated botnets. However, the number of HTTP transactions/sec started declining at 300 Mbps where a Class A botnet lowered the amount of connections by 40.84% while Class B lower the amount by 23.04% and Class C lowered the amount by 31.03%. Attack traffic at 400 Mbps diminished the number of HTTP Transaction/sec by 84.47% for Class A botnets, 81.09% for Class B botnets, and 75.69% for Class C botnets. Higher attack traffic loads triggered a gradual decline in the remaining number of HTTP connections until at 1 Gbps of attack traffic which caused most connections to be lost. The network connectivity of Windows Server 2012 R2 under UDP flooding attacks is demonstrated in Figure 3.42.

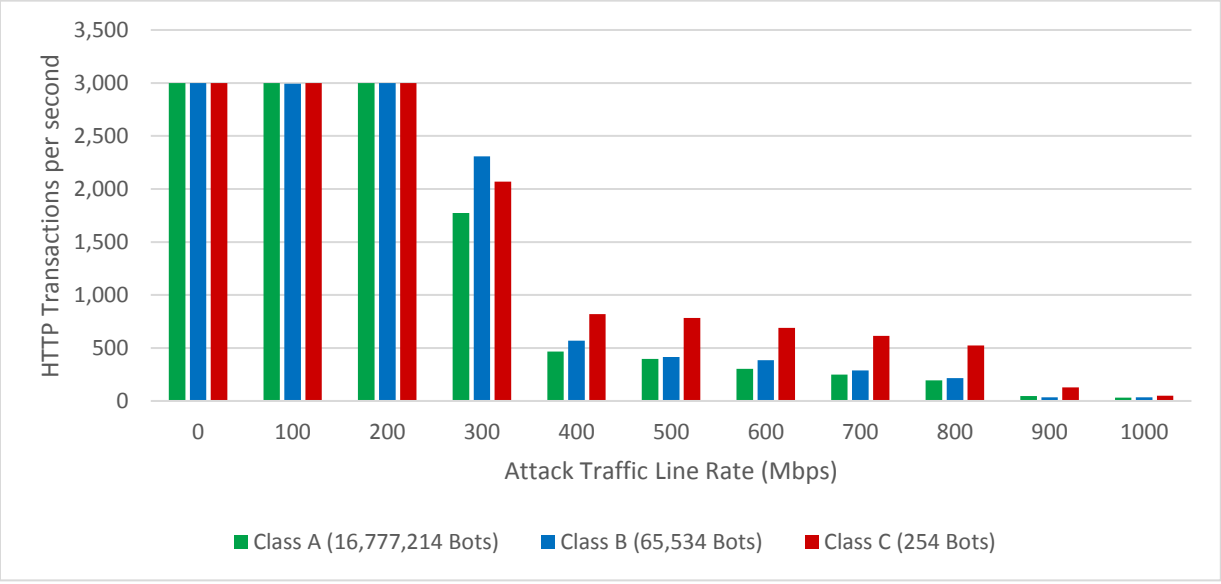


Figure 3.42 HTTP Transactions/sec under UDP Flooding on Windows Server 2012 R2

Similarly, to Windows Server 2012 R2 as UDP datagrams began to flood Windows Server 2016 the baseline performance of 3000 HTTP transactions/sec was maintained up to 200 Mbps of attack traffic while under the effects of three simulated botnets. However, the number of HTTP transactions/sec started declining at 300 Mbps where a Class A botnet lowered the amount of connections by 76.01% while Class B lower the amount by 50.32% and Class C lowered the amount by 55.46%. At an attack traffic of 400 Mbps and higher the number of HTTP transactions lost was mostly equalized across all three botnets. On average the number of HTTP diminished by all three botnets was diminished by %. Higher attack traffic loads triggered a gradual decline in the remaining number of HTTP connections until at 1 Gbps of attack traffic which caused most connections to be lost. The network connectivity of Windows Server 2016 under UDP flooding attacks is demonstrated in Figure 3.43.

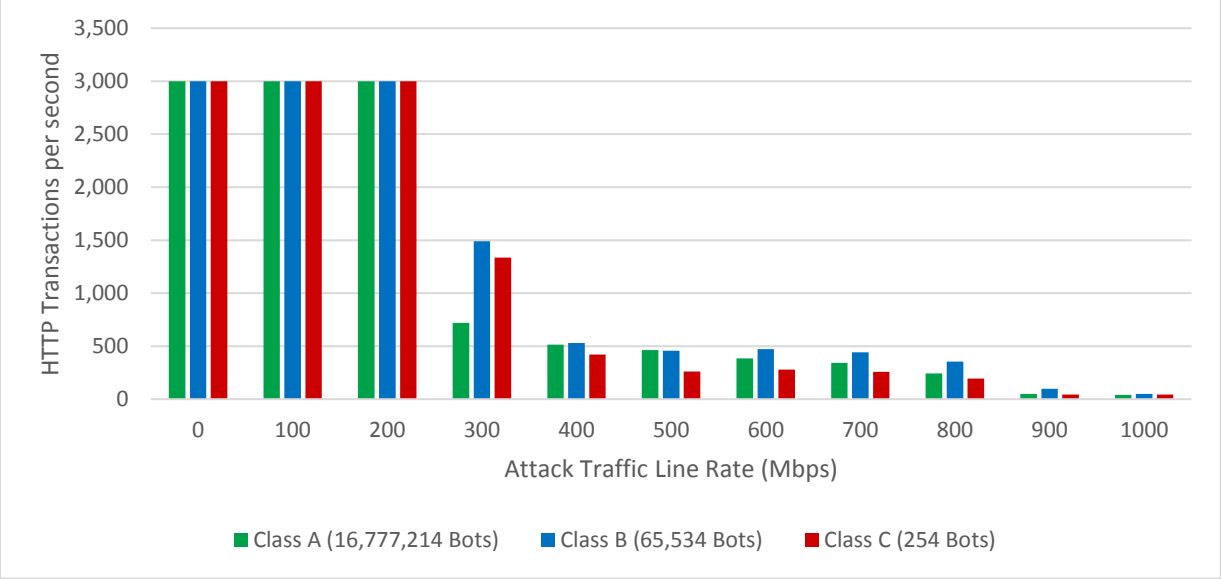


Figure 3.43 HTTP Transactions/sec under UDP Flooding on Windows Server 2016

The amount of memory consumed during by UDP Flooding attacks was relatively low. Similarly, to ICMP based attacks, and TCP SYN Flooding attacks data collected for UDP

flooding attacks would suggest the attacks performed in this Chapter are not memory intensive. The difference in total RAM consumed regardless of botnet size was found to be 168.49 MB for Windows Server 2012 R2 and 111.12 MB for Windows Server 2016. The system characteristics of the server under attack feature 16 GB, at such a low memory consumption UDP flooding attacks can be considered negligible for modern systems in terms of memory. Windows Server 2012 R2 experienced lower memory consumption than Windows Server 2016 under baseline performance Windows Server 2012 R2 consumed memory in range of 551.12 MB to 687.92 MB while Window Server 2016 consumed nearly twice the amounts with a range of 945.84 MB to 1025.98 MB. While botnet sizes did not have a clear impact on memory consumption, higher attack traffic loads intensified the rate at which RAM is utilized for both server operating systems. The overall memory consumption of both operating systems under UDP flooding attacks can be seen in Figures 3.44, and 3.45.

Figure 3.44 Total Memory Consumption under UDP Flooding on Windows Server 2012 R2

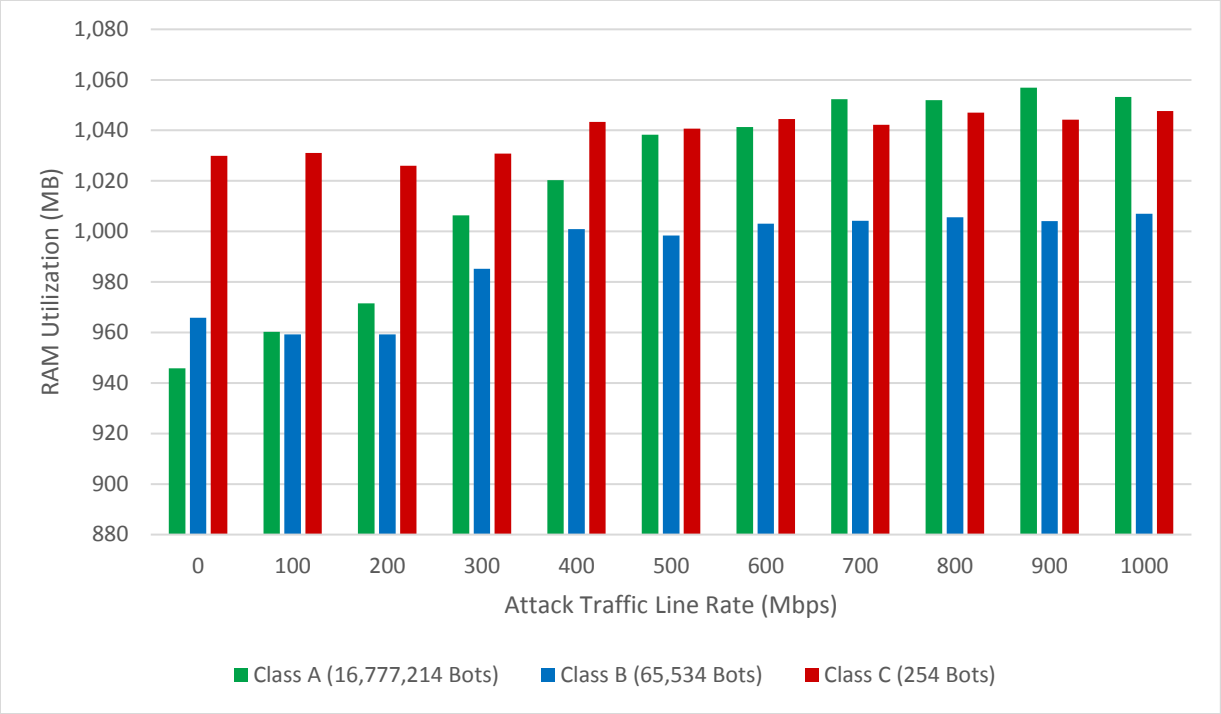


Figure 3.45 Total Memory Consumption under UDP Flooding on Windows Server 2016

A simulated Class C botnet appears to trigger a higher amount of memory consumption than its larger counterparts on Windows Server 2012 R2, while Class A and C simulated botnets appear to consume more memory than Class B on Windows Server 2016. These trends are not a result of the introduction of attack traffic as this were preexisting conditions. It is unclear if background processes required a higher amount of memory to run at the time of the experimental trials as the difference in memory consumed is rather small. In Figures 3.46 and 3.47 the memory consumed during the baseline performance is subtracted from that of each attack load, this is done to accurate visualize the impact of the UDP flooding attacks on memory and how botnet size and attack traffic load affect it.

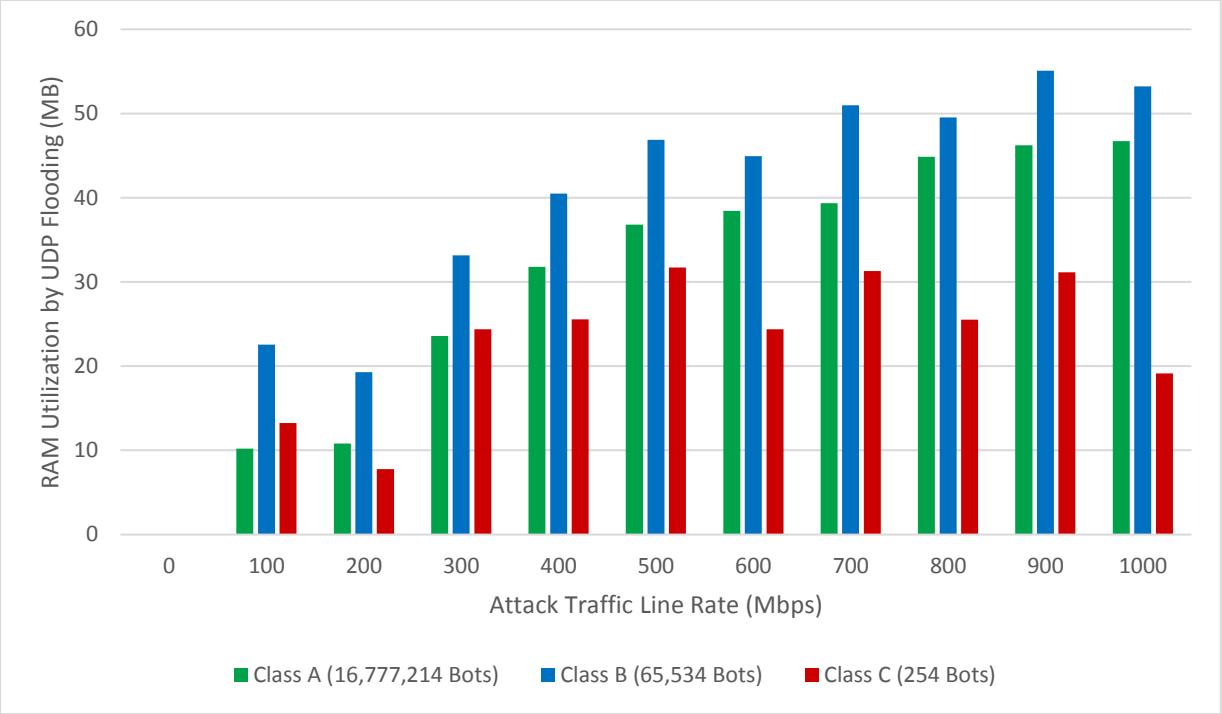


Figure 3.46 Memory Consumed on Windows Server 2012 R2 by UDP Flooding

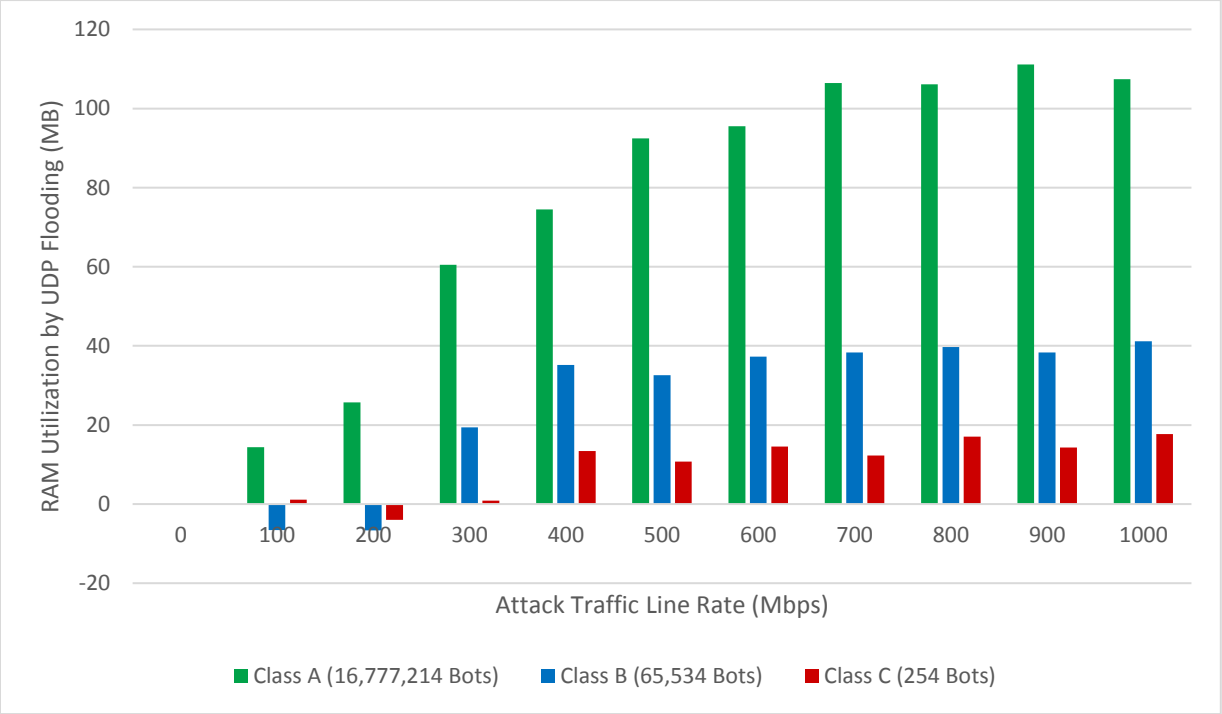


Figure 3.47 Memory Consumed on Windows Server 2016 by UDP Flooding

It was observed on Figure 3.46 that for Windows Server 2012 R2 the size of the botnet did not have an impact on amount of memory consumed by the attack while the rate at which attack traffic reaches the server does present a negative impact to memory consumption. The minimum amount of memory consumed by UDP flooding was 7.77MB while the highest was 55.08MB. Furthermore, on Figure 3.47 it is demonstrated that Windows Server 2016 was affected by both botnet size and attack traffic line rate intensity. Occasionally, at lower loads memory consumed by UDP flooding was lower than that of baseline by 6.5MB while the highest was 111.12 MB. Memory consumed by UDP flooding attacks for both server operating systems was minimal when compared with the amount of memory available (16 GB).

In the terms of CPU power consumed, the effectiveness of UDP flooding was not increased by botnet size, conversely higher attack traffic loads triggered higher a consumption of CPU power. Earlier in this subsection it was demonstrated that the number of UDP datagrams received was the same for all botnet sizes. Similar CPU utilization was experienced regardless of botnet size since during all trials the server operating system had to process the same load. CPU utilization data for both operating systems found Windows Server 2012 R2 to consume slightly less CPU power than Windows Server 2016 at 200Mbps. Overall both server operating systems had the same performance at other loads regardless of botnet size. At baseline performance CPU utilization was kept at approximately 6.61%. The introduction of UDP flooding traffic saw CPU consumption rise to nearly 19.93% at 100 Mbps and was at its highest at 200Mbps where 26.02% CPU power was consumed on average. Subsequently an attack traffic load of 300 Mbps found CPU power consumption to decrease to 17.82%. Traffic loads beyond 400 Mbps triggered CPU consumption to rise gradually and stabilized in the range of 600 Mbps to 1 Gbps with an

average CPU consumption of 23.68%. Total CPU utilization is demonstrated in Figures 3.48 and 3.49.

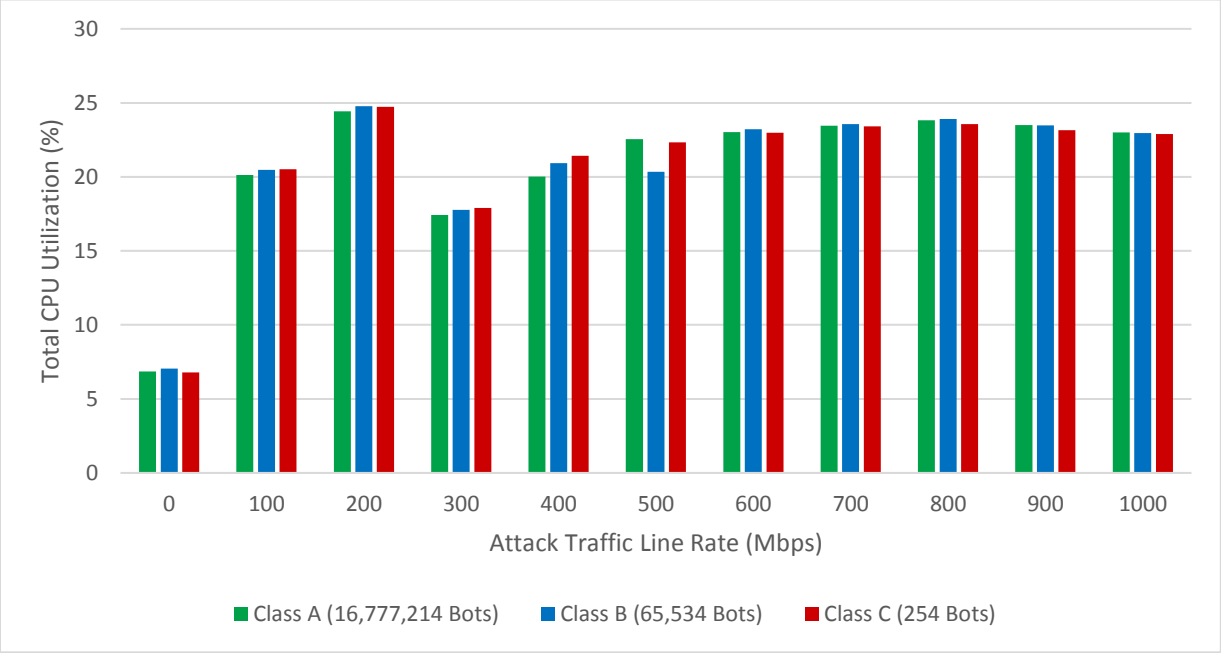


Figure 3.48 CPU Depletion under UDP Flooding on Windows Server 2012 R2

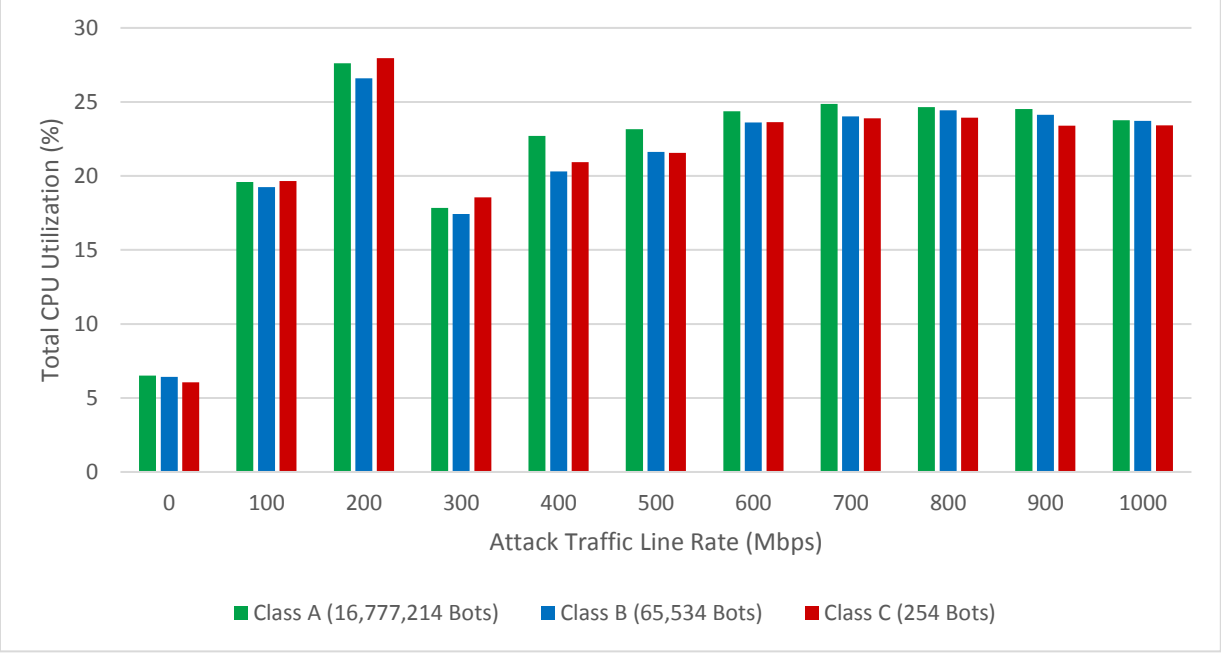


Figure 3.49 CPU Depletion under UDP Flooding on Windows Server 2016

UDP flooding attack only consumed processing power in half of the virtual cores. All simulated botnets consumed a similar amount of processing power therefore the average core utilization for UDP flooding attacks is demonstrated thru Figures 3.50 and 3.51.

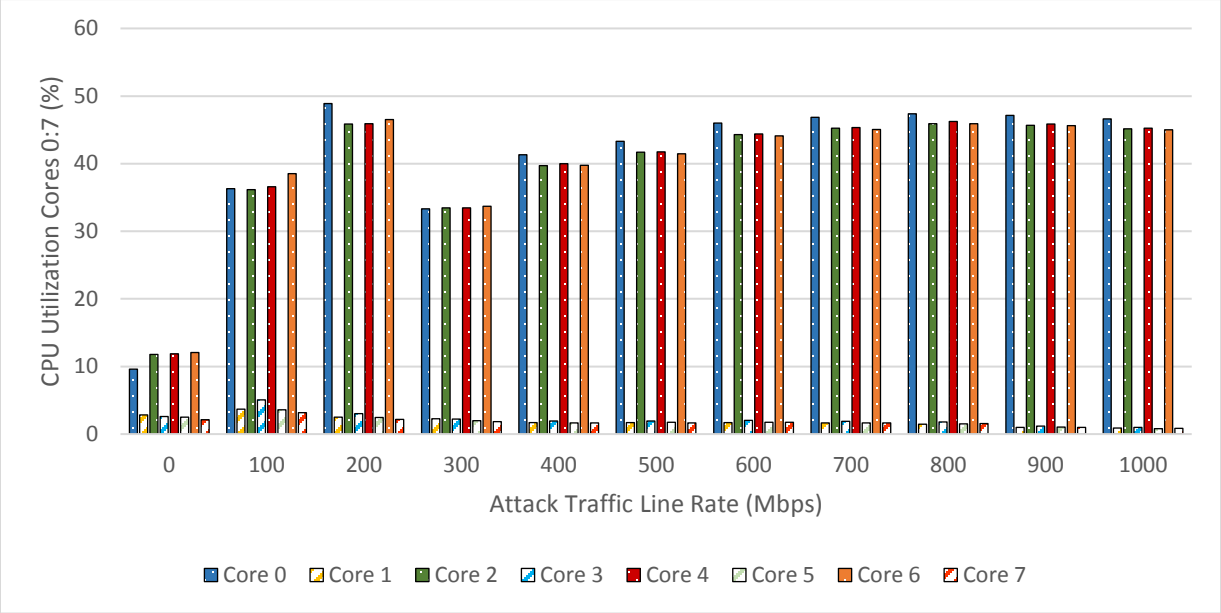


Figure 3.50 CPU Core Utilization under UDP Flooding on Windows Server 2012 R2

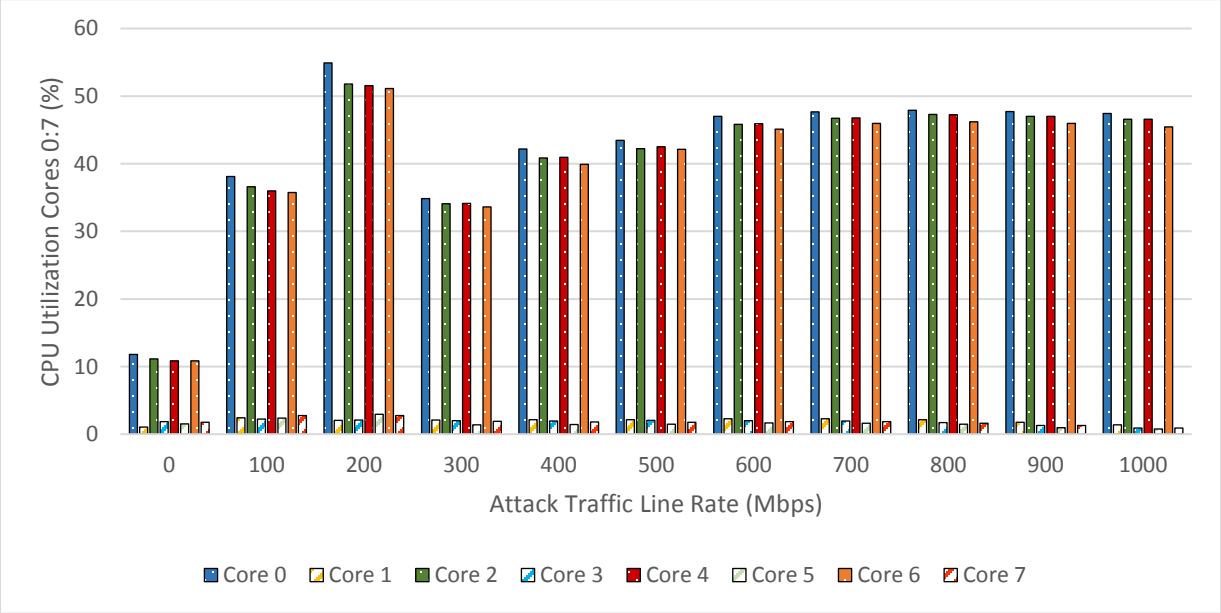


Figure 3.51 CPU Core Utilization under UDP Flooding on Windows Server 2016

Similarly, to ICMP and TCP based attacks performed in this chapter, UDP flooding attacks did not affect half of the virtual cores, this would imply that the network functions of Windows do not take advantage of Hyperthreading technology which helps mitigate the effect of these attack on the targeted system. Lastly CPU temperatures on Windows Server 2012 R2 reached their highest point under 200 Mbps of attack traffic load reaching nearly 73.04 °C on average while higher attack traffic loads yielded temperatures averaging at 66.51 °C. Moreover, temperatures on Windows Server 2016 reached their highest point under 200 Mbps of attack traffic load reaching nearly 75.91 °C on average while higher attack traffic loads yielded temperatures averaging at 68.59 °C. These temperatures are below TjMax. The temperatures of the CPU reflect the amount of processing power consumed and can be seen in Figures 3.52 and 3.53.

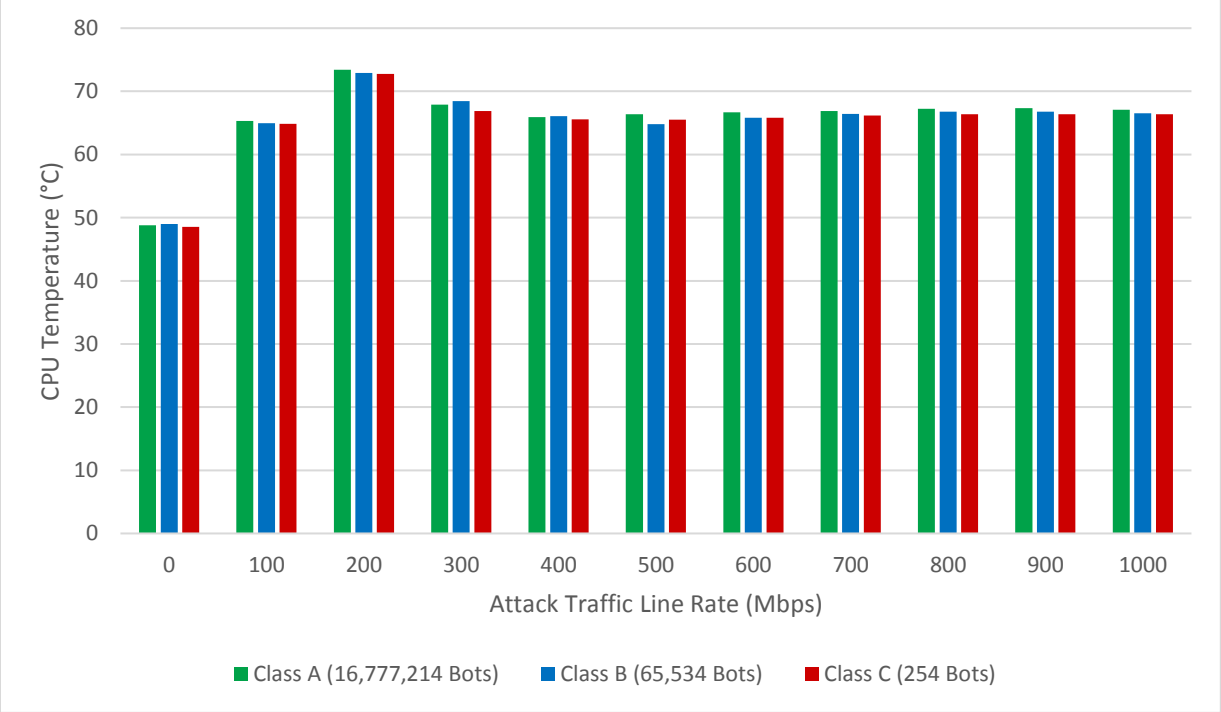


Figure 3.52 CPU Temperature under UDP Flooding on Windows Server 2012 R2

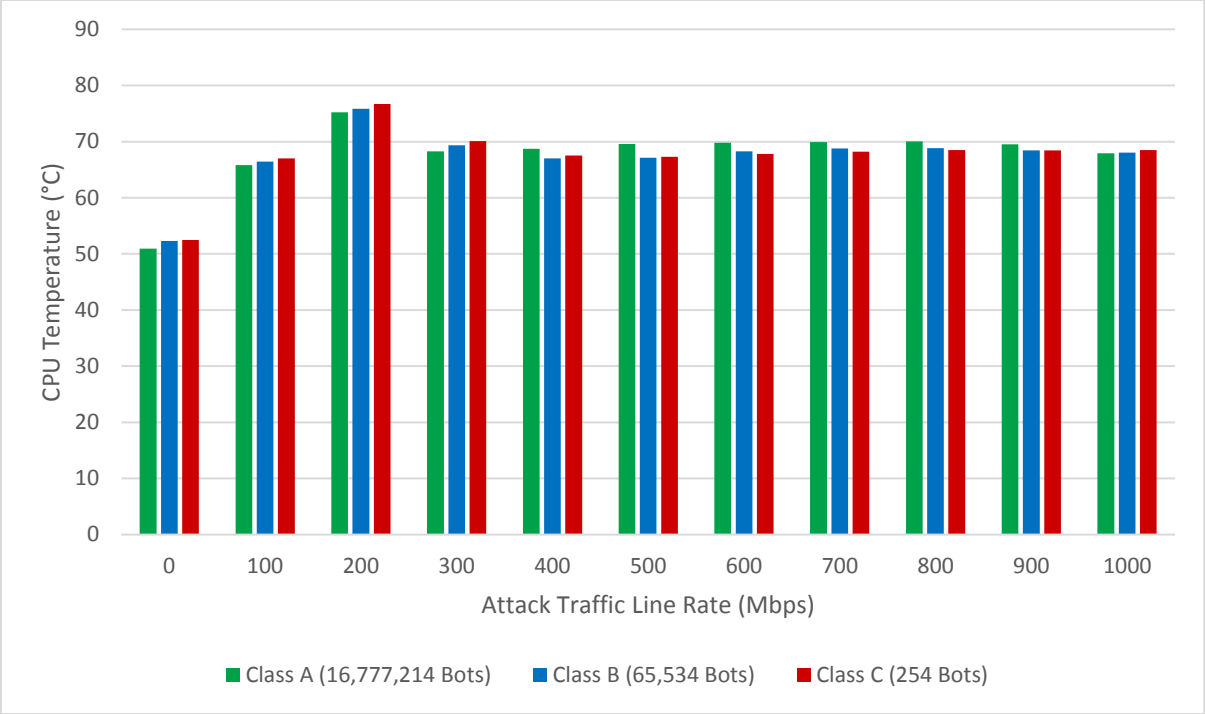


Figure 3.53 CPU Temperature under UDP Flooding on Windows Server 2016

3.4 Chapter Summary

In this Chapter, Microsoft's Windows Server 2016 and Windows Server 2012 R2 were deployed on Apple's MacPro mid(2010). Both server operating systems were subject to three different botnet sizes replicating compromised Class A, B, and C networks which carried out four distinct Distributed Denial of Service attack namely Ping attack, Smurf attack, TCP SYN flooding, and UDP Flooding.

The impact of these attacks was measured utilizing Microsoft's performance monitor and CoreTemp [96-97]. Moreover, Microsoft Excel and MATLAB were used to interpret the data acquired, by translating raw data from .csv files into graphs [98-99]. Graphs were generated with these tools allow for the presentation of data in a clean, concise manner which helps readers visualize and compare the performance achieved by both operating systems under attack.

Under Ping attacks it was observed that both server operating systems received similar amounts of echo request messages regardless of botnet size however no echo replies were generated by either operating system. In terms of network connectivity Windows Server 2012 R2 performed similarly to Windows Server 2016, however Windows Server 2012 R2 had higher connectivity than its successor under Class C botnets. Memory utilization under these attacks is negligible at a low amount of memory consumption often lower than 100 MB. Additionally, Ping attacks presented the lowest amount of CPU consumption out of all four attacks performed where Ping attack consumed at most 18% CPU power which yielded a CPU temperature of 70 °C.

Smurf attacks on the other hand consumed the highest amount of CPU power at most traffic loads, with a maximum CPU utilization of 49.73% on average, which yield CPU temperatures in the 90 °C range. Memory consumption in these attacks was negligible with low memory consumption having a maximum of nearly 100 MB consumed. The botnet size had little effect to no effect on the parameters just mention, which is attributed to the number of echo replies received which was similar regardless of botnet size. However, botnet size did appear to have an impact on HTTP traffic for Windows Server 2016 where HTTP traffic had a higher survivability under a Smurf attack performed using a Class C botnet. Other botnet sizes saw traffic rapidly decline at 300 Mbps. Windows Server 2012 R2 did not perform as favorably as its successor seeing as HTTP traffic was most dropped at 300 Mbps with all three botnet sizes.

The second most devastating attack found in these experiments was TCP SYN flooding. Under these attacks HTTP transactions rapidly declined at 300 Mbps where botnet size affected Windows Server 2012 R2 but not its predecessor. Memory consumed is also negligible for this attack with low memory consumption lower than 90 MB. CPU consumption was relatively

similar for all botnet sizes where Class A and B would on occasion consume 1% to 2% more CPU power than their smaller counterpart. Max CPU consumption for both server operating systems was met at 600 Mbps where roughly 47% of the CPU was utilized by both server operating systems raising the temperature to a threatening 91 °C.

It was observed that UDP Flooding was not benefitted by a larger botnet size. The number of received UDP datagrams was similar across all three botnet sizes which cause CPU utilization to be kept similar across all experiments additionally no ICMP destination unreachable messages were generated under the effects of this type of attack by either operating system. Max CPU consumption was met at 200 Mbps where Windows Server 2012 R2 saw 25% CPU utilization while its successor experienced a CPU utilization of 27%. Memory consumption in these attacks was negligible with low memory consumption having a maximum of nearly 111 MB consumed. In these attacks Windows Server 2012 R2 was able to maintain HTTP connection slightly better than Windows Server 2016, where both experienced a decline at 300 Mbps, and a drastic drop at 400 Mbps which was exacerbated as traffic load increased.

Hyperthreading technology aided in the mitigation of all attacks performed. From the data observed none of the odd numbered virtual cores were ever exhausted by any attack load, botnet size or type of DDoS attack. This suggest that Microsoft's Windows Server operating systems by default does not take advantage of hyperthreading technology to handle network traffic. Therefore, the utilization of the CPU in this scenario will always be nearly half of what other systems lacking hyperthreading technology will have.

CHAPTER IV

SECURITY ASSESMENT OF MACOS HIGH SIERRA UNDER THE EFFECTS OF DISTRIBUTED DENIAL OF SERVICE ATTACKS FROM BOTNETS

This chapter will extend the experiments performed on Microsoft's Windows Server platforms to a UNIX based platform therefore the security features of Apple's macOS 10.13.6 High Sierra are assessed thru an evaluation where the performance of the operating system is compromised thru DDoS attacks executed by simulated botnets replicating compromised Class A, B, and C IP networks. Similar works have been performed by participants at the Network Research Lab (NRL) of the University of Texas Rio Grande Valley [24-36] nonetheless, this study will introduce a newer operating system, and the effects of larger simulated botnet sizes.

Released on September 25, 2017 Apple's macOS 10.13 codename High Sierra is the fourteenth release of Apple's macOS, it's preceded macOS 10.12 codename Sierra released on September 20, 2016 and is succeeded by macOS 10.14 codename Mojave which became available on September 24, 2018 [104]. macOS High Sierra's most recent security update at the time this thesis was written is 10.13.6 (17G6030) was released on March 29, 2019 [105]. In Apple's macOS information website, it is claimed that this Operating System can performed tasks other operating systems are not capable off and its fundamentals are built around privacy and security. This chapter focus on the verification of this claim in terms of DDoS protection [106].

This study is performed to determine the security capabilities achieved by Apple in terms of performance and survivability against common DDoS attacks. No server exclusive versions

are distributed by Apple environments anymore, instead an application available thru Apple's Appstore handles server administrative task apart from web server services [107]. macOS 10.13.6 High Sierra was deployed on Apple's Mac Pro Server (Mac Pro Mid2010). Applying identical hardware characteristics for all operating systems in this thesis will provide clarity on the effects of these experiments as difference in hardware will not have a role on system performance when they are compared in Chapter V. Furthermore, the version deployed of macOS 10.13.6 High Sierra in this study featured all security updates released up to December 2018.

Section 4.1 of this chapter introduces the experimental setup, hardware and software specifications used under the experimental trials for these attacks, and the methodology thru which they are carried out. Parameters of performance evaluation are defined in section 4.2 within this chapter. These parameters will be used to evaluate the performance provided by macOS 10.13.6 High Sierra. Section 4.3 will demonstrate and discuss the data obtain for each parameter of evaluation described. This will permit the understanding of the effects created by these attacks within this environment.

4.1 Experimental Setup

An operating system evaluation for Apple's macOS 10.13.6 High Sierra was conducted within a controlled environment established at the Network Research Lab of the University of Texas Rio Grande Valley. The security features and performance of the operating systems is assessed against common IP layer ICMP, TCP, and UDP based attack traffic originating from simulated botnets emulating compromised Class A, B, and C IP networks. A clean installation of the operating system was deployed on Apple's Mac Pro Server (Mac Pro Mid2010) to properly evaluate the security features offered by the operating system. Additionally, operating system

updates were installed up to releases from December 2018, and firewall settings were left on by default.

For these experiments attack traffic was transmitted as simulated legitimate traffic attempted to reach the web server within the platform. Apple has a built in Apache webserver however it does not count with a built-in software service to aid in launching and maintain a local website therefore a free software tool MAMP was installed to run the Apache web server [108]. Apache web server hosted a simple Hypertext Markup Language (HTML) website named Index.html and is demonstrated on Figure 4.1

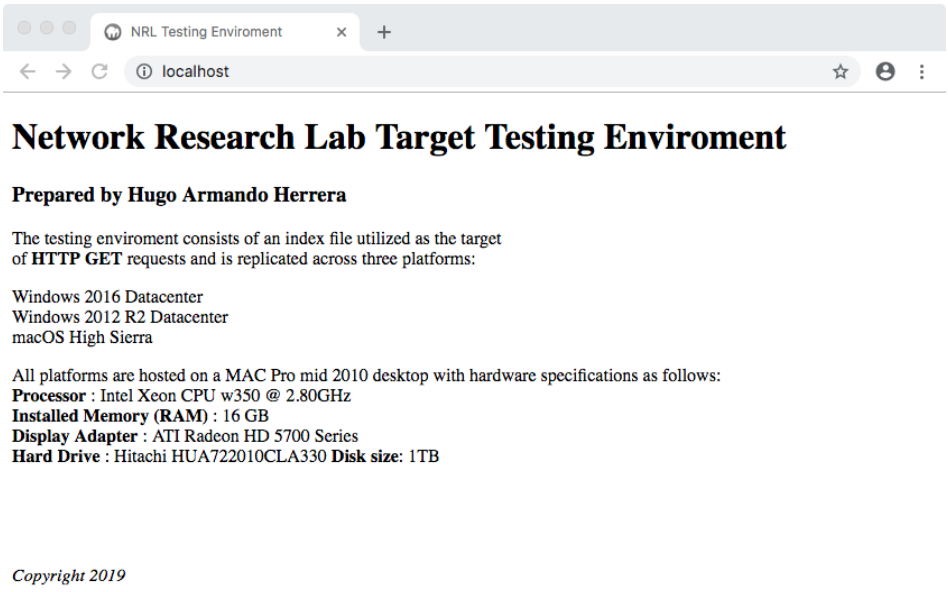


Figure 4.1 Targeted Website on macOS High Sierra

The availability of this website was determined by the amount of consistent Hypertext Transfer Protocol (HTTP) Transaction per second achieved by the legitimate clients. HTTP defined by RFC 7230 is a stateless request/response application layer protocol that operates by exchanging request and response messages thru a reliable connection [91]. For these experiments HTTP GET messages will be used to request the Index.html website file from the webserver at a

rate of 600 HTTP Transaction per second for 55 minutes. A higher rate of HTTP transactions in this platform resulted in occasional to several HTTP request failures. A low amount of HTTP request failures was experienced at 700 HTTP Transactions per second, higher numbers of transactions per second exacerbate the amount of HTTP requests failures. Therefore 600 transactions/sec was chosen as the baseline. This is demonstrated on Figure 4.2

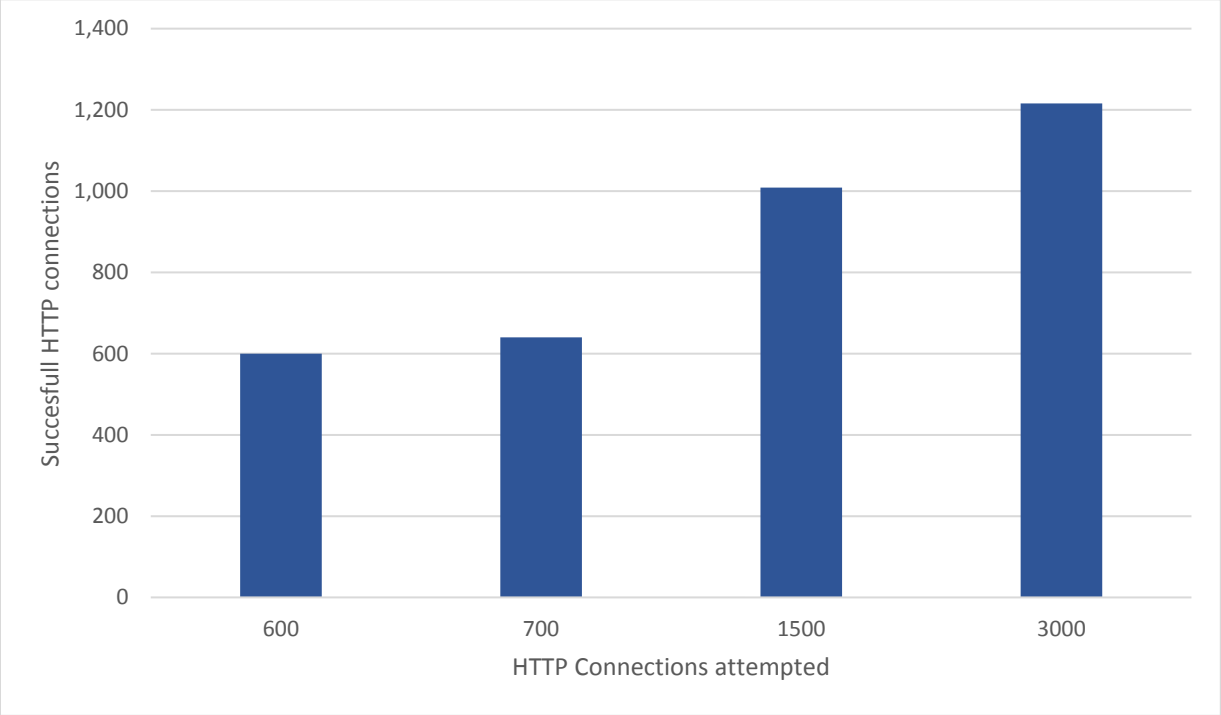


Figure 4.2 Determination of Baseline Performance for HTTP Transactions

Attack traffic is transmitted at different line rates (attack loads), up to a maximum of 1 Gbps. During an experiment attack traffic is initially transmitted at a line rate of 100Mbps for five minutes, subsequent loads increase the line rate of the previous by 100Mbps up to 1Gbps while the parameters of performance are recorded. Legitimate traffic was transmitted five minutes prior to the attack traffic, this was done to provide data for both idle(regular) performance and under-attack performance with legitimate traffic.

The experimental setup for this evaluation was designed to simulate the network described in Figure 3.2 from Chapter III section 3.1 where the Internet and routers can be visualized as a single router and a switch.

4.1.1 Hardware

This subsection will provide hardware specifications of the network and system under attack.

Server Platform

The targeted server system used was Apple's Mac Pro Server (Mac Pro Mid2010) equipped with one 2.8GHz Quad-Core Intel Xeon W3530 "Nehalem" processor (featuring Hyper-Threading technology for up to 8 virtual cores), 16 GB of RAM, [92] and built-in Gigabit Ethernet adapter.

Server specifications are as follows:

Operating Systems: Apple's macOS 10.13.6 High Sierra

CPU: 2.8GHz Quad-Core Intel Xeon W3530 "Nehalem" processor [94]

Number of Processors: 1

Number of Cores: 4 physical cores. 8 virtual cores thru Hyper-Threading

Random Access Memory (RAM): 16 Giga Bytes

Network-Interface-Card (NIC): Two independent 10/100/1000BASE-T Ethernet (RJ-45) interfaces with support for jumbo frames

Graphics Processing Unit: ATI Radeon HD 5770 with 1GB of GDDR5 memory, PCI Express 2.0

Switch

For these experiments Cisco's SRW2024 24-port gigabit switch was utilized. This switch features 24 high-speed ports optimized for bandwidth-intensive applications, with data transfer rates of 10/100/1000Mbps and is compliant with IEEE standards 802.3, 802.3u, 802.3ab, 802.3x, 802.1p, 802.1q. No additional devices were connected at the time of experiments were performed. Documentation for this switch can be found at [95].

4.1.2 Software

To collect and process data regarding the performance parameter several tools were utilized and are documented as follows:

Several terminal commands were used in macOS 10.13.6 High Sierra to observe and record the parameter of performance evaluation. Data logging was achieved by running these commands on the Terminal thru an apple script. The script was designed to launch a limited number of Terminal instances dependent on the data that need to be logged. The Terminal instances received commands to measure the consumption of system resources while recording relevant network traffic data. In addition, TG Pro was used to record CPU temperature [109]. All server fans were controlled by the system during experimental trials. All tools recorded at a sampling rate of one sample per second and stored all data on comma separated value (.csv) files. Execution of monitoring/recording tools has a low impact on server performance as CPU utilization is below 5% and random access memory (RAM) consumption is 33.9 megabytes

(MB) (2 MB from the performance monitoring commands, and 31.9 MB from TG Pro) out of 16 gigabytes (GB).

To process the data recorded a set of short scripts were programmed in MATLAB [98]. The scripts extract relevant data from .csv files and output averages for each traffic load. Microsoft Excel was chosen as the plotting tool for the graphs in the results section [99].

4.2 Parameters of Performance Evaluation

In this experiment, the parameters that are being used in the comparative evaluation of performance were processor utilization, processor temperature, random access memory (RAM) consumption, HTTP transactions per second, number of Echo request and replies received per second, number of Echo replies sent per second, number of ICMP Destination Unreachable messages sent, and UDP datagrams received per second. Some parameters regarding network traffic may be omitted depending on their relevance to the attack implemented. Performance parameters are recorded as the targeted system experiences both legitimate and attack traffic. Performance parameters are more clearly defined in the following pages:

Processor Utilization (Usage of CPU in %) – The central processing unit (CPU) is one of the most crucial hardware components in a computing system as it interprets and executes a wide majority of the commands/instructions relayed by other hardware and software components. For a CPU processing generally involves fetching data, decoding, and executing instructions. Software applications are passive collections of instructions, once the software is executed a process is created containing the program code and its activity. Process are comprised by threads of execution. Threads are short sequences of instructions managed autonomously by the operating system thru a scheduler. Instructions typically involve basic arithmetic, relocating data

on the memory or data comparison. Originally a processor had a single core and only a single thread could be executed at a time. Multithreading was made possible by switching between multiple software threads at a high rate creating the illusion of parallel execution. Modern processors feature multiple cores which has allowed for multiple threads to be executed in parallel thus expediting the run time of several applications. The server used in these experiments has a 2.8GHz Quad-Core Intel Xeon W3530 "Nehalem" processor [94]. This processor features Hyperthreading which allows the operating system to recognize the four physical cores as eight virtual or logical cores thus allowing multicore processors twice as many threads to be processed in parallel which increases the amount of independent instructions in the pipeline. [100] It is worth nothing that not all applications take advantage of hyperthreading, typically applications with high resource consumption such as digital media applications, and gaming titles which make use of 3D rendering, asset streaming, terrain generation, physics effects, video encoding, virus scanning etc. are designed to use hyperthreading [101]. Certain attacks can be extremely detrimental to processor performance and be classified as processor intensive. A computing system may slow down and become unstable if the processor saturation is achieved by attempting to process all incoming attack traffic. Collected data for this parameter will permit the assertion of the impact botnet sizes and types of attacks can have on the system.

Processor Temperature (CPU temperature in °C) – Processor core temperatures are taken into consideration since CPU's core temperatures are directly proportional to processor consumption (Computationally intensive tasks draw more energy and more energy yields more heat). If a processor temperature breaches it's physical limits, the system may become unresponsive and crash. Additionally, physical damage from the heat may be present, prolonged exposure can diminish a processors effectiveness permanently. Generally, CPU core temperatures are

measured at transistor junctions (Tjunction) within each core. This is the hottest area of each core. Intel defines Tjunction as a synonym for core temperatures, they are calculated by subtracting the output from the Digital Thermal Sensor (DTS) from an established Temperature Junction Max (TjMax) also known as the throttle temperature. To prevent thermal damage to the processor core speeds and voltage are reduced (throttle) when a processor's core reaches TjMax. The processor used in these experiments as well as most processors throttle at 100 °C [103]. Alternatively, a measurement for the entire CPU can be obtained from temperature sensor diode at the die. Access to temperatures values can be dependent on both the operating system and the processor itself.

Memory Consumption (RAM consumption in MB) – Random access memory (RAM) is a form of computer data storage which retains short-term machine code and data based on process currently running on a computing system. RAM determines the amount of processes a system can perform. If RAM became fully exhausted, the operating system will resort to use hard drive space as virtual memory. Virtual memory has slower read/write speed compared to RAM, as a result when RAM is exhausted the performance of the system degrades and system stability is lost. The increase of virtual memory used will also be detrimental to system performance.

Echo Request Messages Received per Second (Received Echo Request/Sec) – Echo request messages are an integral part of the ping utility. Echo requests messages received per second will be recorded by the victim system to ascertain if any limits have been imposed by the firewall or built-in security mechanism. As the amount of these requests increases, the victim server will have to process more messages which can have a negative impact on the victim server. This parameter will be observed for Ping attacks.

Echo Reply Messages Received per Second (Received Echo Reply/Sec) – Echo reply messages are an integral part of the ping utility and are only transmitted after an echo request has been received. Echo reply messages received per second will be recorded by the victim system to determine if any security mechanism mitigate Smurf attacks. As the amount of these requests increases, the victim server will have to process more messages which can have a negative impact on the victim server. This parameter will be observed for Smurf attacks.

Echo Reply Messages Sent per Second (Sent Echo Reply/Sec) – Echo reply messages are an integral part of the ping utility and are only transmitted after an echo request has been received. Echo reply messages sent per second will be recorded by the victim system. Per RFC 792 when a system receives an ICMP echo request message it is required to respond with an ICMP echo reply message [61]. Under ping flooding conditions the victim server has to process incoming ICMP echo request packets while attempting to reply to as many as possible (Firewall settings can prevent this). This parameter will be observed for Ping attacks.

ICMP Destination Unreachable Messages Sent per Second (Sent Destination Unreachable/Sec) – ICMP destination unreachable messages are sent when a frame is discarded due to not being able to reach its destination. Per RFC 1122, a host should transmit an ICMP destination unreachable message with code 3 (Port Unreachable) when the designated transport protocol is unable to demultiplex the datagram but lacks a method to notify the sender [59]. This type of messages may be transmitted by the victim server at an alarming rate under UDP Flooding attacks when UDP datagrams reach ports not being listened to by the system. This parameter will be observed for UDP Flooding attacks.

UDP Datagrams Received per Second (Received UDP Datagrams/Sec) – UDP Datagrams received per second will be recorded by the victim system to determine if any limits have been

imposed by the firewall or other built-in security mechanism. This parameter will be observed for UDP Flooding attacks.

HTTP Transactions per second (HTTP Transaction/sec) – Successful HTTP transactions per second will be recorded by an external client transmitting HTTP requests (HTTP GET) to the victim server. Successful HTTP transactions involve the transmission of a request thru the HTTP GET command and receiving a response. The number of successful transactions compared with the baseline will indicate what sort of impact DDoS attacks can have on the server's network connectivity.

4.3 Results and Discussion

Experimental results will be reviewed in this section. In these experiments the performance and security strategies of Apple's macOS 10.13.6 High Sierra operating systems are evaluated under legitimate and attack traffic.

The parameters of evaluation discussed in section 4.2. will be utilized in the following subsections where the endurance of both operating system against ICMP Ping flooding, Smurf attacks, TCP SYN flooding, and UDP flooding is compared. To simulate botnet conditions the IP address of the attack traffic will be spoofed to replicate fully compromised Class A, B, and C networks. Networks in these classifications can hold up to 16777214 hosts, 65534 hosts, and 254 hosts respectively. Spoofed IP address were chosen at random, this is done to replicate a practical scenario where a bot master may choose the same action or may send attack traffic from different sources at random depending on each bot's bandwidth availability. As legitimate traffic attempts to reach the webserver, parameters of evaluation will be recorded as each attack traffic is transmitted using a line rate in the range of 0 to 1 Gbps with a step size of 100 Mbps in the

same form as the attacks of Chapter III. Baseline performance will be accounted for as legitimate traffics reaches the webserver and attack traffic is not present. Baseline performance will dictate normal operating conditions which will be compared with each attack bandwidth implemented. Graphs and tables will provide a visual representation of data acquired.

4.3.1 Layer 3 ICMP Ping Attack

The number of Echo requests received by macOS High Sierra was not affected by the size of the simulated botnets employed. As demonstrated in Figure 4.3 regardless of the simulated botnet network employed the average amount of echo requests received per second was kept within the same range for both server operating systems where only the line rate of attack traffic would impact this number up to 500 Mbps of attack traffic. Higher attack traffic loads saw no increase in the amount of echo requests received.

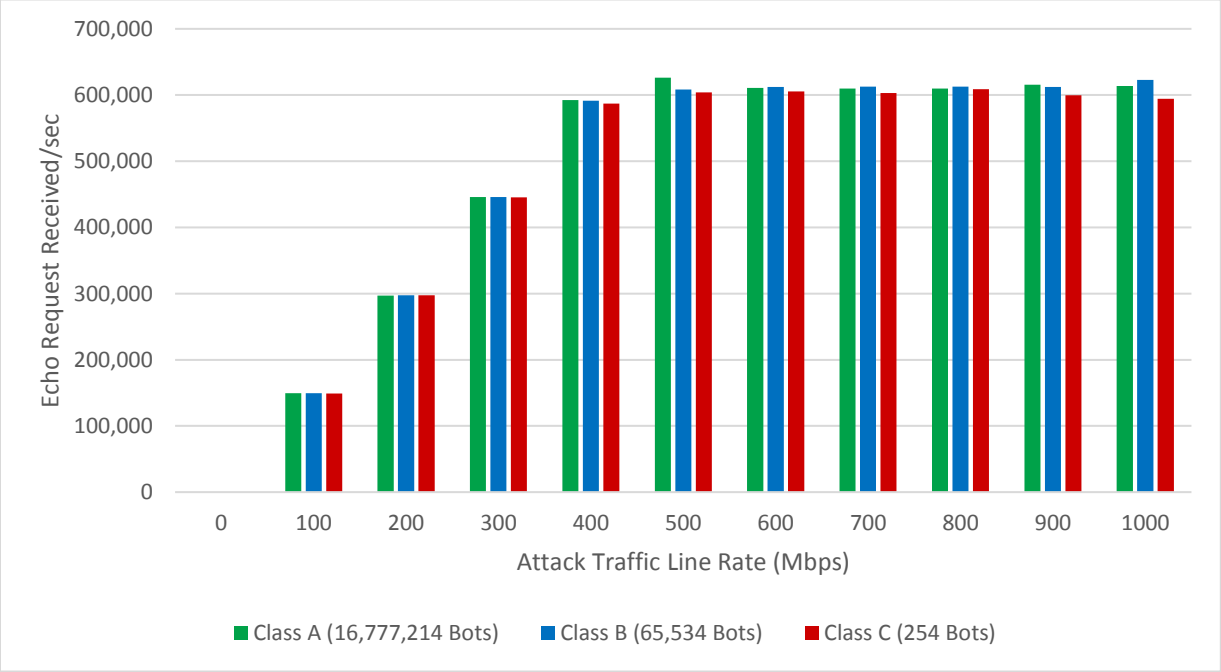


Figure 4.3 Echo Request Received/sec under Ping Attacks on macOS High Sierra

RFC 792 states that when a system receives an ICMP echo request message it is obligated to respond with an ICMP echo reply message [61] however, macOS High Sierra would only reply at a rate of 250 echo replies per second under attack conditions. By limiting the amount of echo replies transmitted the operating system saves processing power and bandwidth.

The baseline performance for HTTP Transactions/sec was found to be maintained up to 200 Mbps of Ping attack traffic. Subsequently, an attack traffic of 300 Mbps triggered a slight decrease in the amount of HTTP transaction of 2.7%. Higher attack traffic loads saw connection gradually decrease as the attack traffic load increased. Based on data collected it appears botnet size does not diminish the amount of HTTP transaction conversely; attack traffic loads can affect this number as it increases. The network connectivity of macOS High Sierra under Ping attacks is demonstrated in Figure 4.4

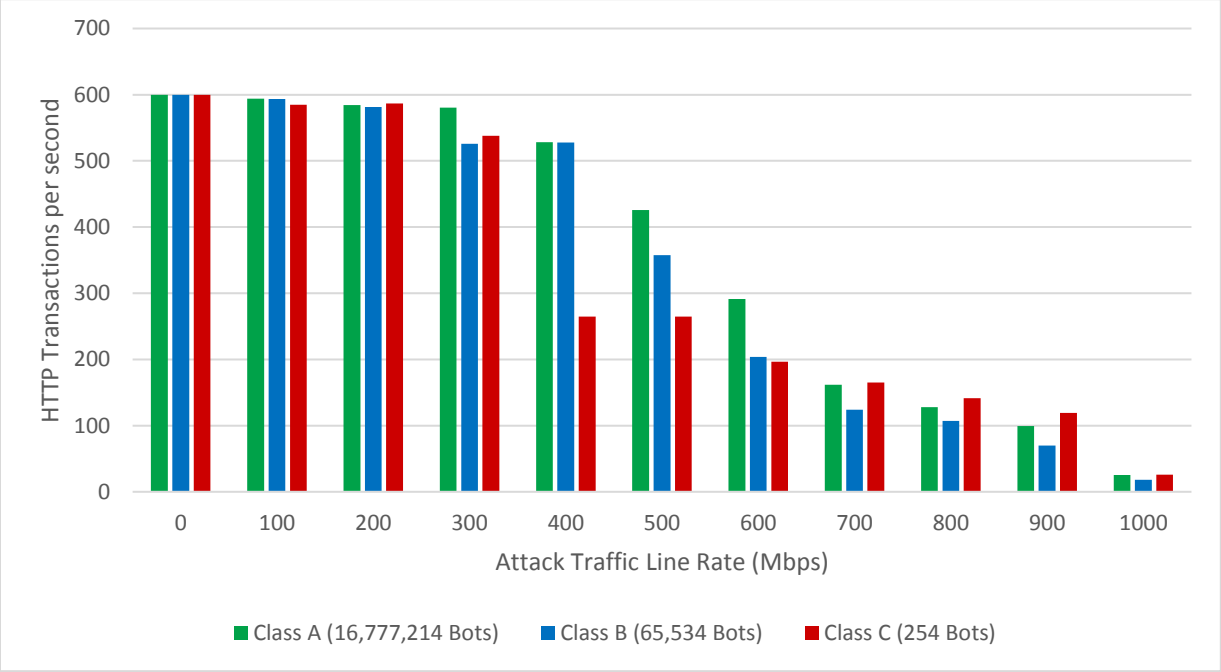


Figure 4.4 HTTP Transactions/sec under Ping Attacks on macOS High Sierra

In these experiments botnet sizes did not affect the rate at which RAM was consumed by the operating system however higher Ping attack traffic loads triggered a maximum consumption of 4.5 GB, which is 1.5 GB higher than baseline performance. The system characteristics of the server under attack feature 16 GB, of RAM. Ping attacks at their worst consumed nearly 9.4% of all available RAM which could detrimental to systems that are actively multitasking highly demanding applications. The overall memory consumption for macOS High Sierra under Ping attacks can be found on Figure 4.5.

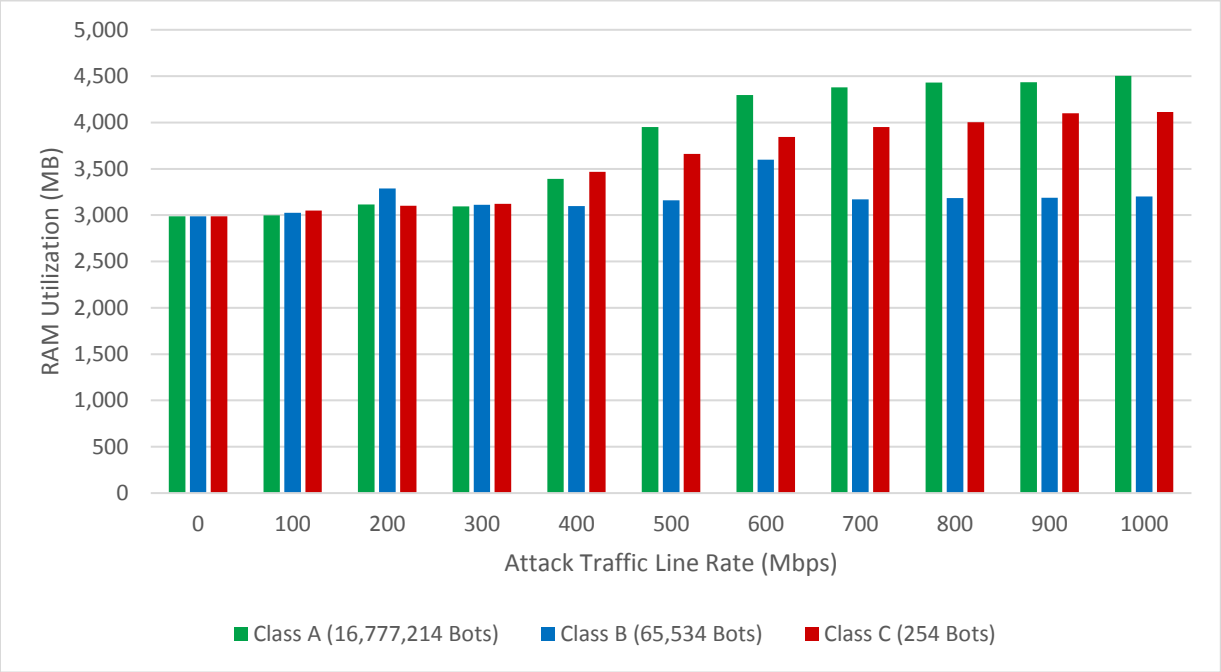


Figure 4.5 Total Memory Consumption under Ping Attacks on macOS High Sierra

Baseline CPU performance was not affected either by botnet size. CPU consumption was at highest under an attack traffic load of 400 Mbps with a maximum CPU consumption of 21.91%. Baseline performance for CPU utilization was kept at 9.34%. The introduction of Ping attack traffic saw CPU consumption rise to nearly 12.50% at 100 Mbps, consumption kept rising until an attack traffic load of 400 Mbps was reached. Interestingly Ping attack traffic loads higher

than 500 Mbps experienced a gradual decline of CPU consumption. Earlier in this subsection it was demonstrated that the number of Echo requests received per second was the same for all botnet sizes. Similar CPU utilization was experienced regardless of botnet size at different attack traffic loads, this could be attributed to a similar amount of echo request being received independently of botnet size. It is likely that CPU consumption was lowered after 500 Mbps because as seen on Figure 4.4 the rate at which HTTP transactions are made is diminished with higher traffic loads which would cause the web server in macOS High Sierra to utilize less resources as HTTP connections cannot be made, this behavior is exhibited also by Smurf Attacks. Total CPU utilization is demonstrated in Figure 4.6.

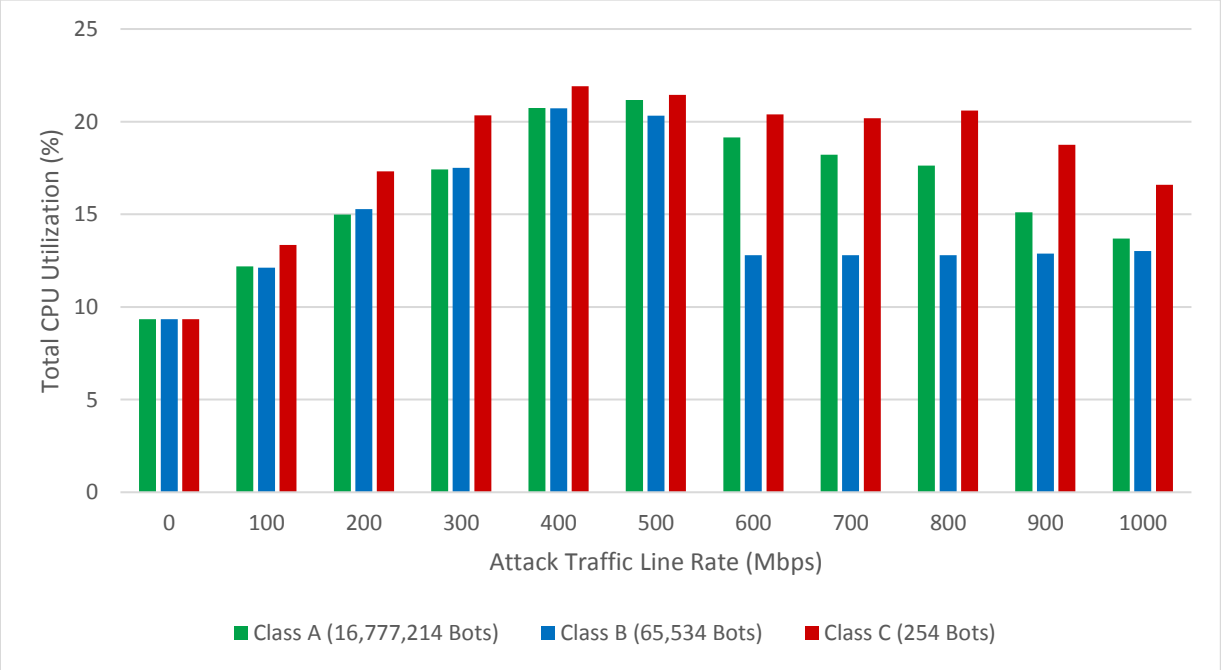


Figure 4.6 CPU Depletion under Ping Attacks on macOS High Sierra

Lastly CPU temperatures on macOS High Sierra reached their highest point under 500 Mbps of attack traffic load reaching nearly 57 °C. Higher attack traffic loads kept similar temperatures on average above 50 °C without surpassing the maximum established at 500 Mbps

where a high consumption of CPU power was experienced. These temperatures are below TjMax and could be consider safe for this hardware. The temperatures of the CPU reflect the amount of processing power consumed and can be seen in Figure 4.7.

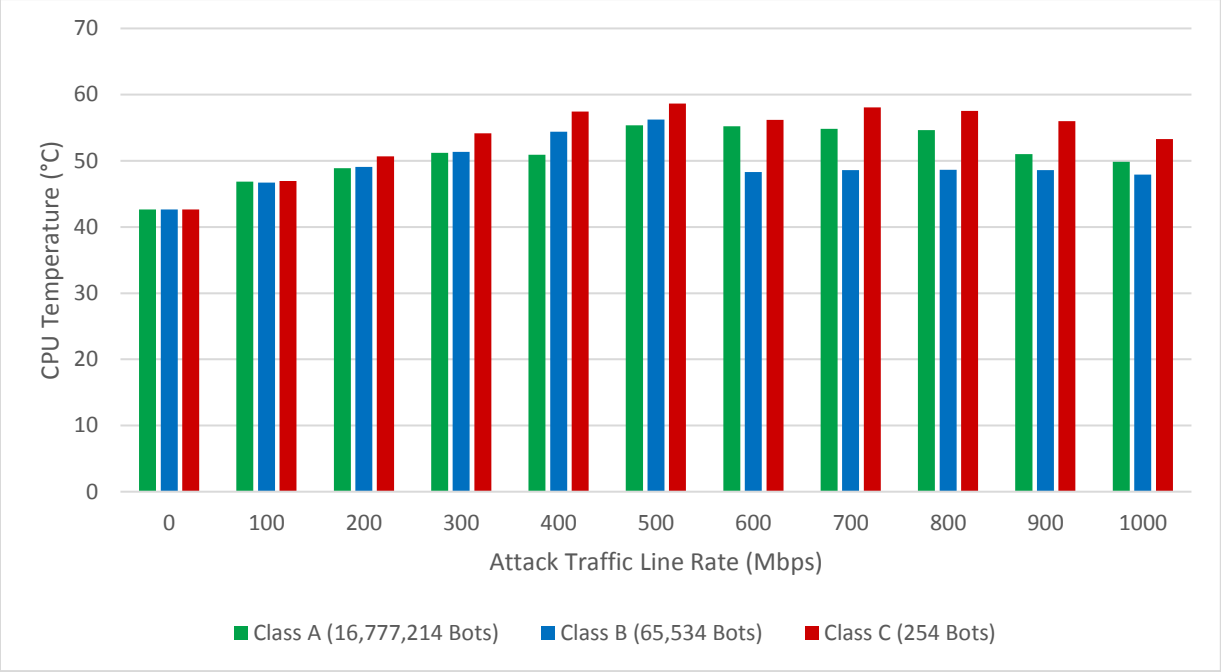


Figure 4.7 CPU Temperature under Ping Attacks on macOS High Sierra

4.3.2 Layer 3 ICMP Smurf Attack

The number of Echo reply messages reaching the targeted server was found to not benefit from larger botnet sizes but was found to increase with higher attack traffic loads up to a maximum of 600,000 echo reply messages on average as seen on Figure 4.8 where the rate at which Echo reply messages reached the targeted system for different attack traffic loads.

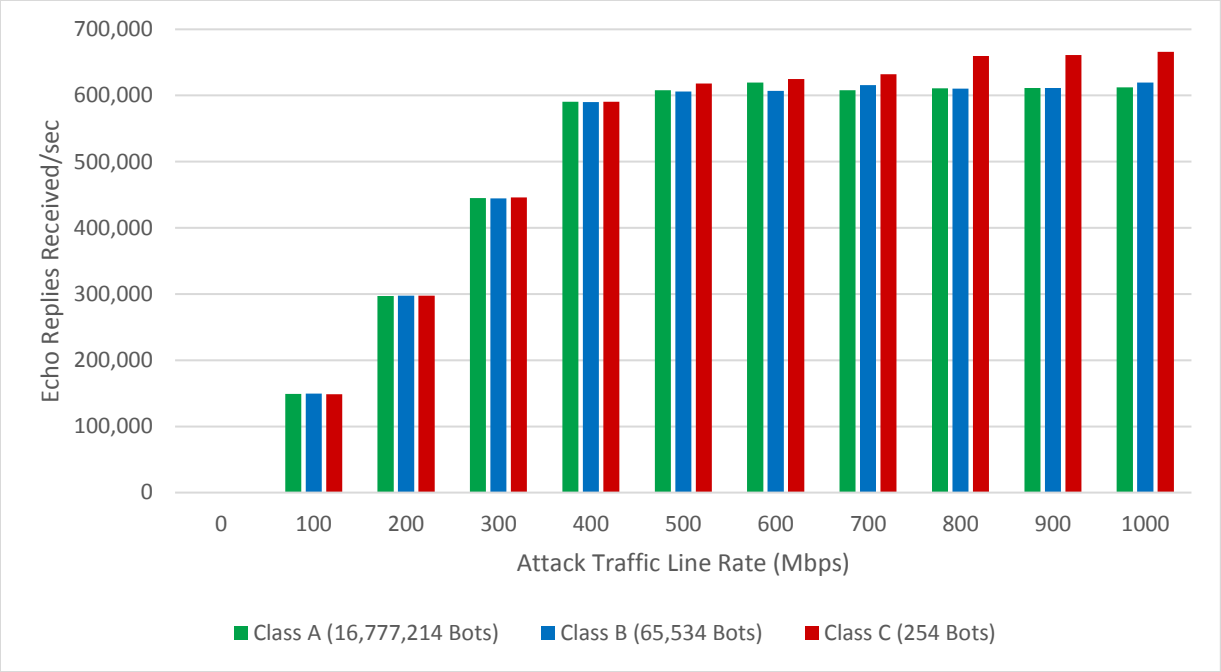


Figure 4.8 Echo Request Received/sec under Smurf Attacks on macOS High Sierra

Furthermore, the maximum average number of messages was found at 1Gbps with 665,894 Echo reply messages which is a small increment when compared with the number of messages met at 500 Mbps.

The baseline performance for HTTP Transactions/sec was found to decline as soon as attack traffic reached the server where lower attack traffic loads in the range of 100 to 300 Mbps experience a slight decline in connectivity however, at 400 Mbps and higher attack traffic loads a trend became more apparent and connections gradually decayed to a minimum of 15 transactions per second under 1 Gbps of attack traffic. Similarly, to Ping attacks, Smurf attacks did not benefit from higher botnet sizes but benefited instead from higher attack traffic loads. The network connectivity of macOS High Sierra under Smurf attacks is demonstrated in Figure 4.9

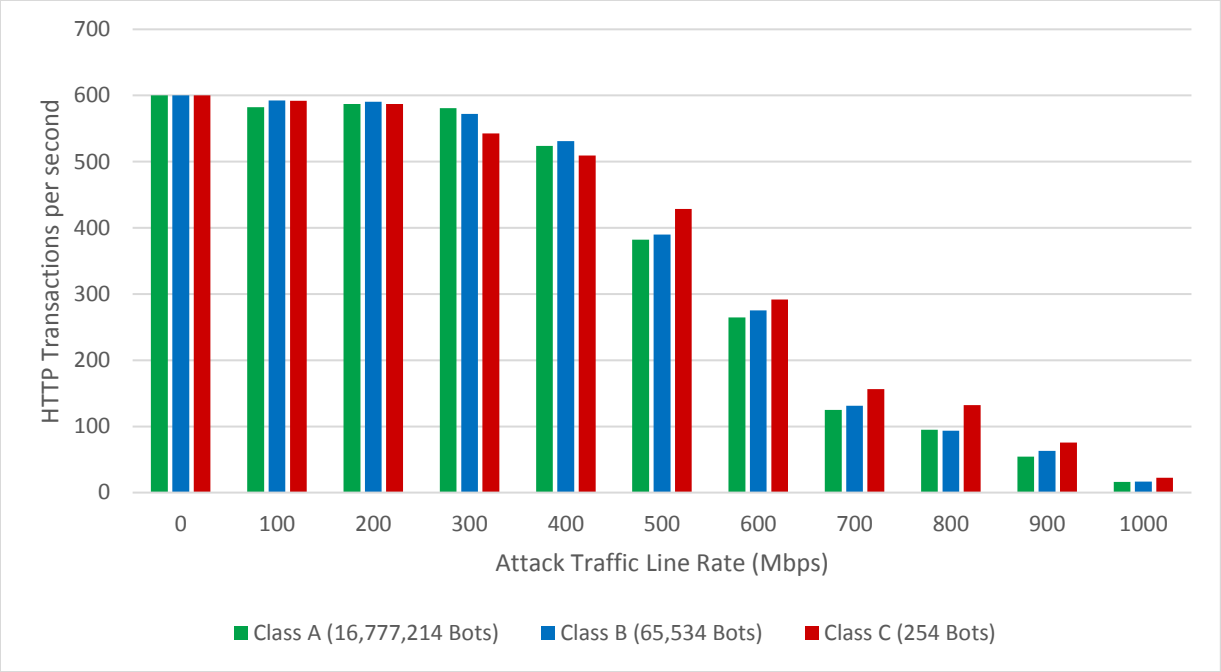


Figure 4.9 HTTP Transactions/sec under Smurf Attacks on macOS High Sierra

As for memory consumption, botnet sizes did little to no impact on the rate at which RAM was consumed by the operating system yet higher Smurf attack traffic loads caused a maximum consumption of 4.46 GB, which is 1.46 GB higher than baseline performance. It was also observer that Class A botnets would consumed almost 200 MB more than its smaller counterparts. The system characteristics of the server under attack feature 16 GB, of RAM. Smurf attacks at their worst consumed nearly 9.13% of all available RAM which could be detrimental to systems that are actively multitasking highly demanding applications. The overall memory consumption for macOS High Sierra under Smurf attacks can be found on Figure 4.10.

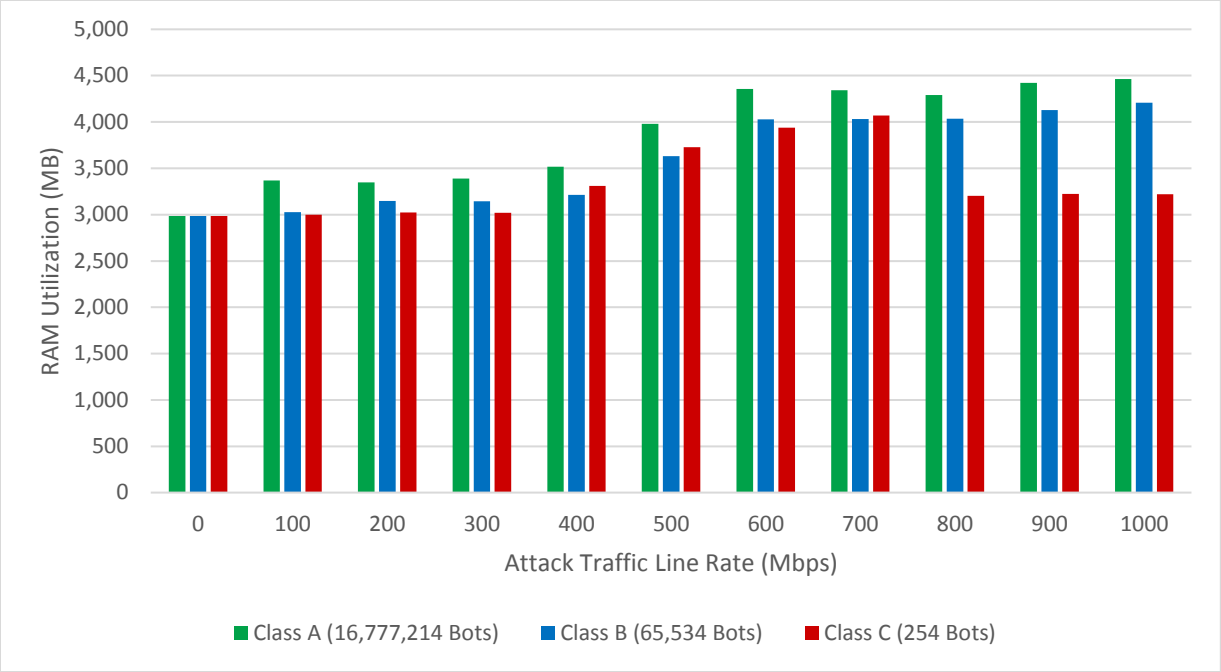


Figure 4.10 Total Memory Consumption under Smurf Attacks on macOS High Sierra

Large botnet sizes had no impact on CPU performance. Similar CPU utilization was experienced regardless of botnet size at different attack traffic loads, this could be attributed to a similar amount of echo replies being received. Baseline performance for the CPU was 9.34%. The introduction of Smurf attack traffic saw CPU consumption rise to 12% at 100 Mbps, subsequent higher attack traffic loads increase this utilization until 22% CPU utilization was achieved by attack traffic loads in the range of 400 Mbps and 500 Mbps. It is likely that CPU consumption was lowered after 500 Mbps because as seen on Figure 4.9 the rate at which HTTP transactions are made is diminished with higher traffic loads which would cause the web server in macOS High Sierra to utilize less resources as HTTP connections cannot be made, this behavior is exhibited also by Ping Attacks. Total CPU utilization is demonstrated in Figure 4.11.

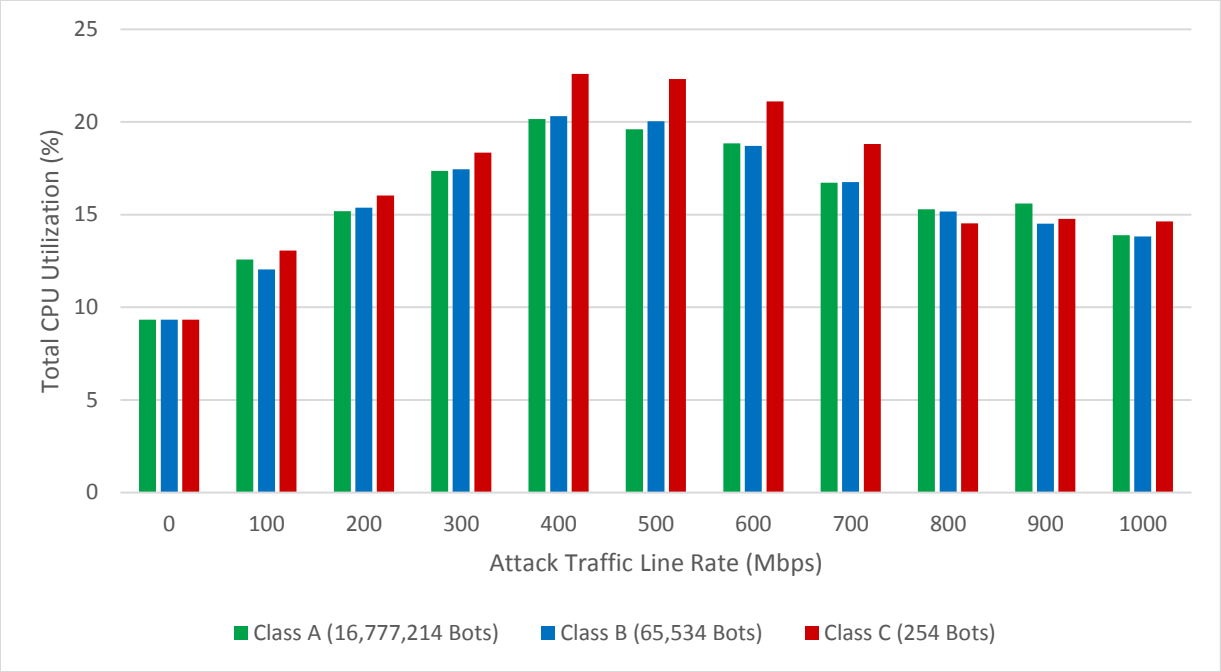


Figure 4.11 CPU Depletion under Smurf Attacks on macOS High Sierra

Lastly CPU temperatures on macOS High Sierra reached their highest point under 600 Mbps of attack traffic load reaching nearly 58 °C. Higher attack traffic loads kept similar temperatures on average above 50 °C without surpassing the maximum established at 600 Mbps where a high consumption of CPU power was experienced. These temperatures are below TjMax and could be consider safe for this hardware. The temperatures of the CPU reflect the amount of processing power consumed and can be seen in Figure 4.12.

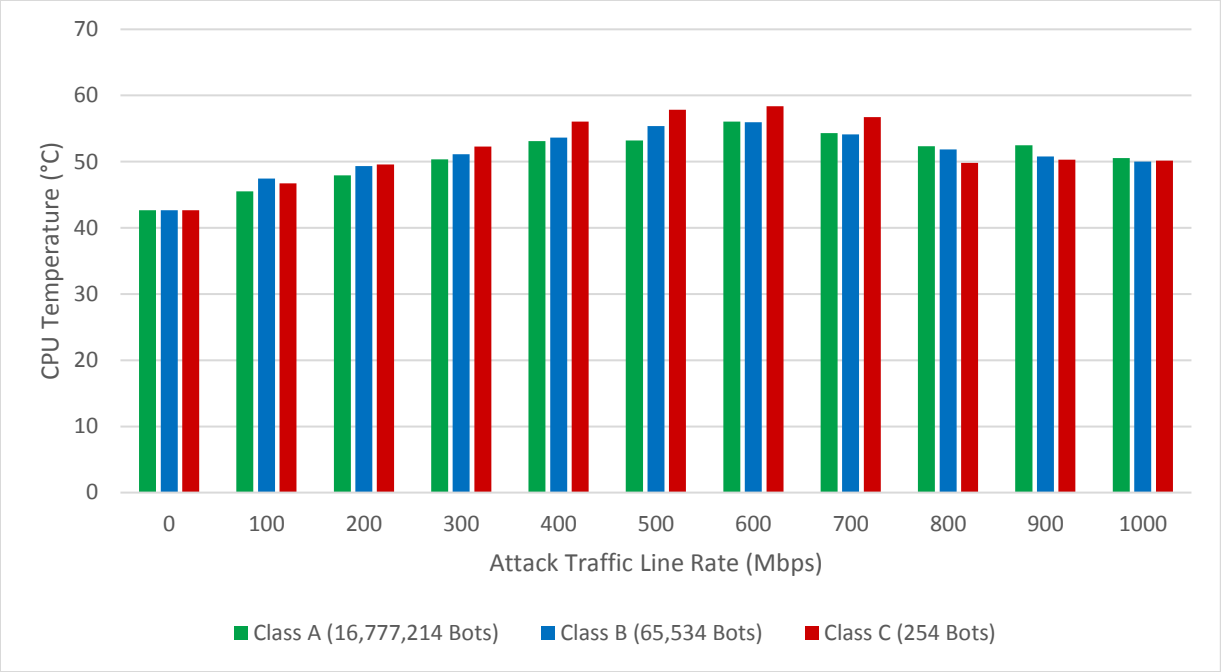


Figure 4.12 CPU Temperature under Smurf Attacks on macOS High Sierra

4.3.3 Layer 4 TCP SYN Flood Attack

TCP SYN Flooding does not benefit from a larger botnet size when the objective is to eliminate all network connectivity from macOS High Sierra. The baseline performance for HTTP transactions per second was not able to be maintained at any traffic load. At 100 Mbps HTTP transaction decayed to 8 transactions per second. Higher attack traffic loads did not allow a single HTTP connection to be made. This can be attributed to the nature of TCP SYN Flooding. As the transmission control block fills up legitimate HTTP connections are not able to be performed. The network connectivity of macOS High Sierra under TCP SYN flooding attacks is demonstrated in Figure 4.13

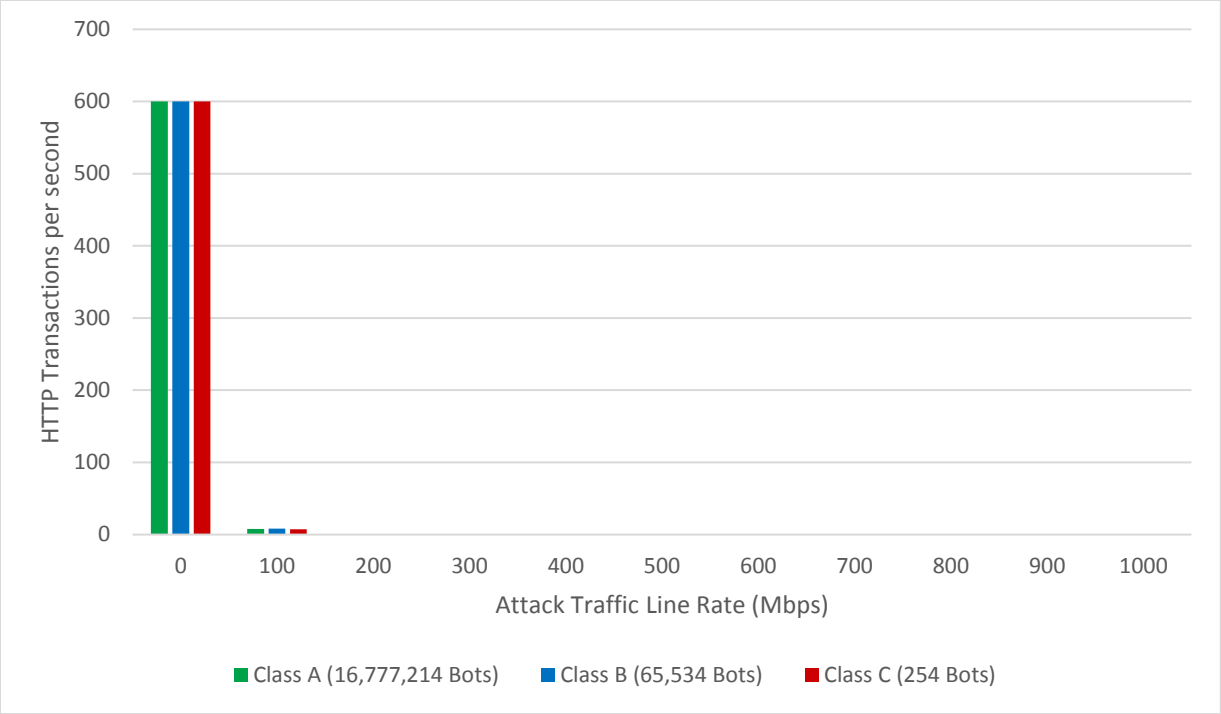


Figure 4.13 HTTP Transactions/sec under TCP SYN Flooding on macOS High Sierra

Of all DDoS attacks performed on macOS High Sierra, TCP SYN flooding attacks had the lowest impact on memory consumption for this platform. Memory consumption for TCP SYN flooding was lower than that of ICMP based attacks and UDP flooding attacks. Additionally, the consumption of memory fluctuated across botnet sizes, implying that botnet size does not have a direct effect on memory consumption. The highest amount of memory consumption experienced by macOS High Sierra was of 3.4 GB, being 414 MB higher than baseline performance. The system characteristics of the server under attack feature 16 GB, of RAM. TCP SYN flooding attacks at their worst consumed nearly 2.6% of all available RAM. With such a low memory consumption TCP SYN flooding attacks can be considered negligible for modern systems such as macOS High Sierra in terms of memory. The overall memory consumption for macOS High Sierra under Smurf attacks can be found on Figure 4.14.

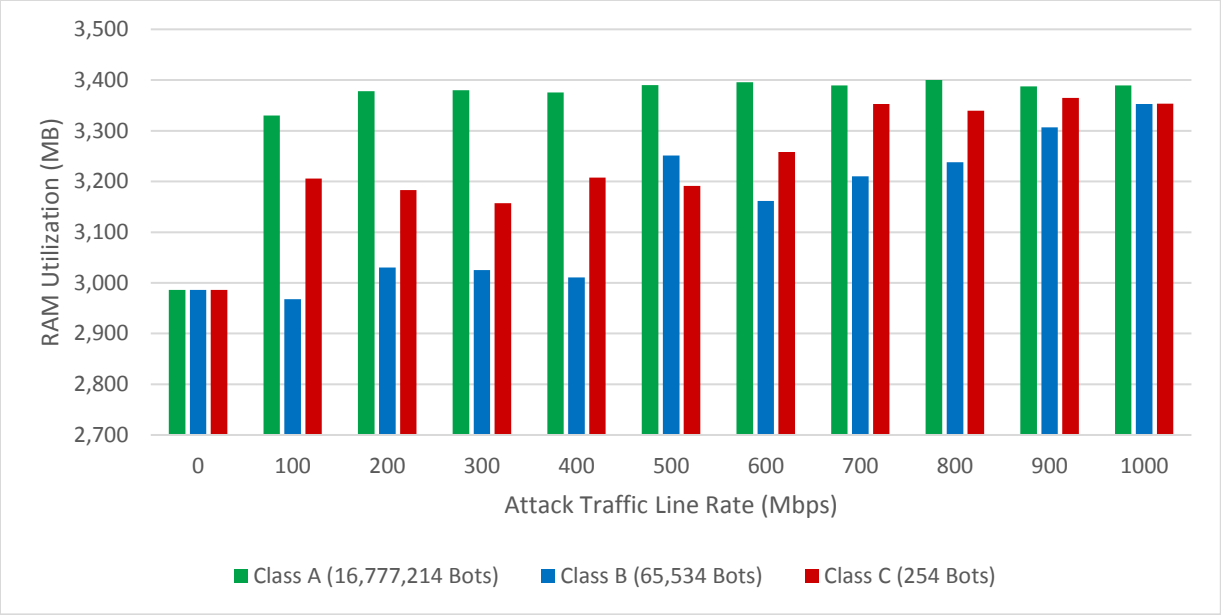


Figure 4.14 Total Memory Consumption under TCP SYN Flooding on macOS High Sierra

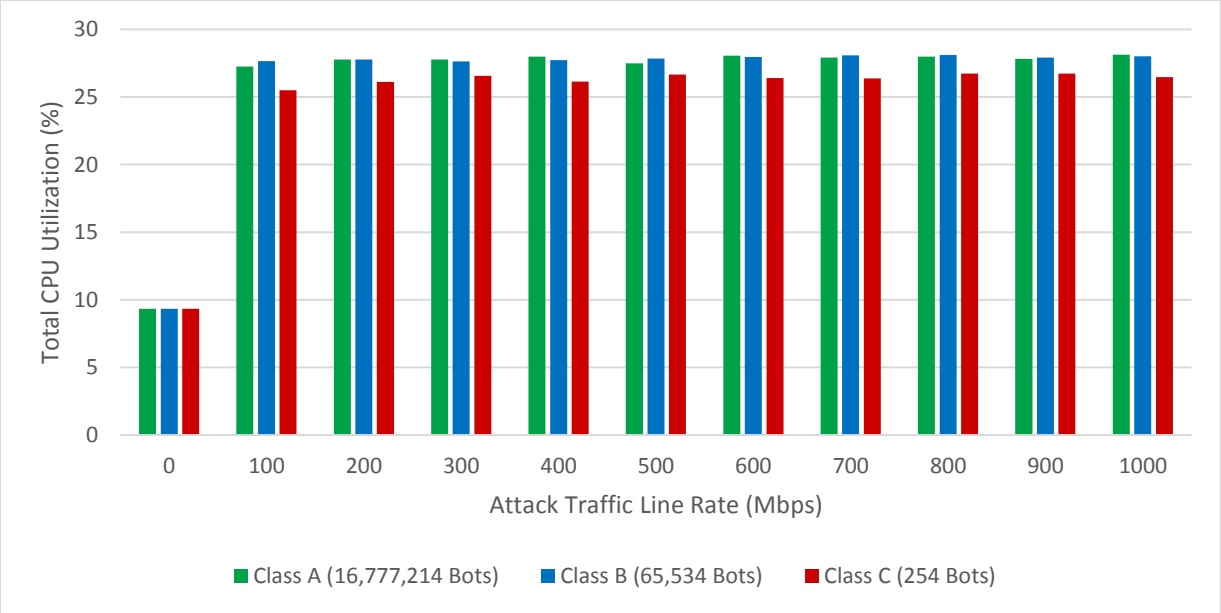


Figure 4.15 CPU Depletion under TCP SYN Flooding on macOS High Sierra

The effect of larger botnets employed was negligible, as for all traffic loads and botnet sizes kept CPU Utilization at 27% on average. It appears macOS High Sierra has a small limit to the amount of TCP connections or half open connection it can hold. This is backed up by the low

amount of HTTP transaction able to be achieved as demonstrated by Figure 4.2, where the baseline performance was limited by the amount of connection macOS High Sierra would allow. Total CPU utilization is demonstrated in Figure 4.15.

Lastly CPU temperatures on macOS High Sierra were rather warm and constant through the attacks. CPU temperature reached average temperatures around 72 °C with a max temperature of 77.34°C. These temperatures are below TjMax but are considered warm and just a few steps below threatening temperatures. The temperatures of the CPU reflect the amount of processing power consumed and can be seen in Figure 4.16.

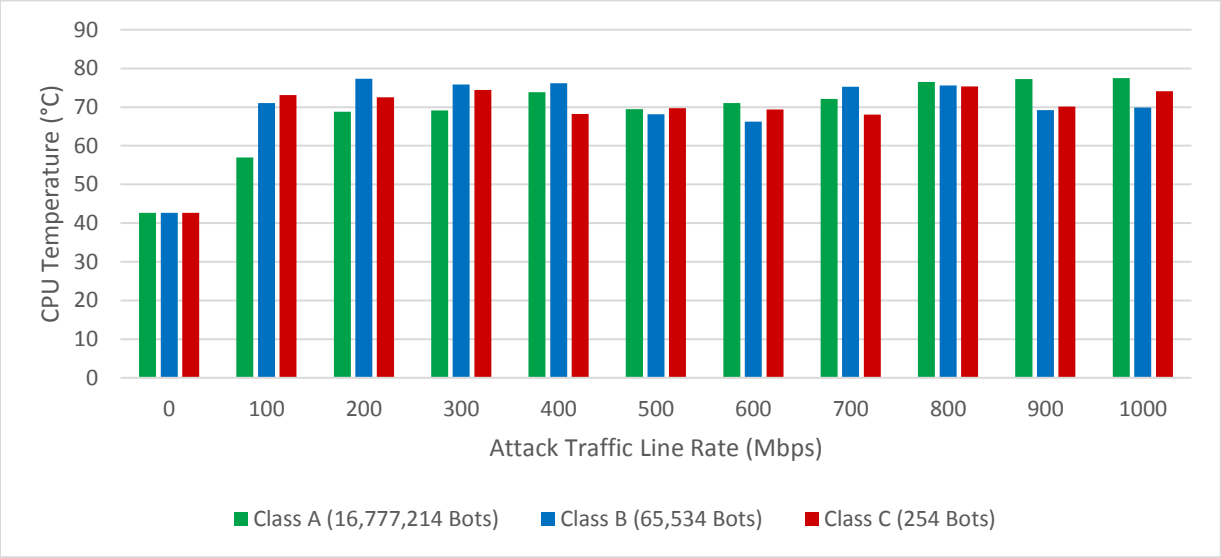


Figure 4.16 CPU Temperature under TCP SYN Flooding on macOS High Sierra

4.3.4 Layer 4 UDP Flood Attack

The number of UDP datagrams reaching the targeted server was found to not benefit from larger botnet sizes as the number of datagrams received was similar at all attack traffic loads regardless of the botnet size. The number of datagrams received while under UDP flooding attack was limited to nearly 600,000 datagrams. This behavior is demonstrated on Figure 4.17

where it can be seen that as attack traffic load increases the number of datagrams does so as well. The number of UDP datagrams kept rising to 500 Mbps of attack traffic where the maximum number of datagrams received on average was found to be 632,128.12 UDP datagrams. Moreover, UDP attack traffic transmitted targeted a port not been listened to by an application. Per RFC 1122 in the event a system receives a UDP datagram at a port that is not being listened to by an application will prompt a response in the form of an ICMP packet stating the destination was unreachable. During UDP flooding experiments it was found that macOS High Sierra would transmit ICMP destination unreachable messages however it would limit the amount to a maximum of 250 messages in the same manner as ICMP messages were limited for Ping attacks. By limiting the number of ICMP messages transmitted the operating system saves processing power and bandwidth.

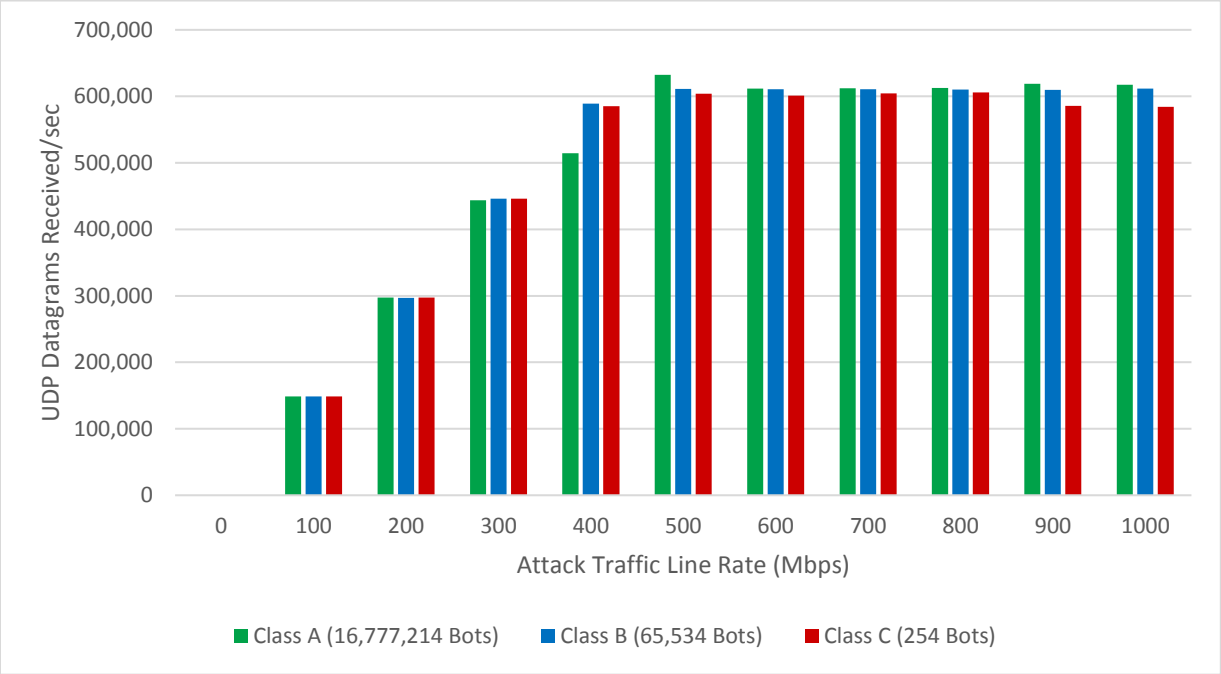


Figure 4.17 UDP Datagrams Received/sec under UDP Flooding on macOS High Sierra

Baseline performance for HTTP transaction per second was upheld mostly up to 100 Mbps, meanwhile attack traffic loads in the range of 200Mbps to 400 Mbps experienced roughly 12% of all HTTP connections. At 500 Mbps and higher attack traffic loads a trend became more apparent and connections gradually decayed to a minimum of 17 transactions per second under 1 Gbps of attack traffic. Similarly, to ICMP based attacks, UDP flooding attacks did not benefit from higher botnet sizes but benefited instead from higher attack traffic loads. The network connectivity of macOS High Sierra under UDP flooding attacks is demonstrated in Figure 4.18.

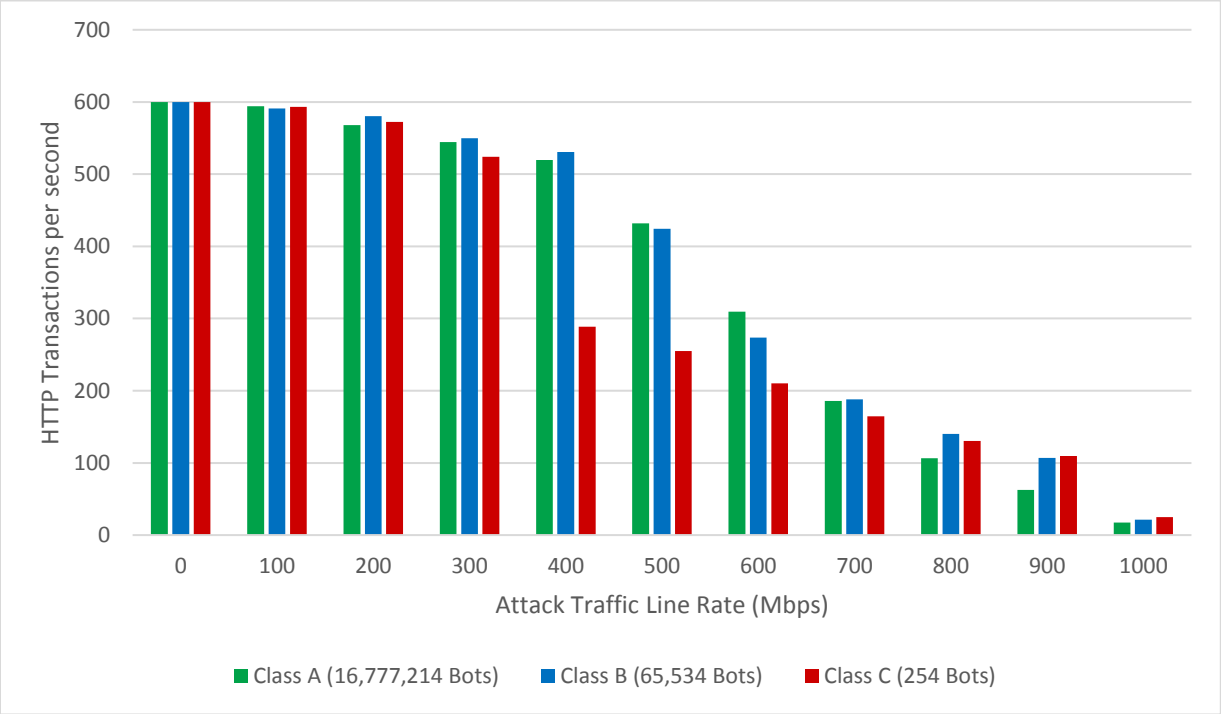


Figure 4.18 HTTP Transactions/sec under UDP Flooding on macOS High Sierra

UDP flooding attacks did not appear to benefit from a larger botnet size as memory consumption for different attack loads fluctuated giving no clear advantage to either size. UDP flooding at its worst cause baseline performance of 2.98 GB to rise to 4.2 GB at 500 Mbps of UDP Flood attack traffic, but on average was kept closer to 3.5 GB. The system characteristics

of the server under attack feature 16 GB, of RAM. UDP flooding attacks at their worst consumed nearly 7.63% of all available RAM which could be detrimental to systems that are actively multitasking highly demanding applications however on average 3.25% of all available memory was consumed. The overall memory consumption for macOS High Sierra under UDP flooding attacks can be observed on Figure 4.19.

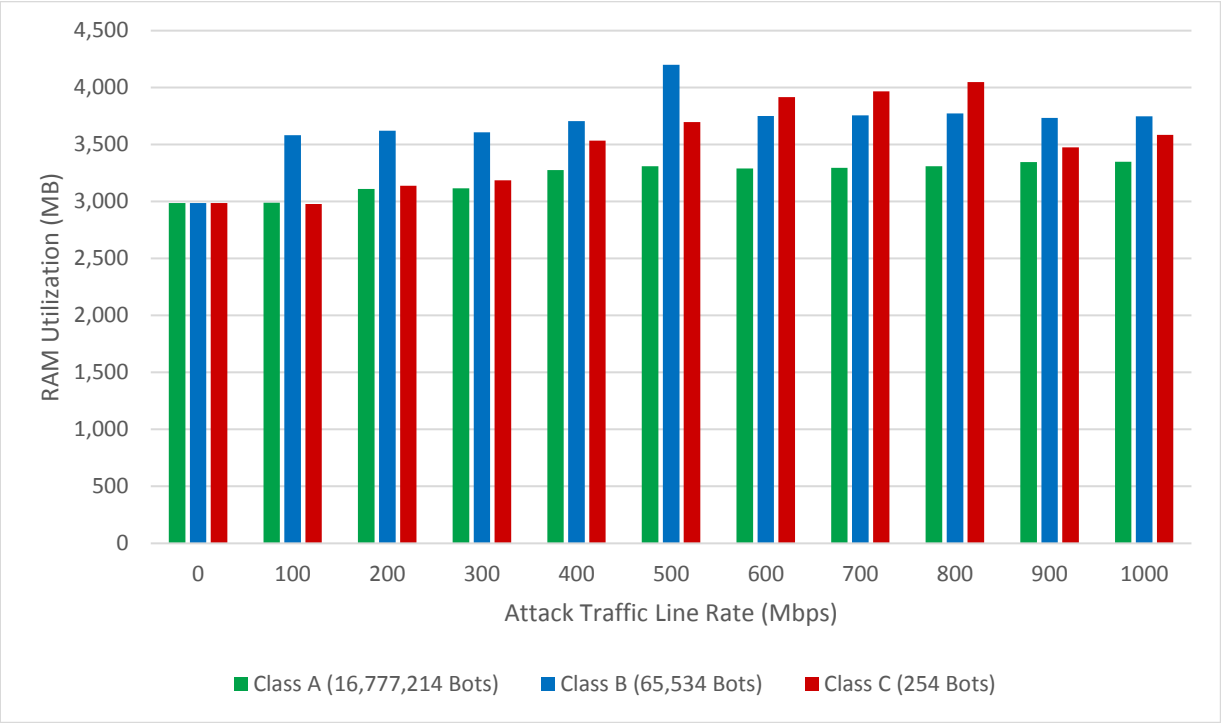


Figure 4.19 Total Memory Consumption under UDP Flooding on macOS High Sierra

UDP flooding did not benefit from larger botnet sizes in terms of CPU utilization. Furthermore, CPU utilization fluctuated at higher traffic loads dependent on the processing of HTTP transaction. If less transactions are being performed it is possible that the HTTP traffic lost was consuming more CPU power than the attack traffic. Baseline performance for the CPU was 9.34%. The introduction of UDP flooding attack traffic saw CPU consumption rise to nearly 13% at 100 Mbps, subsequent higher attack traffic loads increase this utilization until 21% CPU

utilization was achieved by attack traffic loads in the range of 400 Mbps and 500 Mbps. Total CPU utilization is demonstrated in Figure 4.20.

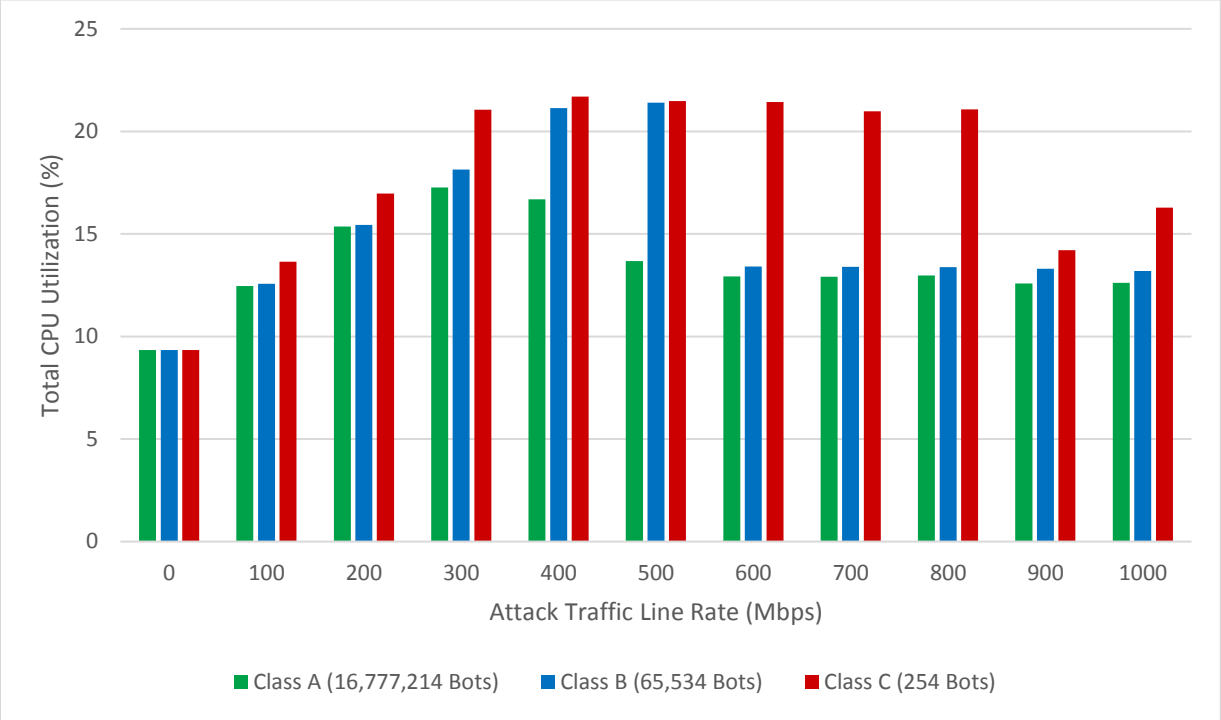


Figure 4.20 CPU Depletion under UDP Flooding on macOS High Sierra

Lastly CPU temperatures on macOS High Sierra reached their highest point under the range of 500 Mbps and 600Mbps of attack traffic load reaching nearly 58 °C. Higher attack traffic loads kept similar temperatures on average almost 50 °C. These temperatures are below TjMax and could be consider safe for this hardware. The temperatures of the CPU reflect the amount of processing power consumed and can be seen in Figure 4.21.

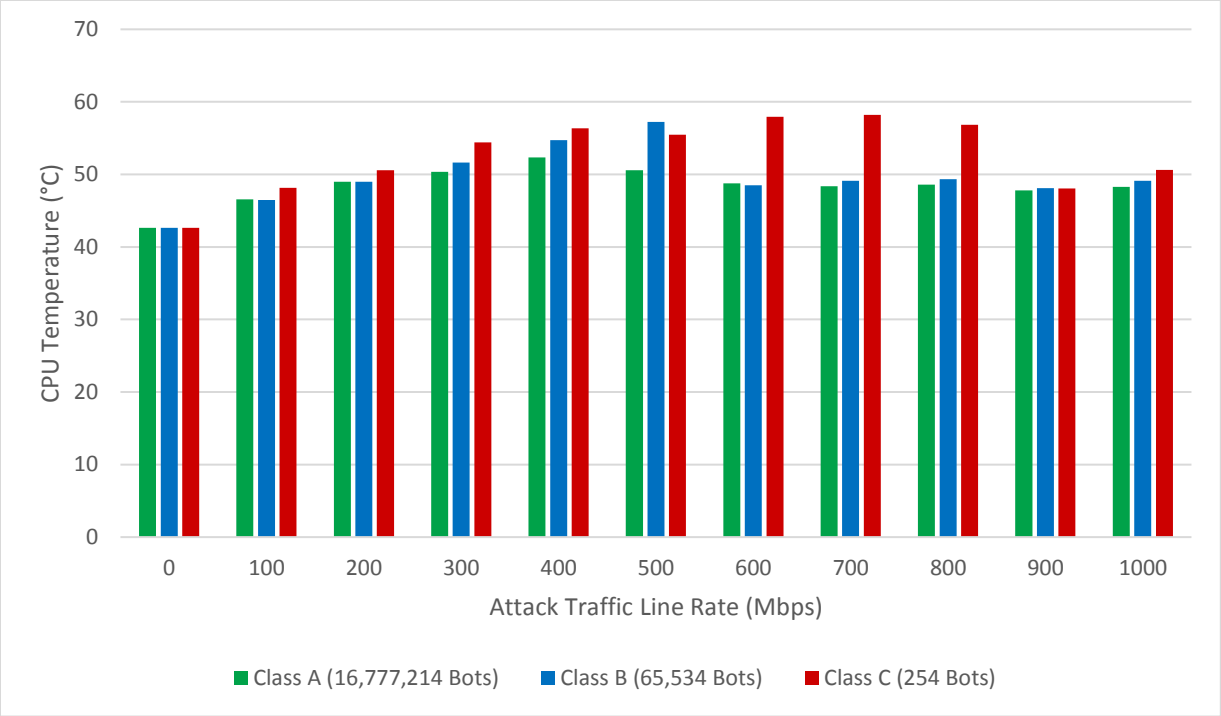


Figure 4.21 CPU Temperature under UDP Flooding on macOS High Sierra

4.4 Chapter Summary

In this Chapter, macOS High Sierra 10.13.6 was deployed on Apple's MacPro mid(2010). The operating systems was subject to three different botnet sizes replicating compromised Class A, B, and C networks which carried out four distinct Distributed Denial of Service attack namely Ping attack, Smurf attack, TCP SYN flooding, and UDP Flooding.

The impact of these attacks was measured utilizing several terminal commands and TG Pro for temperature readings [109]. Moreover, Microsoft Excel and MATLAB were used to interpret the data acquired, by translating raw data from .csv files into graphs [98-99]. Graphs were generated with these tools allow for the presentation of data in a clean, concise manner which helps readers visualize and compare the performance achieved by macOS High Sierra while under attack.

At the cost of connectivity, macOS High Sierra defends its processing power rather well. Hyperthreading technology combined with the security features of macOS High Sierra make a good combination at limiting the impact of these attacks on processing power. Additionally, none of the attacks benefited from larger botnets sizes conversely, the attack traffic line rate used was more impactful in most attacks at the load increased.

Frequently these attacks triggered a decline in CPU utilization when higher amounts of HTTP traffic were lost. This behavior was experienced by both ICMP based attack and UDP flooding. Unlike the other attack performed in this platform TCP SYN flooding maintained CPU utilization throughout the attack as HTTP connections lost were replaced by half open connections generated by the attack.

Under Ping attacks it was observed that macOS High Sierra received similar amounts of echo request messages regardless of botnet size additionally, as a security mechanism macOS limited the amount of echo replies generated to 250 per second allowing to preserve processing power. It was apparent that the number of echo request received became limited at 500 Mbps where on average subsequent traffic loads would hardly go beyond 600,000 echo requests received per second. In terms of network connectivity, a trend was seen where at 300 Mbps nearly 100 HTTP transactions were lost and was worsened at higher traffic loads as bandwidth became congested. Ping attacks had the highest utilization of memory where 1.5 GB of RAM would be consumed by the attack at the highest load. High consumption was experienced as early as 500 Mbps with 900MB being consumed by the attack. Additionally, at its worst Ping attacks presented a CPU consumption of 21.91% which yielded a CPU temperature of 57 °C.

Smurf attacks had similar effectiveness against CPU utilization. The highest amount of CPU utilization was of 22%, which yield a CPU temperature of 56 °C. Smurf attacks had the

second highest utilization of memory were 1.46 GB of RAM would be consumed by the attack at the highest load. The botnet size had little to no effect on the parameters just mention, which is attributed to the number of echo replies received which was similar regardless of botnet size. The same behavior experienced for Ping attacks was seen on network connectivity for Smurf attacks where connections would gradually extinguished as attack traffic increased.

The most devastating attack for macOS High Sierra was found in these experiments to be TCP SYN flooding. Under these attacks HTTP transactions rapidly declined at 100 Mbps to nearly none regardless of botnet size. Memory consumed by this attack was the lowest out of the four types implemented, with only a 414 MB consumed by the attack. TCP SYN flooding consumed the highest amount of CPU power with any botnet size. CPU consumption regardless of attack traffic line rate was maintained at 27% yielding temperatures in the range of 72°C to 77°C.

It was observed that UDP Flooding was not benefitted by a larger botnet size. The number of received UDP datagrams was similar across all three botnet sizes. It is worth noting that ICMP destination unreachable messages were limited to a transmitted at a rate of 250 messages per second under the effects of this type of attack. Max CPU consumption was met at attack traffic loads in the range of 300 Mbps to 800 Mbps where CPU utilization was 21% on average. HTTP connections behaved similarly to ICMP based attacks. Memory consumption for UDP fluctuated around but would rise to nearly 1GB for higher attack traffic loads.

CHAPTER V

COMPARATIVE EVALUATION OF MACOS HIGH SIERRA WINDOWS SERVER 2012 R2 AND WINDOWS 2016 DATACENTER UNDER THE EFFECT OF DISTRIBUTED DENIAL OF SERVICE ATTACKS FROM BOTNETS

In this chapter the security features of Microsoft's Windows Server 2012 R2, Windows Server 2016 and Apple's macOS High Sierra are assessed thru a comparative evaluation where the performance of each operating system is compromised thru DDoS attacks executed by simulated botnets. Since in previous chapters the effectiveness of the attacks was not greatly increased by the botnet size, this comparison will use averaged values from all three botnets and use those for the comparison. Similar works has been performed by participants at the Network Research Lab (NRL) of the University of Texas Rio Grande Valley [24-36] nevertheless, this study will introduce findings for newer operating systems.

This comparison is performed to determine what form of advantages each operating system has in terms of performance and survivability against common DDoS attacks. All operating systems were implemented on Apple's Mac Pro Server (Mac Pro Mid2010). Applying identical hardware characteristics for all operating systems will provide clarity in these experiments as difference in hardware will not have a role on system performance. Furthermore, all operating systems used in this study feature all security updates released up to December 2018.

5.1 Comparative Evaluation under Distributed Denial of Service Attack

5.1.1 Layer 3 ICMP Ping Attack

As Ping attack traffic was introduced, it was observed that macOS High Sierra would limit the number of Echo requests received and the number of echo replies sent. Based on data collected macOS High Sierra began limiting the amount of request received at 500 Mbps to be approximately 600,000 per second. Meanwhile both Windows Server versions did not limit the number of requests received, but instead experienced dropped packets. The same number of packets is being sent per second at 500 Mbps macOS is receiving around 600,000 messages per second while both Windows Server versions reported approximately 500,000 messages per second. This is demonstrated on Figure 5.1 Additionally, while under attack both Windows Server releases would not transmit any echo reply messages while macOS would perform a maximum of 250 echo reply messages per second.

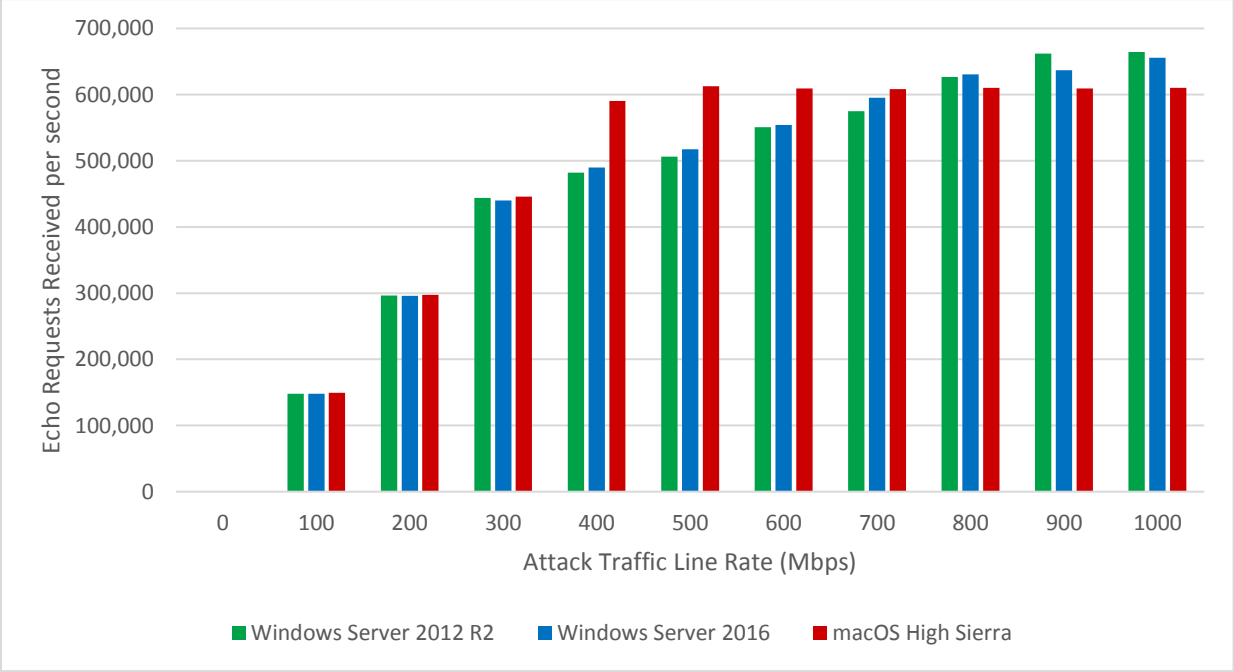


Figure 5.1 Echo Requests under Ping Attack

A baseline performance for HTTP transaction for both Windows Server was established at 3000 HTTP transactions per second. The baseline performance was attempted on macOS High Sierra but was unable to deliver. Therefore, the number of transactions was scaled down for macOS High Sierra to 600 transaction per second which would be maintained if no attack traffic was present. When attack traffic was introduced the baseline performance was mostly maintained up to 200 Mbps however, at 300 Mbps all platforms started losing connections and at 400 Mbps both Windows Servers lost a drastic amount of connections, being able to only perform 653.17 by Windows Server 2012 R2, 775.95 by Windows Server 2016 and 440.19 by macOS High Sierra. Higher attack traffic loads caused the operating systems to experience a gradual decline in connections where most were lost at 1 Gbps. The connectivity for all operating systems in these experiments can be observed on Figure 5.2.

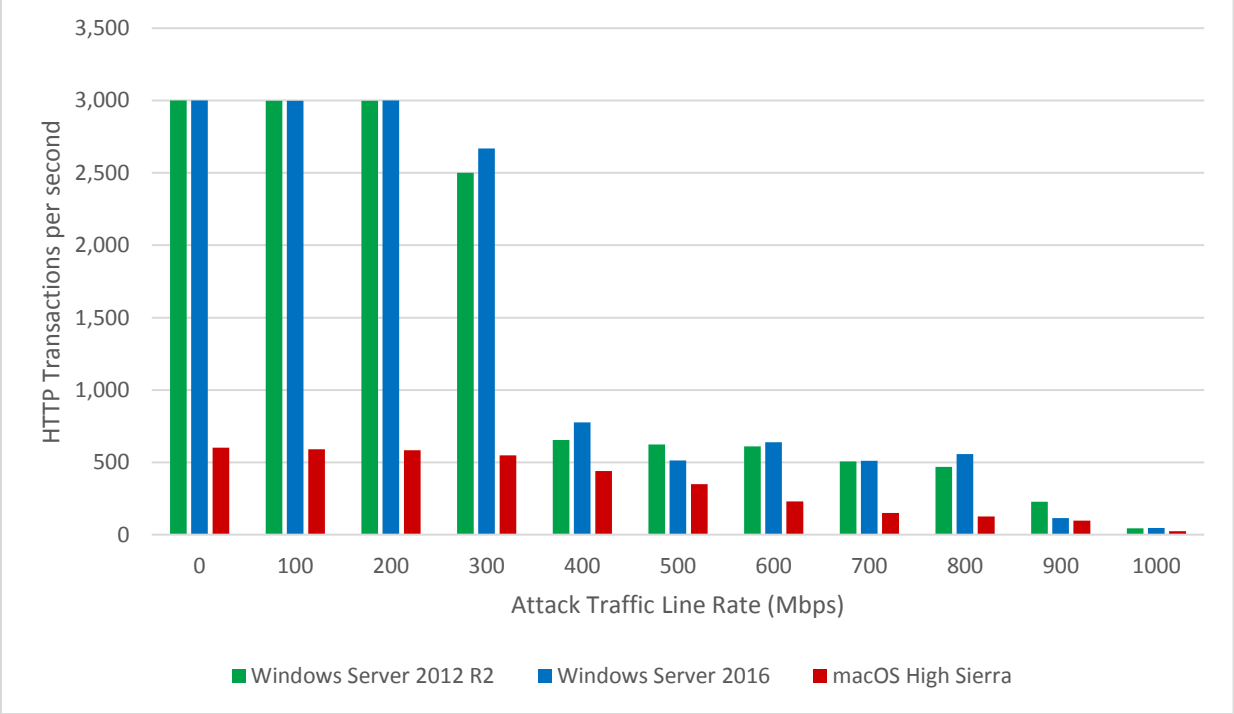


Figure 5.2 HTTP Transactions under Ping Attack

Memory consumption during Ping attacks was kept at a minimal for both Windows Server versions, with a slight increase at higher loads where a maximum of roughly 50 MB and 100 MB were consumed by the Ping attacks on Windows Server 2012 R2 and Windows Server 2016 respectively. Conversely, macOS High Sierra demonstrated a higher amount of memory consumption than both Windows Server versions at baseline performance. As Microsoft's server platforms consumed less than 1 GB of memory macOS High Sierra established a baseline at nearly 3 GB of memory consumed. Attack traffic loads caused this number to rise as the intensity of the attack did. Between 400 Mbps and 600 Mbps we see the consumption of memory increase at an accelerated rate, while higher attack traffic loads did not seem to be affect the consumption of memory. The consumption of memory was limited at higher attack traffic loads as a consequence of the limitations imposed on the number of echo request received as demonstrated by Figure 5.3.

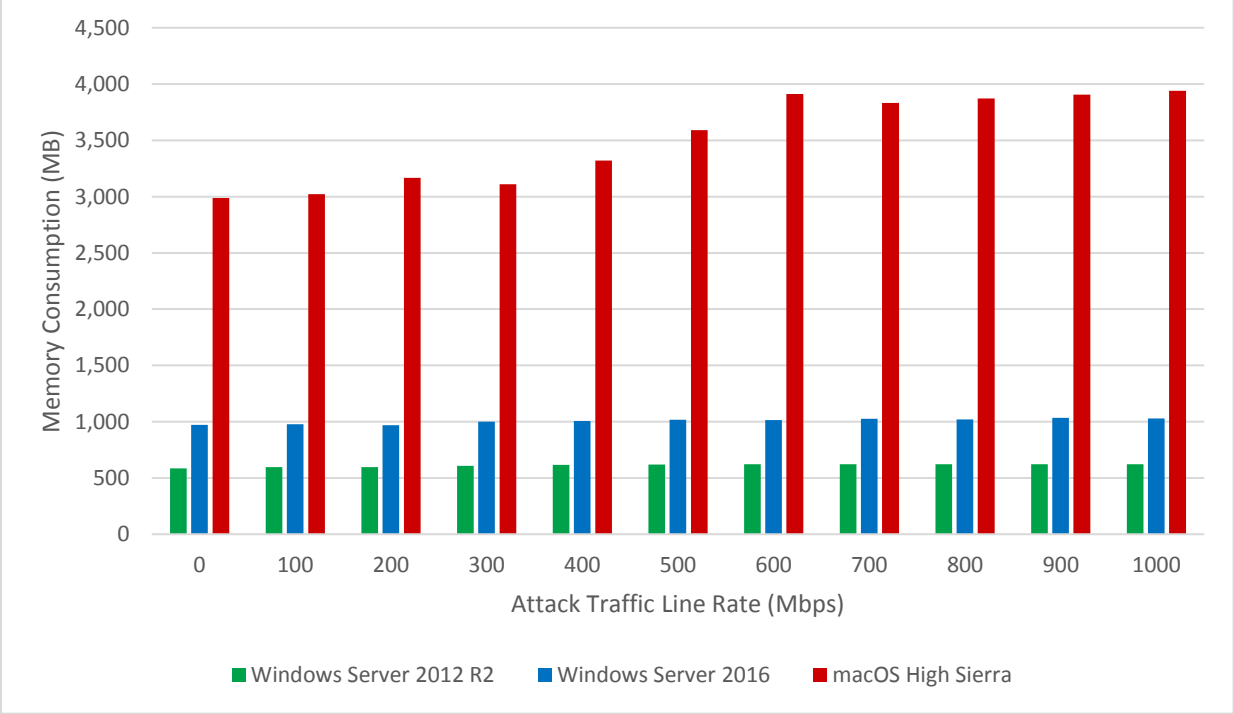


Figure 5.3 Memory Consumption under Ping Attack

Both Windows Server version had a baseline CPU consumption of roughly 6% while macOS High Sierra reported a consumption of roughly 9%. The percentage of CPU consumption by both Windows Server version was at its highest under a Ping attack traffic load of 200 Mbps with a consumption if approximately 18%. Higher attack traffic loads saw this consumption diminished starting at 300 Mbps and increasing once more along with the intensity of the attack. Meanwhile macOS High Sierra experienced an increase in CPU consumption as Ping attack traffic load increased and reached it. Maximum CPU consumption was found to be nearly 21% at attack traffic loads of 400 Mbps and 500 Mbps. Attack traffic loads beyond 500 Mbps triggered a decrease in CPU consumption. The decrease in CPU consumption can be attributed to the processing power no longer allocated for the HTTP transaction which were lost at higher attack traffic loads, and the limitation of ICMP echo request messages received. The total amount of CPU utilization is demonstrated by Figure 5.4.

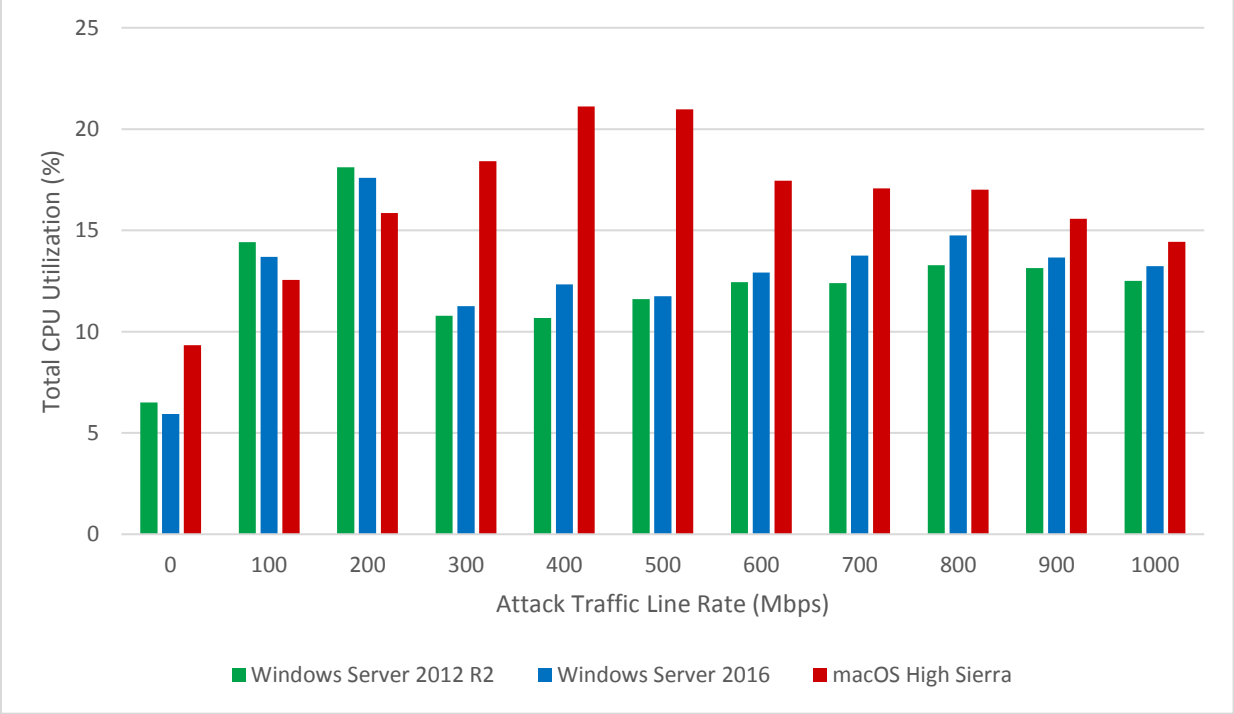


Figure 5.4 Total CPU Utilization under Ping Attack

Under Ping attack traffic macOS High Sierra was able to maintain lower CPU temperatures than both Windows Server platforms despite having a higher amount of CPU utilization. As attack traffic was intensified Apple's platform underwent an increase in temperature proportional to the amount of CPU utilization experienced where it was lowered at higher attack traffic loads as a result in the limitation of ICMP echo requests received and HTTP GET requests no longer being processed as a consequence of the attack. The highest temperature experienced by macOS High Sierra was of 56.75 °C at 500 Mbps of attack traffic. On the other hand, Microsoft's platforms experienced a maximum temperature of 71.09 °C by Windows Server 2016 and 69.08 °C by Windows Server 2012 R2. These temperatures were triggered at 200 Mbps when CPU consumption was at is highest. Higher traffic loads saw temperature decline but were maintained between 50 °C and 60°C by both server operating systems. CPU temperatures are demonstrated by Figure 5.5.

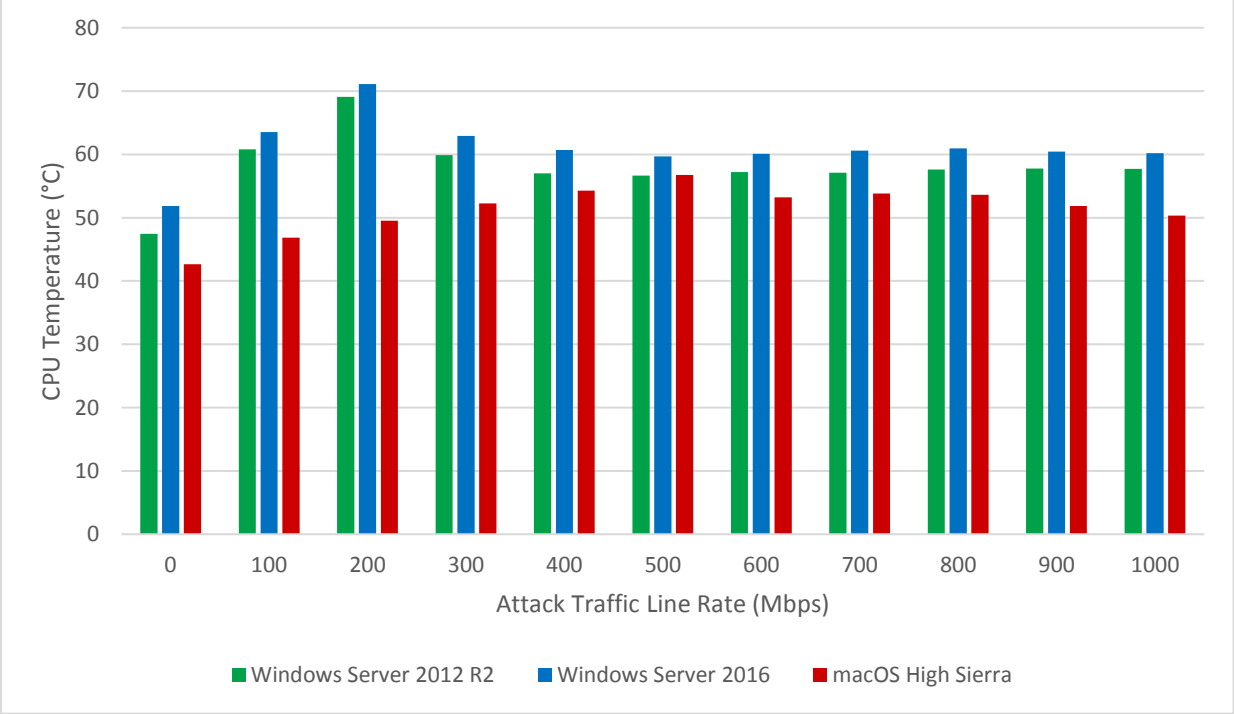


Figure 5.5 CPU Temperature under Ping Attack

5.1.2 Layer 3 ICMP Smurf Attack

Under Smurf attack traffic it was observed that all three operating systems would limit the amount of echo request received. This trend can be observed at traffic loads in the range of 400 Mbps and 1 Gbps where Windows Server 2012 R2 only received up to approximately 450,000 ICMP echo reply messages, while Windows Server 2016 only accepted a maximum of 500,000 at 400 Mbps and approximately 470,000 at higher attack traffic loads. Lastly macOS High Sierra limited the amount of received ICMP echo requests at 600,000 messages per second. By limiting the amount of echo request messages received the operating systems will be able to save valuable resources. The rate at which ICMP echo request messages are received per second is demonstrated by Figure 5.6.

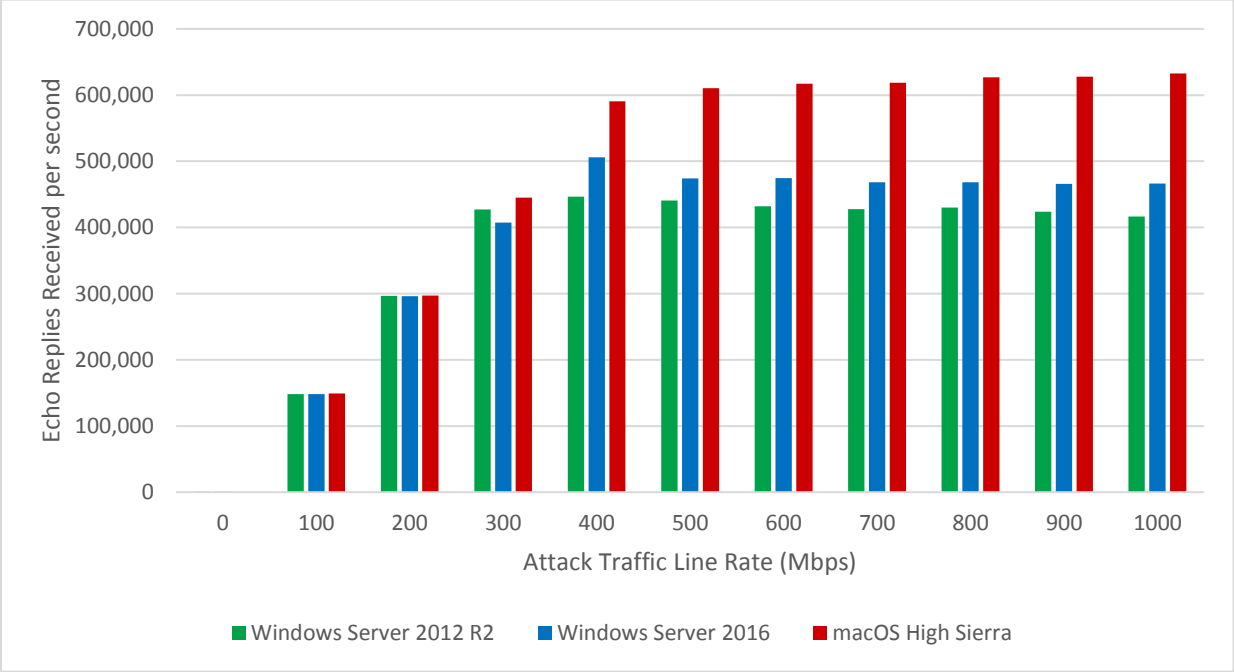


Figure 5.6 Echo Replies under Smurf Attack

Smurf attack traffic had a highly adverse effect on the baseline performance for HTTP Transaction per second achieved by the operating systems. In the data collected its demonstrated

that the baseline performance for all three operating systems is maintained up to 200 Mbps of attack traffic. Higher attack traffic loads were extremely detrimental to the baseline performance established by both Windows Server operating systems where the 2016 version was able to perform only 715.75 transactions per second while the 2012 R2 version only performed 324.43 transaction per second. This type of DDoS attack was more impactful for Windows Server 2012 R2 as it lost most of its connections at 400 Mbps of attack traffic. Traffic loads in the range of 400 Mbps and 1 Gbps saw macOS High Sierra and Windows Server 2016 provide similar performance in terms of HTTP GET requests processed. The connectivity of the three operating systems is demonstrated in Figure 5.7.

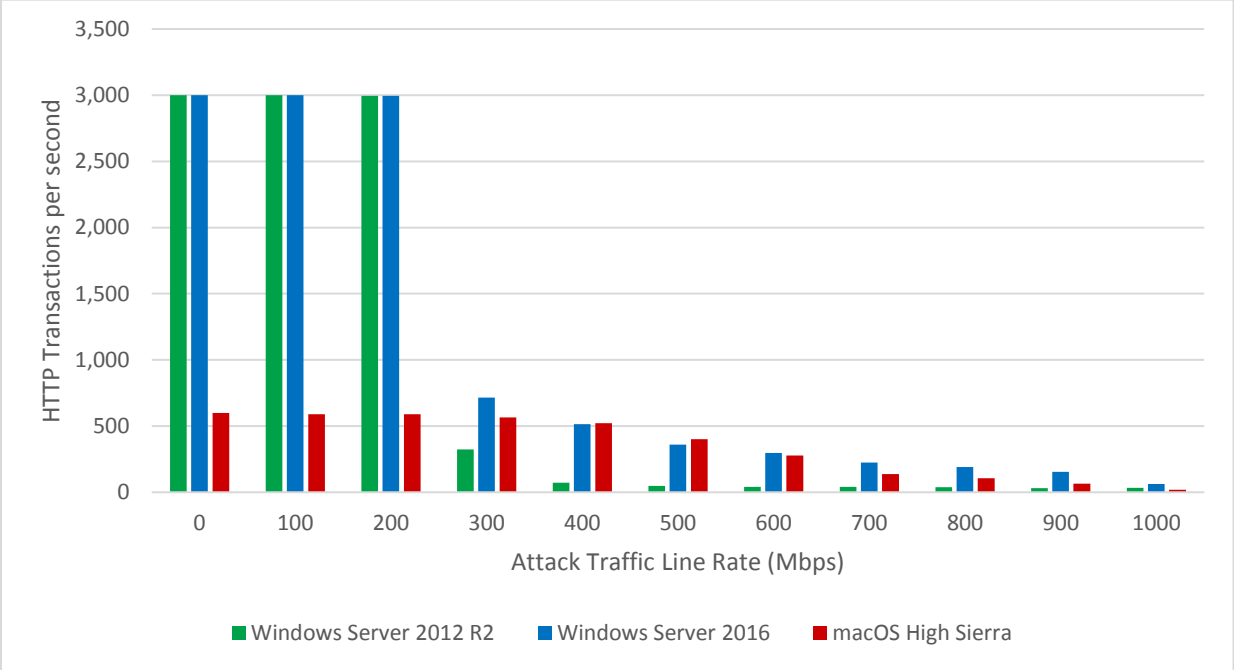


Figure 5.7 HTTP Transactions under Smurf Attack

Smurf attacks were not impactful to the memory consumption on Windows Server environments however, macOS High Sierra experienced a maximum memory consumption of 4.15 GB of RAM memory at 700 Mbps being 1.15 GB higher than baseline performance. As

Microsoft's server platforms consumed less than 1 GB of memory macOS High Sierra established a baseline at nearly 3 GB of memory consumed. Attack traffic loads caused this number to rise as the intensity of the attack did. Between 400 Mbps and 700 Mbps we see the consumption of memory increase at an accelerated rate. Higher attack traffic loads did not seem to be affect the consumption of memory as much as 600 Mbps and 700 Mbps did. The consumption of memory was limited at higher attack traffic loads because of the limitations imposed on the number of echo replies received. The total amount of memory consumed is demonstrated by Figure 5.8.

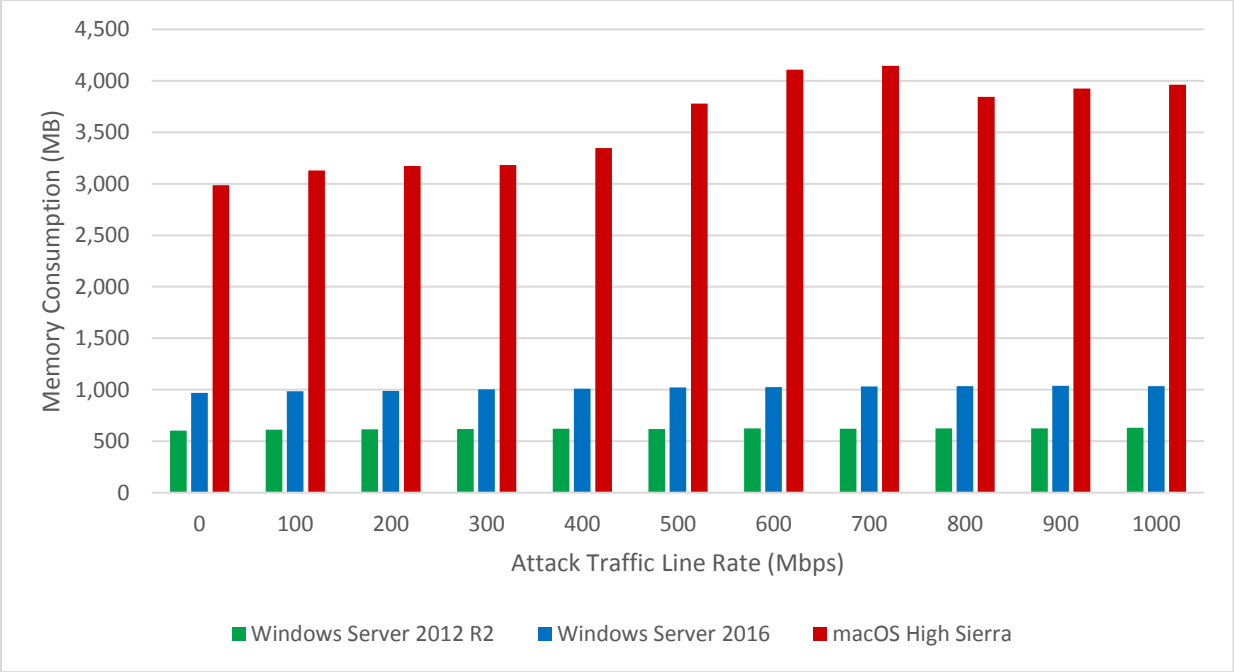


Figure 5.8 Memory Consumption under Smurf Attack

The baseline performance of the all three operating systems was compromised as Smurf attack traffic began reaching the server. In attack traffic load ranges of 100 Mbps to 400 Mbps had CPU Utilization for all platforms increase gradually. Under Smurf attack traffic Windows Server 2016 experienced the highest amount of CPU utilization of all three operating systems.

This was experienced at 400 Mbps of attack traffic where CPU consumption was 48.53%. Moreover, Windows Server 2012 R2 had a maximum CPU utilization of 40.95% at 500 Mbps and macOS High Sierra had a maximum CPU utilization of 21.02% at 400 Mbps. Attack traffic loads in the range of 400 Mbps to 1Gbps had the number of received ICMP echo replies limited at these points which kept CPU utilization from escalating with higher attack traffic loads. Additionally, macOS experienced a drop in CPU utilization after 400 Mbps where attack traffic caused HTTP GET requests to stop being processed by the OS. Total CPU utilization is demonstrated thru Figure 5.9.

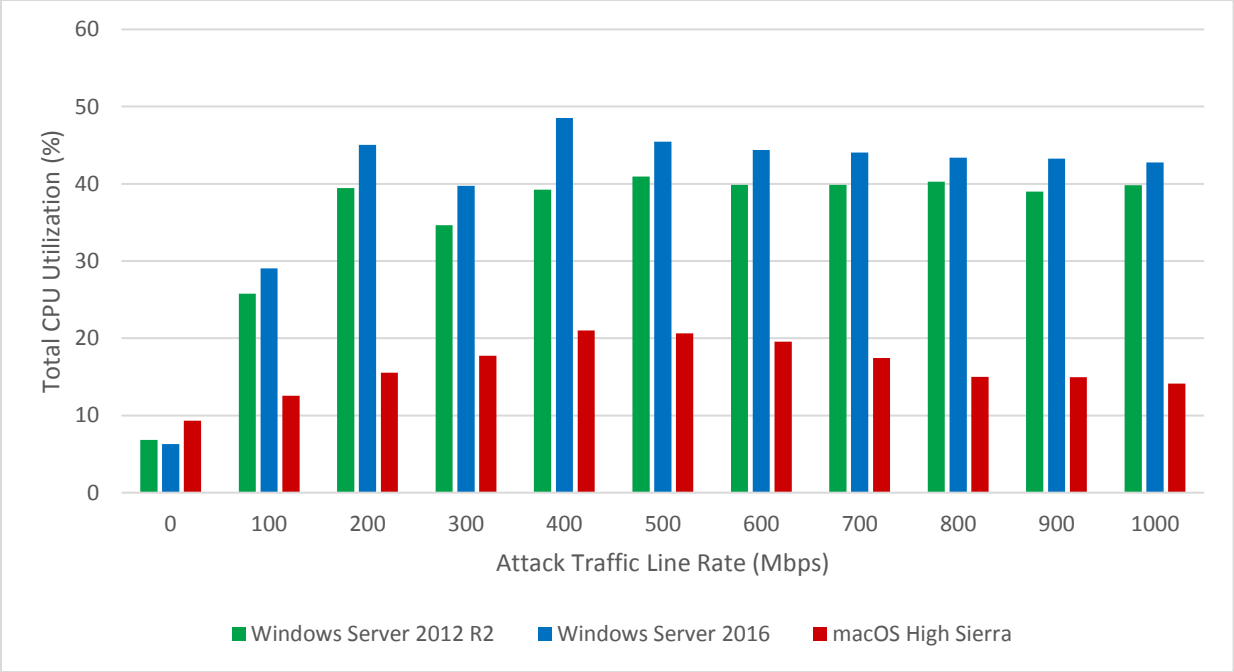


Figure 5.9 Total CPU Utilization under Smurf Attack

As for CPU Temperature, macOS High Sierra was able to maintain lower temperatures than both Windows Server operating systems while under Smurf attacks. The lower temperatures collected are related directly to the amount of CPU power consumed. Since macOS High Sierra had lower CPU utilization than both Windows Server environments it would also have lower

temperatures. The escalation of attack traffic on Apple's platform increased CPU temperature proportionally to the amount of CPU utilization. The highest attack traffic loads lowered CPU utilization as a result in the limitation of ICMP echo replies received and HTTP GET requests no longer being processed because of the attack. The highest temperature experienced by macOS High Sierra was of 56.79 °C at 600 Mbps of attack traffic however most temperatures in this platform were kept in between the range of 45 °C to 55 °C. On the other hand, Microsoft's platforms experienced a maximum temperature of 91.74 °C by Windows Server 2016 and 88.98 °C by Windows Server 2012 R2 at attack traffic loads of 400 Mbps and 500 Mbps respectively. Attack traffic loads in the range of 400 Mbps to 1 Gbps triggered higher CPU utilization which kept temperatures close to 90 °C. This temperature is considered to be dangerous as physical damage can occur to the processor. CPU temperatures for all three platforms is demonstrated by Figure 5.10.

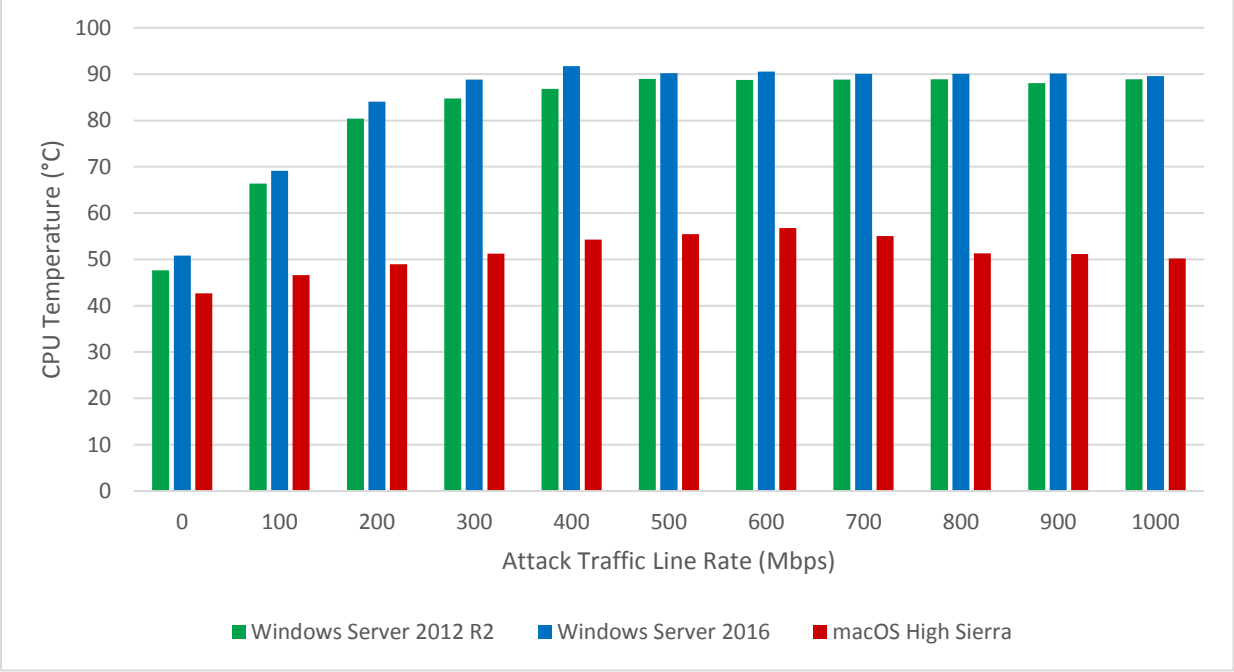


Figure 5.10 CPU Temperature under Smurf Attack

5.1.3 Layer 4 TCP SYN Flood Attack

The most detrimental attack to HTTP connections in this thesis was TCP SYN Flooding attack. Network connectivity can be observed in Figure 5.11.

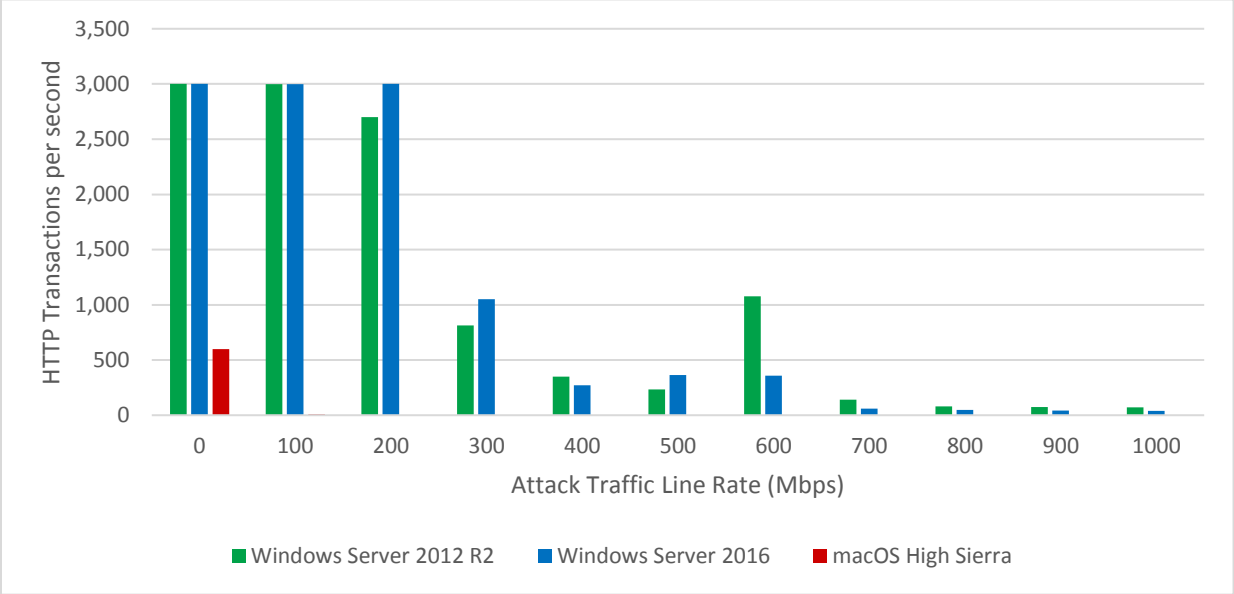


Figure 5.11 HTTP Transactions under TCP SYN Flood

Under TCP SYN flooding traffic the baseline performance established by all three operating systems was greatly impacted. macOS High Sierra was affected the most by this attack. As TCP SYN traffic reached macOS High Sierra, all HTTP connections were lost, this behavior was maintained for all traffic loads. Windows Server 2012 R2 was not able to maintain its baseline performance of 3000 HTTP transactions past 100 Mbps of attack traffic. It suffered a 10% drop at 200 Mbps and a 72.86% drop at 300 Mbps. Attack traffic of higher intensity caused most connections to be dropped. However, it was observed through all experimental trials that a small portion of HTTP Transactions was recovered at 600 Mbps. It is unclear why this recovery is present considering CPU utilization is higher at this load. Further experimentation and insight into defense mechanism and background processes implemented by Microsoft might address this

anomaly. Similarly, Windows Server 2016 was able to maintain its baseline performance up to 200 Mbps of attack traffic. Subsequently, an attack traffic load of 300 Mbps triggered a 65% loss in HTTP transactions while higher loads cause a gradual decline from roughly 300 HTTP transaction in the range of 400 Mbps to 600 Mbps to nearly none in the range of 700 Mbps to 1 Gbps.

TCP SYN flooding attack traffic had the lowest effect on memory consumption. Under TCP SYN attack traffic it was observed that memory consumption for all operating systems gradually increased in small steps as attack traffic intensity was increased. While under attack memory consumption for both Windows Server releases was kept below 1 GB. Windows Server 2012 R2 consumed a maximum of 609 MB, whereas Windows Server 2016 consumed a maximum of 989 MB of memory. On the other hand, maximum memory consumption for macOS High Sierra was 378.94 MB higher than baseline performance reaching a consumption of 3.37 GB. The total amount of memory consumed is demonstrated by Figure 5.12.

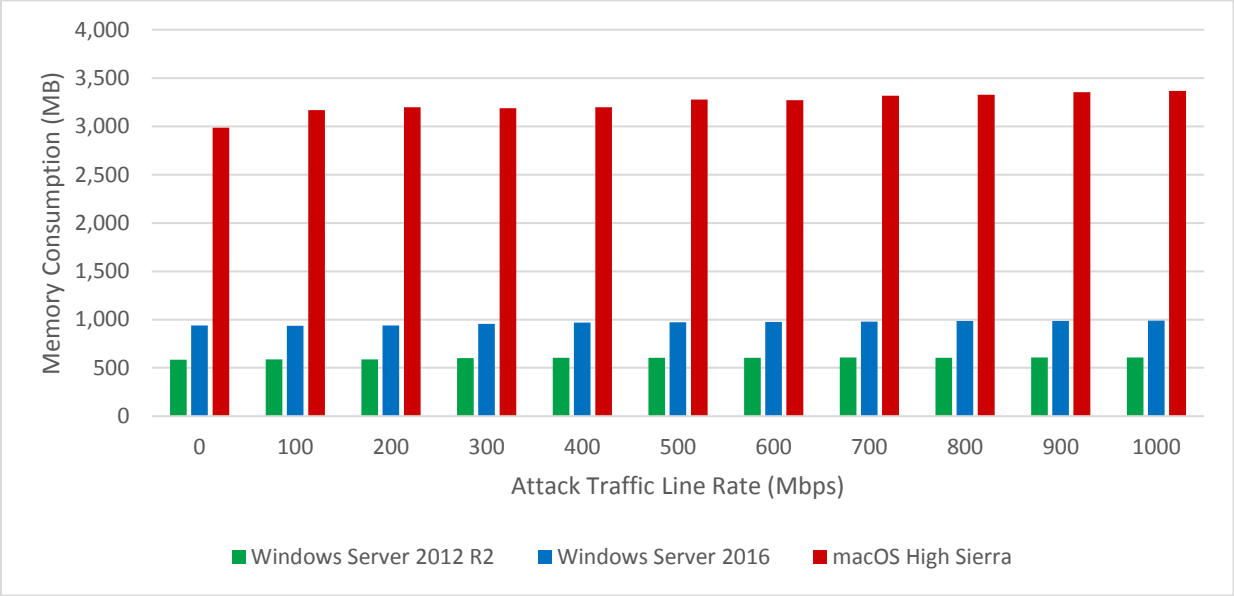


Figure 5.12 Memory Consumption under TCP SYN Flood

During experimental trials it was demonstrated that the CPU Utilization of macOS High Sierra was lower than that achieved by both Windows Server environments. Thus it can be said that macOS High Sierra provided higher performance than both Windows Server environments at the cost of active connections as demonstrated in Figure 5.13.

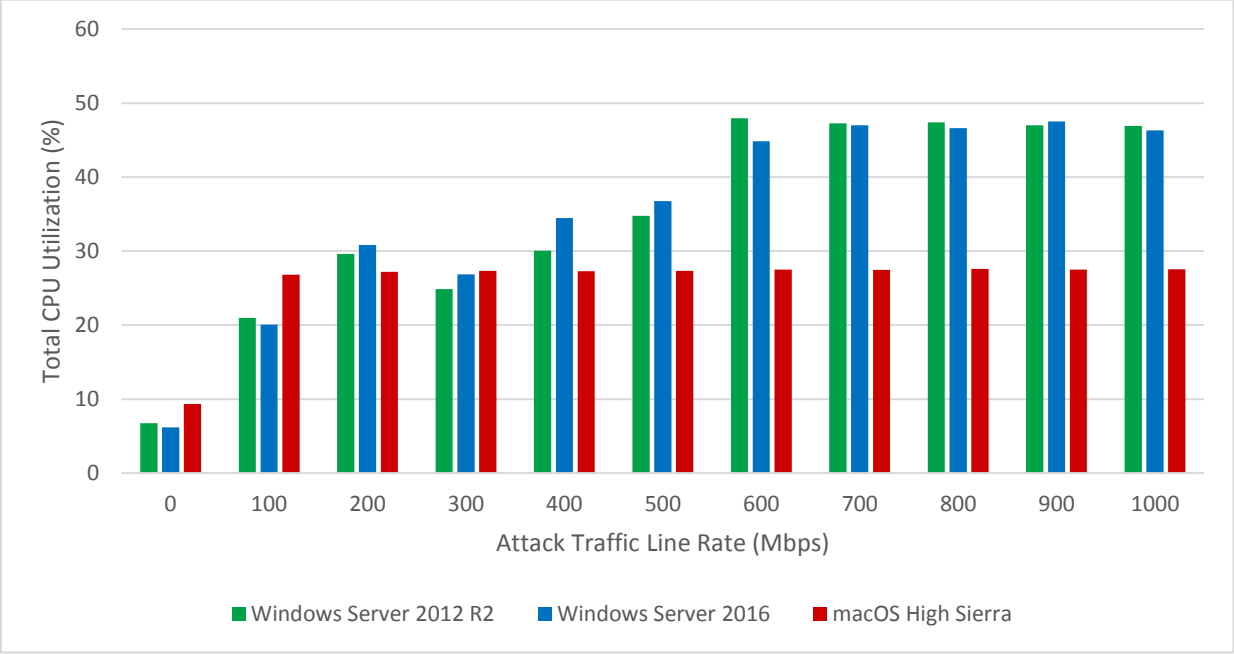


Figure 5.13 Total CPU Utilization under TCP SYN Flood

The baseline performance of macOS High Sierra was compromised starting at 100 Mbps of TCP SYN attack traffic causing a CPU utilization of nearly 27% which was maintained across all traffic loads. CPU consumption did not rise for macOS High Sierra at higher attack traffic loads since within the transmission control block the number of active and half open connections is limited in this environment, this is evident when observing the decline in HTTP connections which depend on an initial TCP connection. Both Windows Server versions performed similar while under attack. As TCP SYN attack traffic began reaching the server CPU consumption rose gradually and was limited in the range of 600 Mbps to 1 Gbps where consumption was

approximately 47% for traffic loads in this range. The limitation in consumption can be attribute to Hyper threading technology. Hyper threading technology splits the four physical cores within the processor into eight logical/virtual cores allowing for additional threads to be processed at the same. However, applications must be designed to take advantage this technology, otherwise four of the eight logical cores will be interpret as the only physical cores resulting in CPU consumption lower than 50%. It is possible for more than 50% of the CPU to be utilized if other applications that support this technology are running parallel to the attack.

Since macOS High Sierra maintained the same CPU consumption throughout the attack, the temperatures collected for CPU reported nearly the same 70 °C experienced for all attack traffic loads starting at 200 Mbps. In contrast, Microsoft's platforms experienced an increase in temperature which gradually escalated. Attack traffic loads in the range of 200 to 500 Mbps saw temperatures rise and be kept in the range of 70 °C and 80 °C. Meanwhile maximum temperature in the range of 90 °C to 93 °C were reached as attack traffic loads in the range of 700 Mbps and 1 Gbps were received by the targeted server. Temperatures in this range are considered to be dangerous as physical damage can occur to the processor due to the heat being too intense for the inner components of the processor. CPU temperatures for all three platforms is demonstrated by Figure 5.14.

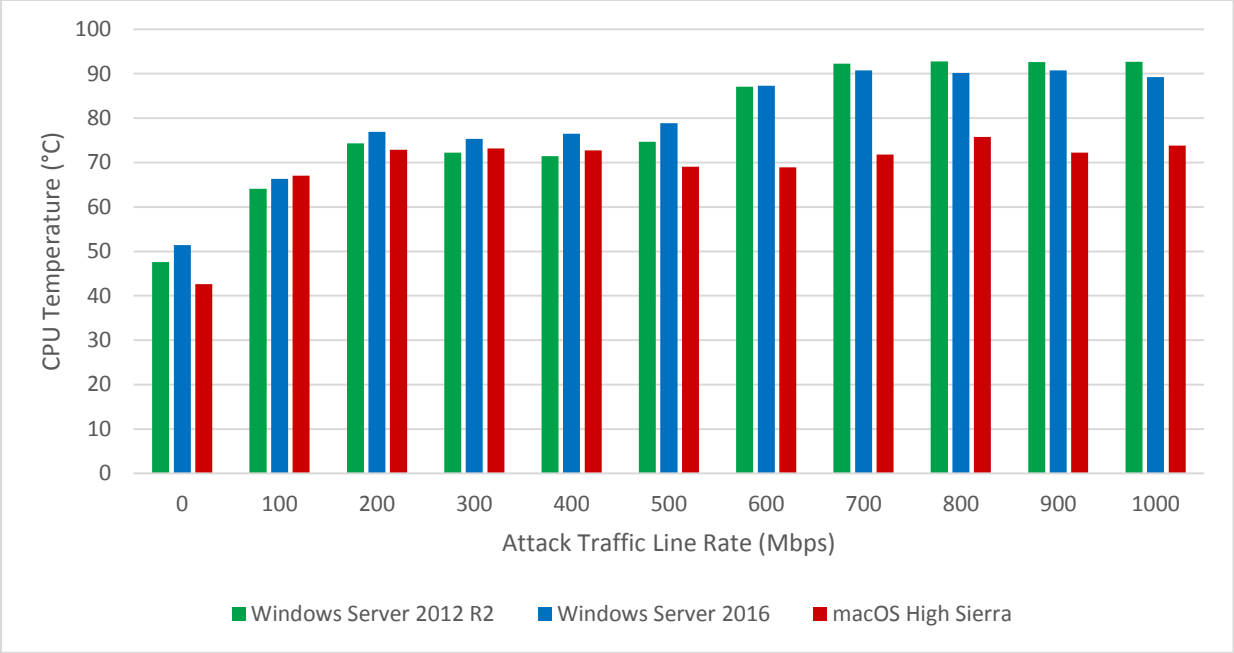


Figure 5.14 CPU Temperature under TCP SYN Flood

5.1.4 Layer 4 UDP Flood Attack

UDP flood attack traffic in this experiments targeted a port not being listened to by the operating system. Based on data collected it was observed that UDP flood attack traffic was limited by macOS High Sierra where the number of UDP datagrams received would hardly surpass 600,000 datagrams per second. Furthermore, macOS High Sierra began limiting the amount of UDP datagrams received at 500 Mbps while transmitting back a maximum of 250 destination unreachable messages for all attack traffic loads. On the other hand, both Windows Server versions began limiting the number of datagrams received at 800 Mbps where approximately no more than 640,000 messages would be received. Additionally, both Windows Server version experienced dropped attack traffic packets. This is evident when compared to macOS High Sierra. Moreover, both Windows Server environments refused to transmit any

ICMP destination unreachable packets. The rate at which UDP datagrams are received per second is demonstrated by Figure 5.15.

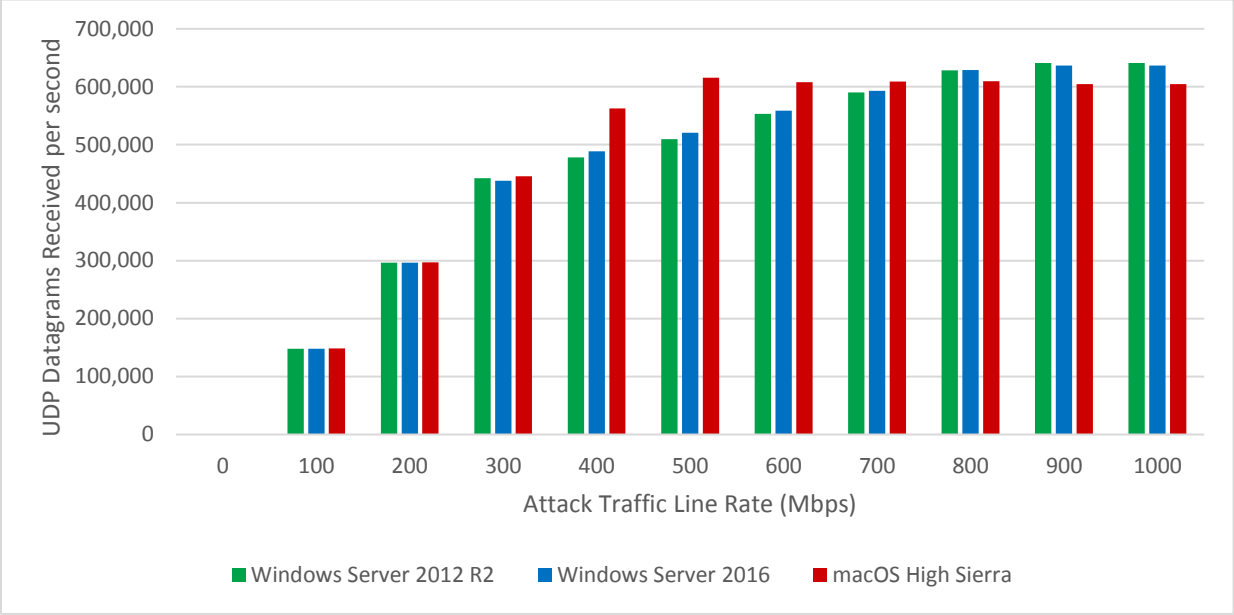


Figure 5.15 UDP Datagrams received/sec under UDP Flood

The Operating systems in this experiment while under attack do not comply with RFC 1122 which states that an ICMP destination unreachable message must be transmitted if an UDP datagram is received and the port is not being listened to or is closed. These behavior help mitigate the attack as processing of the ICMP message is not performed.

Baseline performance for HTTP transactions was mostly maintained by all three operating systems up to 200 Mbps of attack traffic. Windows Server 2012 R2 seems to perform better then Windows Server 2016. At 300 Mbps the 2012 R2 release lost 31.64% of all connection while the 2016 release lost 60.6% of all connections. However both server operating systems still outperformed macOS High Sierra. Attack traffic loads in the range of 400 Mbps to 1 Gbps cause all three operating systems to lose all connections at a gradual rate, where

Windows Server 2012 R2 outperformed Windows Server 2016 which outperformed macOS High Sierra. Network connectivity for all three operating systems is displayed on Figure 5.16.

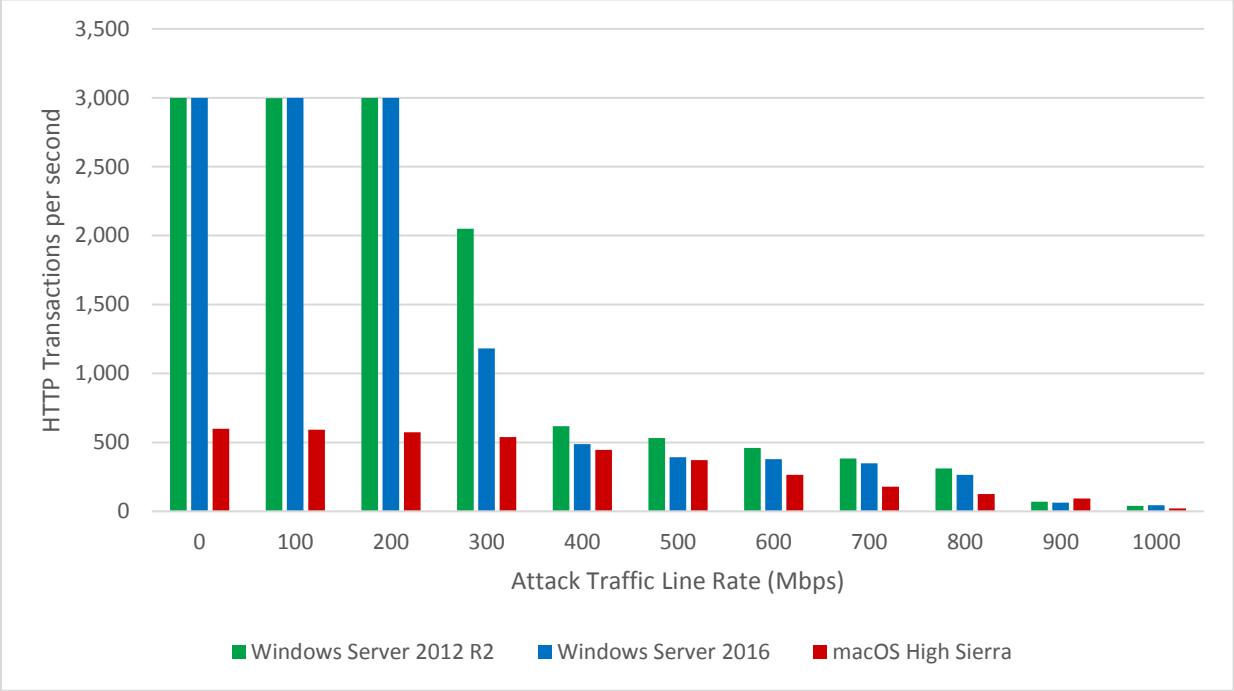


Figure 5.16 HTTP Transactions/sec under UDP Flood

The amount of memory consumed by Microsoft's platforms was maintained at a minimum where Windows Server 2016 consumed 1036 MB under 1Gbps of attack traffic while Windows Server 2012 R2 consumed a maximum of 643.41 MB at 900 Mbps. Both Windows Server versions, consumed 50 MB more than they were at baseline performance. Meanwhile, macOS High Sierra consumed more memory than both Windows Server versions. As attack traffic load escalated the amount of memory consumed did as well. macOS High Sierra has a baseline performance of 2.99 GB which at its worst was increased to 3.74 GB at 500 Mbps of attack traffic. In these attacks apple's platform experienced an escalation in memory consumed between 0 Mbps and 500 Mbps. The amount of memory consumed declined slightly at higher attack traffic loads since the number UDP datagrams was limited and several HTTP GET

requests stopped being processed as a result of the attack. Total memory consumption of each operating system is demonstrated by Figure 5.17.

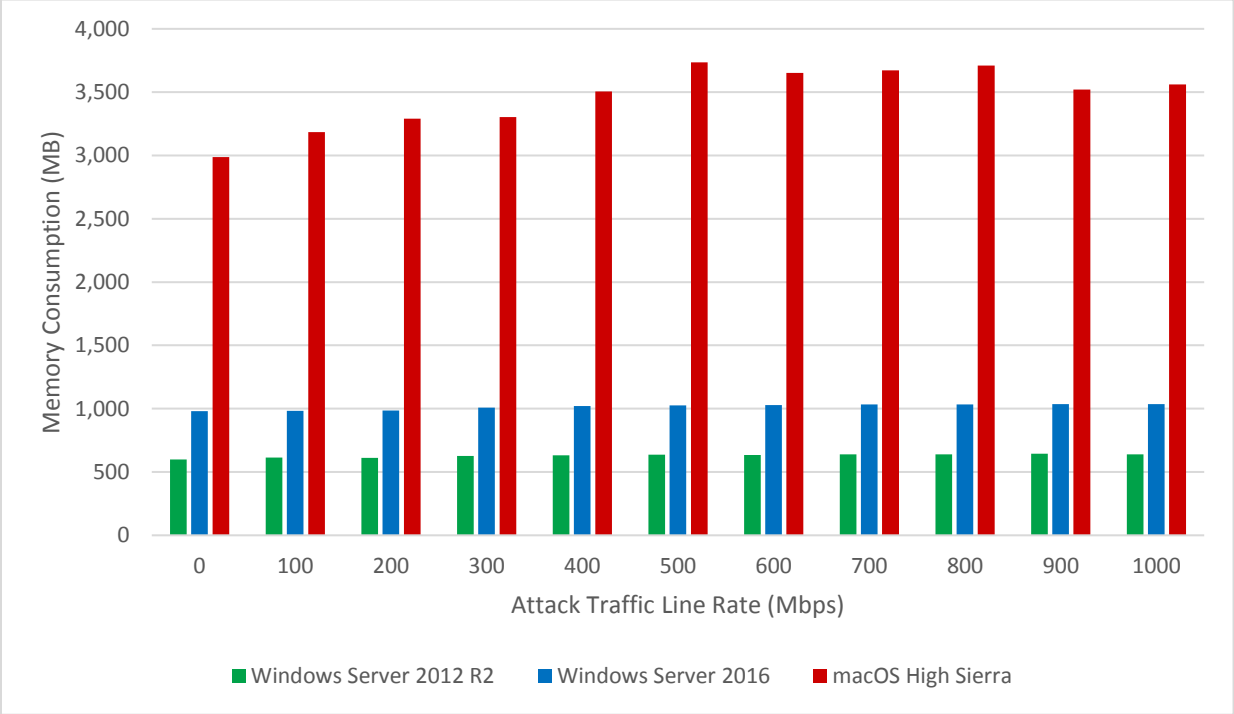


Figure 5.17 Memory Consumption under UDP Flood

Under UDP flood attack traffic it is observed that both Windows platforms consumed more CPU time than macOS High Sierra. Both of Microsoft's platforms underwent their highest CPU utilization at 200 Mbps of UDP flood attack traffic. Windows Server 2012 R2 consumed 24.65% CPU time while Windows Server 2016 consumed 27.39% at 200 Mbps. Higher attack traffic loads caused this number to decrease at 300 Mbps and escalate once more as the attack traffic load increased. Between 600 Mbps and 1 Gbps CPU utilization reached a threshold of roughly 23% to 24% utilization. As for macOS High Sierra, baseline performance was compromised for CPU utilization as traffic load increased between the ranges of 100 Mbps to 400 Mbps reaching a maximum CPU utilization of 19.84%. High attack traffic loads in the range

of 500 Mbps to 1 Gbps caused a decline in CPU consumption attributed to a limit on UDP datagrams received and HTTP GET requests no longer being processed. The total amount of CPU utilization is demonstrated by Figure 5.18.

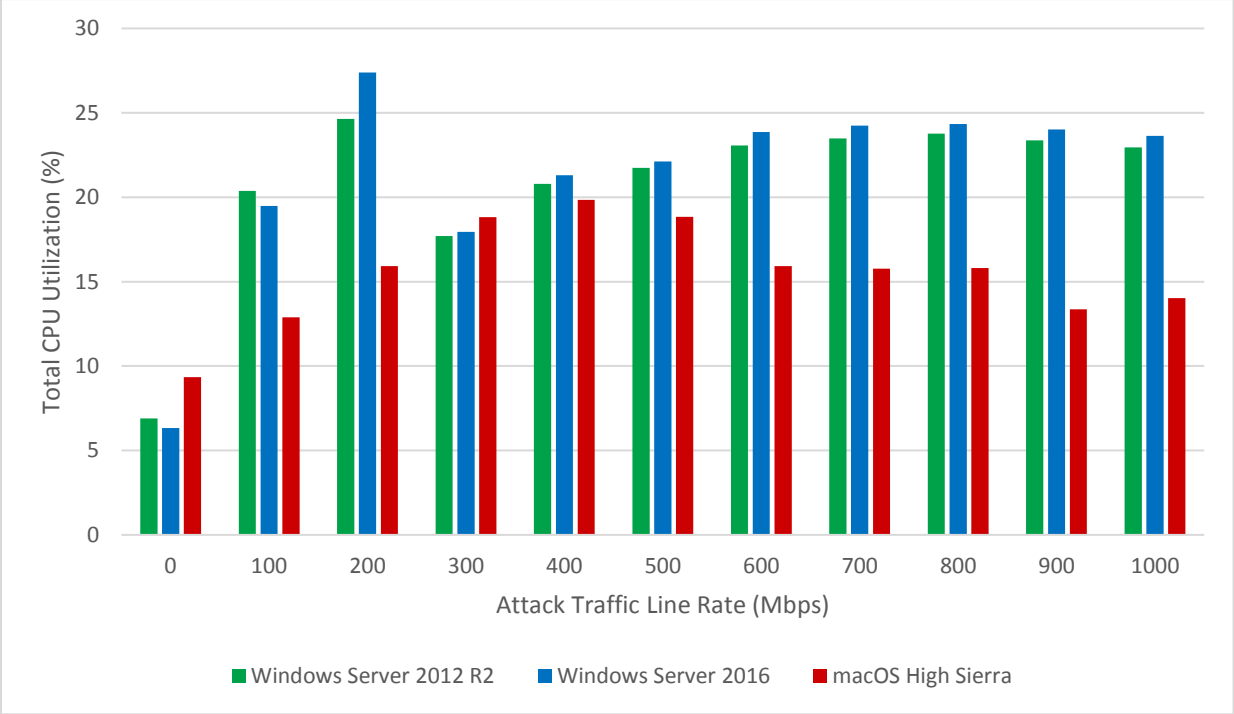


Figure 5.18 Total CPU Utilization under UDP Flood

As demonstrated by Figure 5.18 macOS High Sierra had lower CPU utilization than both Windows environments. Since CPU utilization is directly related to CPU temperature lower temperatures were expected and delivered by Apple's platform. CPU temperatures for the three operating systems are displayed on Figure 5.19.

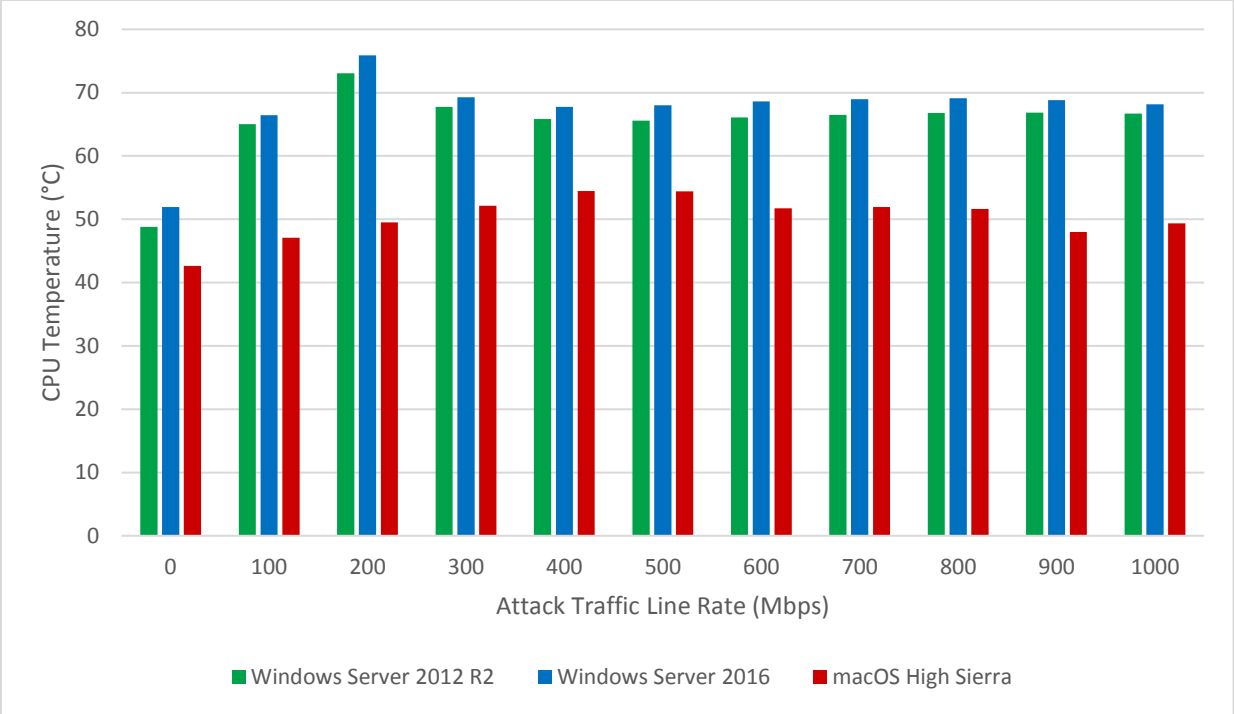


Figure 5.19 CPU Temperature under UDP Flood

CPU temperature on macOS increased as attack traffic load increased between the ranges of 100 Mbps to 400 Mbps reaching a maximum temperature of 54.47 °C while higher attack traffic loads in the range of 500 Mbps to 1 Gbps lowered CPU temperature since UDP datagrams received had been limited and HTTP GET requests could no longer be processed under this loads. Furthermore, at 200 Mbps of attack traffic Windows Server 2012 R2 attained a maximum temperature of 73.04 °C while Windows Server 2016 reached 75.91 °C. Higher attack traffic loads on both server operating systems triggered temperatures in the range of 66 °C to 69 °C.

5.2 Chapter Summary

This Chapter brings together all data collected for all three operating systems. Since in previous Chapters it was found out that the botnet had little to no effect on the performance given by the operating systems deployed on Apple's Mac Pro (Mid 2010) namely Apple's

macOS High Sierra 10.13.6, Microsoft's Windows Server 2012 R2 and Windows Server 2016 was averaged out to visualize the effect this attacks would have more commonly.

The three operating systems were exposed to four distinct Distributed Denial of Service attacks to be precise Ping attack, Smurf attack, TCP SYN flooding, and UDP Flooding. Where their impact on the operating system performance was measured utilizing Microsoft's performance monitor and CoreTemp [96-97] for Windows environments, while macOS performance was measured using several terminal commands and TG Pro for temperature readings [109]. Moreover, Microsoft Excel and MATLAB were used to interpret the data acquired, by translating raw data from .csv files into graphs [98-99]. The graphs generated with this tools serve for the presentation of data in a clean, concise manner which helps readers visualize and compare the performance achieved by all three operating systems as they undergo different Distributed Denial of Service attacks.

In this experiments it was found that memory consumption for both Windows Server environments to negligible with a low increase in consumption. Conversely macOS High Sierra was affected by the attacks causing consumption of nearly 500 MB of RAM up to 1.5 GB.

According to data collected macOS High Sierra would limit incoming attack traffic at a load of 500 Mbps where UDP datagrams, ICMP echo reply and requests messages were limited to 600,000 per second. Meanwhile both Windows platforms would experience drop packets and seemed to limit them at 800 Mbps to approximately 650,000 per second while a lower limit was imposed on received echo replies of approximately 500,000 per second which was evident at 400 Mbps. By limiting incoming attack traffic CPU consumption was prevented from rising and on macOS High Sierra would even decline as HTTP transaction became lost due to attack traffic saturating the network card.

Smurf attacks and TCP SYN flooding attacks were the most damaging attack implemented in this thesis. With high CPU utilization the temperature of the CPU increased and for TCP SYN it would reach dangerous levels on Windows environments. Moreover, on Windows environments Smurf attacks would consume a high amount of CPU time at any load while TCP SYN traffic required a higher load of attack traffic but would consume more CPU time than Smurf attacks did. Conversely, the CPU utilization of macOS High Sierra was not impacted as severely as it did on Windows environments. Apple's platform mitigated the effect of this attack by limiting the incoming attack traffic. While preventing CPU utilization from escalating some network connectivity was sacrificed to this end. Furthermore, Ping attacks and UDP flooding attacks had lower CPU utilization barely reaching or getting close to 20% usage, which can be attributed by limits imposed on egressing traffic and the limitation or refusal to reply to Ping requests or transmitting ICMP destination unreachable messages.

The most dangerous attack to network connectivity in this experiments was TCP SYN flooding as half open connection flood the transmission control block thus preventing new connections from being opened. This attack caused macOS High Sierra to lose all of its HTTP connections almost immediately meanwhile both Windows server environments lost more connections more connections at different attack traffic loads than with any other attack.

CHAPTER VI

CONCLUSION

The security and overall system performance of Microsoft's Windows Server 2012 R2, Windows Server 2016 and Apple's macOS High Sierra with all system updates released up to December 2018 were evaluated against four different Distributed Denial of Service attacks, namely Ping attacks, Smurf attacks, TCP SYN flooding and UDP flooding. All four attacks were performed while simulating three different botnet sizes replicating compromised Class A, B, and C networks. Based on data acquired little to no impact was created by the botnet sizes. The extent of the impact would go as far as increasing CPU consumption by 1% to 5% which is not of much importance since higher attack traffic loads can increase consumption and diminish available bandwidth and system resources with ease.

As experiments were performed not a single system crash was experienced through all trials performed. This can be attributed to inclusion of hyper threading technology and the mitigation tactics implemented by the operating systems deployed in this thesis.

At baseline performance both Windows platforms excelled at maintaining a high amount of HTTP connections which allowed for 3000 transactions per second to be processed while Apple's platform struggled to provide more than 600 HTTP transactions per second. macOS High Sierra was unable to perform more of these transactions as a consequence of limitation in network traffic. Additionally baseline performance for these systems saw low CPU consumption for all three platforms where Windows platforms would utilize roughly 7% CPU power and

macOS High Sierra would consume approximately 9%. Meanwhile memory consumption for Windows Server 2012 R2 would be of 600 MB, whereas Windows Server 2016 would consume 1 GB and macOS High Sierra would consume 3 GB. Out of the three platforms testes Windows 2012 R2 would provide the best system performance for the application of a simple web server.

In these experiments it was observed that the traffic sent by Ping attacks was not entirely accepted by any of the operating systems deployed on Apple's hardware. Windows Server 2012 R2 was found to limit the number of echo requests received at approximately 660,000 echo request. This behavior is shared by Windows Server 2016 with a similar limit while Apple's macOS High Sierra would limit the amount of incoming requests at 600,000 messages per second. Additionally no echo replies were transmitted by either of Microsoft's platforms while macOS High Sierra would transmit a maximum of 250 replies per second. This mitigation tactics allowed for the conservation of some resources on the server. Both Windows platforms outperformed macOS High Sierra in terms of CPU and memory consumption and network connectivity. Both server operating systems were found to consume less than 18% of all processing power while macOS High Sierra consumed nearly 22% at its worst. Additionally memory consumption was negligible for both of Microsoft platforms. By contrast macOS High Sierra consumed a higher amount of memory while under baseline performance and as attack traffic was introduced where usually 1 GB of memory would be consumed at higher intensities of the attack. More importantly macOS High Sierra platform was unable to deliver the same performance in network connectivity that its Windows counterparts. Not only was macOS High Sierra unable to provide the same number of HTTP transaction per second but under Ping attack it would also see its transactions diminished more than those on Windows 2012 R2 and 2016.

As Smurf attack traffic began reaching the targeted server it was observed that macOS High Sierra would limit the number of echo replies received in the same form it limited echo request messages where nearly 600,000 messages of either type of traffic were received. Meanwhile both Windows server environments imposed lower limits for incoming ICMP echo reply messages as approximately 470,000 messages were received at higher attack traffic loads for Windows Server 2016 and roughly 430,000 messages were received by Windows Server 2012 R2. Despite the limitation of ICMP echo reply messages both Windows Server versions had a high amount of CPU consumption where Windows Server 2016 experienced a CPU utilization in the ranges of 44% to 48% and CPU utilization for Windows Server 2012 was maintained around the range of 38% to 40%. Conversely the high amount of network limitation imposed by Apple allowed macOS High sierra to consume 21% CPU time or less. Although, regardless of limits in network traffic macOS High Sierra experienced a memory consumption of up to 1.1 GB approximately, while both Windows platforms only experienced a light increase when compared to baseline performance. Furthermore, HTTP transaction for Windows Server 2016 and macOS High Sierra were similar at higher attack traffic load in the range of 400 Mbps to 1 Gbps whereas Windows Server 2012 was not able to keep up after 400 Mbps of attack traffic. One of the more concerning effects of this type of DDoS attack is the rise in temperature experienced by both Windows Server versions which reached nearly 90 °C throughout most of the attack.

When it came to subjecting the operating systems against TCP SYN flooding, it was observed that memory consumption for both Windows environments to be negligible while macOS High Sierra experienced an increase of roughly 400 MB. It is worth mentioning that this form of DDoS attack consumed the least amount of memory on macOS High Sierra out of all

attacks implemented. However, under this attack macOS High Sierra lost all HTTP connections almost immediately, while both Windows Server versions struggled to maintain legitimate traffic after 300 Mbps of attack traffic was transmitted. As for CPU utilization macOS High Sierra experienced a utilization of roughly 27% through the entire length of the attack as the same amount of open connection were present for all traffic loads. Windows Servers on the other hand experienced a CPU consumption of nearly 47% at traffic loads in the range of 600 Mbps to 1 Gbps reaching CPU temperatures which surpassed 90 °C and putting the physical integrity of the processor at risk. Meanwhile temperatures for macOS High Sierra were maintained in the range of 68 °C and 72 °C.

Based on data collected macOS High Sierra limited the amount of received UDP datagrams per second to be nearly 600,000 while both Windows Server environment went no further than approximately 640,000 datagrams. All three operating system limited the amount of ICMP destination unreachable messages. Both Windows Server versions refused to transmit any of this messages while macOS High Sierra would only transmit 250. This filtering allowed all three operating systems to save system resources and kept CPU utilization within the range of 23% and 27% for both Windows Server versions. Meanwhile macOS High Sierra saw CPU utilization rise no more than 20% and decline at higher attack traffic loads where several HTTP connections were lost and stopped being processed. Lastly memory consumption for both Windows platforms was kept at a minimal while UDP memory consumption increased by nearly 700 MB.

Hyper threading technology offers the virtualization of a four physical core to provide eight logical ones. By providing the operating system with eight logical cores the number of thread that can be executed is doubled. However, applications must be designed with hyper

threading technology in mind otherwise the piece of software will take over only what it perceives as the physical cores resulting in maximum CPU utilization equal or lower than 50% by that application. According to data collected from these attacks it appears that none of the operating systems deployed utilized Hyper threading technology to process incoming network traffic which help mitigate the effect of these attacks.

All three platforms feature great strategies at defending from these attacks but based on data collected none of the operating systems are able to maintain connectivity as attack traffic increases. At the cost of connectivity, macOS High Sierra defends its processing power rather well. Hyper threading technology combined with the security features of macOS High Sierra make a good combination at limiting the impact of these attacks on processing power however, if a process in this environment are memory intensive the performance of macOS High Sierra will deteriorate greatly. Meanwhile both Windows Server version allow for higher amounts of network traffic to be processed with minimal impact to system resources. However, current default firewall settings can permit CPU consumption to rise recklessly and without Hyper threading technology system performance can be severely compromised under TCP SYN flooding and Smurf attacks.

REFERENCES

[1] Stop. Think. Connect, National Cybersecurity and Communications Integration Center (NCCIC). "Securing the Internet of Things | US-CERT". Us-Cert.Gov, 2018, https://www.us-cert.gov/ncas/tips/ST17-001.

[2] McDowell, Mindi. "Understanding Hidden Threats: Rootkits and Botnets | US-CERT". Us-Cert.Gov, 2018, https://www.us-cert.gov/ncas/tips/ST06-001.

[3] Hoque, Nazrul, Dhruba K. Bhattacharyya, and Jugal K. Kalita. "Botnet in DDoS attacks: trends and challenges." IEEE Communications Surveys & Tutorials 17.4 (2015): 2242-2270.

[4] Stallings, William. Network Security Essentials. 4th ed., Prentice Hall., 2011.

[5] Guttman, Barbara, and Edward A. Roback. An introduction to computer security: the NIST handbook. DIANE Publishing, 1995.

[6] IHS Statista. (2019). IoT: number of connected devices worldwide 2012-2025 | Statista. https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/.

[7] Aimoto, Shaun. “Android Malware on Google Play Adds Devices to Botnet.” Symantec, 18 Oct. 2017, www.symantec.com/connect/blogs/android-malware-google-play-adds-devices-botnet-and-performs-ddos-attacks.

[8] Greenberg, Andy. “The Reaper Botnet Has Already Infected a Million Networks.” Wired, Conde Nast, 20 Oct. 2017, www.wired.com/story/reaper-iot-botnet-infected-million-networks/.

[9] Fruhlinger, Josh. “The Mirai Botnet Explained: How IoT Devices Almost Brought down the Internet.” CSO Online, CSO, 9 Mar. 2018, www.csoonline.com/article/3258748/security/the-mirai-botnet-explained-how-teen-scammers-and-cctv-cameras-almost-brought-down-the-internet.html.

[10] Newman, Lily Hay. “What We Know About Friday's Massive East Coast Internet Outage.” Wired, Conde Nast, 3 June 2017, www.wired.com/2016/10/internet-outage-ddos-dns-dyn/.

[11] Tan, Jonathan. “Should You Fear the Reaper?” Networks Asia, 8 Jan. 2018, www.networksasia.net/article/should-you-fear-reaper.1515387121.

[12] Raynwood, Dan. “CIA Website Taken down by DDoS Attack.” SC Media UK, 16 June 2011, www.scmagazineuk.com/article/1483783.

[13] Vijayan, Jaikumar. “LulzSec Members Plead Guilty to DDoS Attacks on Sony, CIA, Others.” Computerworld, Computerworld, 25 June 2012,

www.computerworld.com/article/2504861/security0/lulzsec-members-plead-guilty-to-ddos-attacks-on-sony--cia--others.html.

[14] Spring, Tom. “Blizzard Entertainment Hit With Weekend DDoS Attack.” The First Stop for Security News | Threatpost, 14 Apr. 2017, threatpost.com/blizzard-entertainment-hit-with-weekend-ddos-attack/127440/.

[15] Graham, Alan. 888 Poker Suffers DDoS Attacks, Pokernewsreport.com, www.pokernewsreport.com/bigger-online-super-series-cancelled-due-to-ddos-attacks-21870.

[16] Morbin, Tony. “National Lottery Hit by DDoS Attack - down 90 Mins at Peak Demand Time.” SC Media UK, 2 Oct. 2AD, www.scmagazineuk.com/national-lottery-hit-ddos-attack-down-90-mins-peak-demand-time/article/1474021.

[17] Leswing, Kif. “A Massive Cyberattack Knocked out Major Websites across the Internet.” Business Insider, Business Insider, 21 Oct. 2016, www.businessinsider.com/amazon-spotify-twitter-github-and-etsy-down-in-apparent-dns-attack-2016-10.

[18] “DDoS Attack Takes Down Amazon, Spotify, Reddit, and Other Sites.” SecureLink, 30 May 2018, www.securelink.com/blog/dyn-ddos-attack-takes-down-twitter-amazon-netflix/.

[19] Jonker, Mattijs, et al. "Millions of targets under attack: a macroscopic characterization of the DoS ecosystem." Proceedings of the 2017 Internet Measurement Conference. ACM, 2017.

[20] Froelich, Warren. “A Third of the Internet Is Under Attack.” UC San Diego, 1 Nov. 2017, ucsdnews.ucsd.edu/pressrelease/a_third_of_the_ internet_is_ under _attack.

[21] Khalimonenko, Alexander. “DDoS Attacks in Q1 2018.” Securelist - Kaspersky Lab's Cyberthreat Research and Reports, 26 Apr. 2018, securelist.com/ddos-report-in-q1-2018/85373/.

[22] Ibragimov, Timur, et al. “DDoS Attacks in Q2 2018.” Securelist - Kaspersky Lab's Cyberthreat Research and Reports, 24 July 2018, securelist.com/ddos-report-in-q2-2018/86537/.

[23] Kupreev, Oleg, et al. “DDoS Attacks in Q3 2018.” Securelist - Kaspersky Lab's Cyberthreat Research and Reports, 31 Oct. 2018, securelist.com/ddos-report-in-q3-2018/88617/.

[24] Gunnam, Ganesh Reddy, and Sanjeev Kumar. "Do ICMP Security Attacks Have Same Impact on Servers?" Journal of Information Security 8.03 (2017): 274. doi: 10.4236/jis.2017.83018

[25] Surisetty, Sirisha, and Sanjeev Kumar. "Apple's Leopard Versus Microsoft's Windows XP: Experimental Evaluation of Apple's Leopard Operating System with Windows XP-SP2 under Distributed Denial of Service Security Attacks." Information Security Journal: A Global Perspective 20.3 (2011): 163-172. doi: 10.1080/19393555.2011.569908

[26] Junior, Rodolfo Baez, and Sanjeev Kumar. "Apple's Lion vs Microsoft's Windows 7: Comparing Built-In Protection against ICMP Flood Attacks." Journal of Information Security 5.03 (2014): 123. doi: 10.4236/jis.2014.53012

[27] Kumar, Sanjeev, and Sirisha Surisetty. "Microsoft vs. Apple: Resilience against distributed denial-of-service attacks." IEEE Security & Privacy 10.2 (2012): 60-64. doi: 10.1109/MSP.2011.147

[28] Gade, Raja Sekhar Reddy, Hari Vellalacheruvu, and Sanjeev Kumar. "Performance of Windows XP, Windows Vista and Apple's Leopard computers under a denial of service attack." 2010 Fourth International Conference on Digital Society. IEEE, 2010. doi: 10.1109/ICDS.2010.39

[29] Kumar, Sanjeev. "PING attack–How bad is it?" Computers & Security 25.5 (2006): 332-337. doi: 10.1016/j.cose.2005.11.004

[30] Kumar, Sanjeev. "Smurf-based distributed denial of service (ddos) attack amplification in internet." Second International Conference on Internet Monitoring and Protection (ICIMP 2007). IEEE, 2007. doi: 10.1109/ICIMP.2007.42

[31] Herrera, Hugo Armando, William Robert Rivas, and Sanjeev Kumar. "Evaluation of Internet Connectivity Under Distributed Denial of Service Attacks from Botnets of Varying Magnitudes." 2018 1st International Conference on Data Intelligence and Security (ICDIS). IEEE, 2018. doi: 10.1109/ICDIS.2018.00026

[32] Sundar, Koushicaa, and Sanjeev Kumar. "Blue Screen of Death Observed for Microsoft Windows Server 2012 R2 under DDoS Security Attack." Journal of Information Security 7.04 (2016): 225. doi: 10.4236/jis.2016.74018

[33] Kumar, Sanjeev, and Raja Sekhar Reddy Gade. "Evaluation of Microsoft Windows Servers 2008 & 2003 against Cyber Attacks." Journal of Information Security 6.02 (2015): 155. doi: 10.4236/jis.2015.62016

[34] Petana, Einar, and Sanjeev Kumar. "TCP SYN-based DDoS attack on EKG signals monitored via a wireless sensor network." Security and Communication Networks 4.12 (2011): 1448-1460. doi: 10.1002/sec.275

[35] Kumar, Sanjeev, and Einar Petana. "Mitigation of TCP-SYN attacks with Microsoft's Windows XP Service Pack2 (SP2) software." Seventh International Conference on Networking (icn 2008). IEEE, 2008. doi: 10.1109/ICN.2008.77

[36] Surisetty, Sirisha, and Sanjeev Kumar. "Is Apple's iMac Leopard Operating System Secure under ARP-Based Flooding Attacks?." 2010 Fifth International Conference on Internet Monitoring and Protection. IEEE, 2010. doi: 10.1109/ICIMP.2010.30

[37] "The Morris Worm." FBI, FBI, 2 Nov. 2018, www.fbi.gov/news/stories/morris-worm-30-years-since-first-major-attack-on-internet-110218.

[38] Kaspersky, Eugene. "A Brief History of DDoS Attacks." Nota Bene Eugene Kasperskys Official Blog, eugene.kaspersky.com/2016/12/06/a-brief-history-of-ddos-attacks/.

[39] Bertino, Elisa, and Nayeem Islam. "Botnets and internet of things security." Computer 2 (2017): 76-79.

[40] What Is a DDoS Botnet | Cloudflare. Cloudflare, www.cloudflare.com/learning/ddos/what-is-a-ddos-botnet/.

[41] Ollmann, Gunter. "Botnet communication topologies–understanding the intricacies of botnet command-and-control." Damballa White Paper (2009).

[42] What Is a Botnet | Preventing Botnet Attacks | Kaspersky Lab US. Kaspersky US, usa.kaspersky.com/resource-center/threats/botnet-attacks.

[43] “What Is A Botnet?” Norton Family Premier, Symantec Corporation, us.norton.com/internetsecurity-malware-what-is-a-botnet.html.

[44] Khalimonenko, Alexander, et al. “DDoS Attacks in Q3 2017.” Securelist - Kaspersky Lab's Cyberthreat Research and Reports, 6 Nov. 2017, securelist.com/ddos-attacks-in-q3-2017/83041/.

[45] “Creating and Managing Strong Passwords.” "Plan, Do, Check, Act" | US-CERT, www.us-cert.gov/ncas/current-activity/2018/03/27/Creating-and-Managing-Strong-Passwords.

[46] Grassi, Paul A, et al. “National Institute of Standards and Technology.” NIST Special Publication 800-63B, pages.nist.gov/800-63-3/sp800-63b.html.

[47] “What Is SETI@Home?” SETI@Home, setiathome.ssl.berkeley.edu/.

[48] “Small Contributions, Great Causes.” GPUGRID, www.gpugrid.net/science.php.

[49] Choosing BOINC Projects, University of California, Berkley, boinc.berkeley.edu/projects.php.

[50] Day, John D., and Hubert Zimmermann. "The OSI reference model." Proceedings of the IEEE 71.12 (1983): 1334-1340.

[51] Briscoe, Neil. "Understanding the OSI 7-layer model." PC Network Advisor 120.2 (2000).

[52] Miller, Rachelle L. "The OSI model: An overview." SANS Institute InfoSec Reading Room (2001).

[53] MacMichael, Duncan. “Windows Network Architecture and the OSI Model - Windows Drivers.” Windows Drivers | Microsoft Docs, Microsoft Hardware Dev Center, 19 Apr. 2017, docs.microsoft.com/en-us/windows-hardware/drivers/network/windows-network-architecture-and-the-osi-model.

[54] Russell, Andrew L. “OSI: The Internet That Wasn't.” IEEE Spectrum: Technology, Engineering, and Science News, IEEE Spectrum, 30 July 2013, spectrum.ieee.org/tech-history/cyberspace/osi-the-internet-that-wasnt.

[55] “About.” IETF, www.ietf.org/about/.

[56] Forouzan, Behrouz A. TCP/IP Protocol Suite. 4th ed., McGraw-Hill Higher Education, 2010.

[57] Stallings, William Walter. Data and Computer Communications. 8th ed., Pearson/Prentice Hall, 2007.

[58] Leiner, Barry, et al. "The DARPA Internet protocol suite." IEEE Communications Magazine 23.3 (1985): 29-34.

[59] Braden, Robert. "RFC 1122: Requirements for Internet hosts–communication layers." (1989).

[60] Braden, Robert. "RFC 1123–Requirements for Internet Hosts–Application and Support." (1989).

[60] Postel, Jon. "RFC 791: Internet protocol." (1981).

[61] Postel, Jon. Internet control message protocol. No. RFC 792. 1981.

[62] Mogul, Jeffrey Clifford, and Jon Postel. Internet standard subnetting procedure. No. RFC 950. 1985.

[63] Bonica, R., et al. Extended ICMP to support multi-part messages. No. RFC 4884. 2007.

[64] Gont, Fernando. Deprecation of ICMP Source Quench Messages. No. RFC 6633. 2012.

[65] Gont, F., and C. Pignataro. Formally deprecating some icmpv4 message types. No. RFC 6918. 2013.

[66] Internet Control Message Protocol (ICMP) Parameters, IANA, 15 June 2018, www.iana.org/assignments/icmp-parameters/icmp-parameters.xhtml.

[67] Khalimonenko, Alexander, et al. “DDoS Attacks in Q4 2017.” Securelist English, Kaspersky US, 6 Feb. 2018, securelist.com/ddos-attacks-in-q4-2017/83729/.

[68] Khalimonenko, Alexander, et al. “DDoS Attacks in Q2 2017.” Securelist English, Kaspersky US, 1 Aug. 2017, securelist.com/ddos-attacks-in-q2-2017/79241/.

[69] Khalimonenko, Alexander, and Oleg Kupreev. “DDOS Attacks in Q1 2017.” Securelist English, Kaspersky US, 11 May 2017, securelist.com/ddos-attacks-in-q1-2017/78285/.

[70] “Ping (ICMP) Flood DDoS Attack.” Cloudflare, www.cloudflare.com/learning/ddos/ping-icmp-flood-ddos-attack/.

[71] “Smurf DoS Attack.” Smurf DoS Attack, Symantec Corporation, us.norton.com/online-threats/glossary/s/smurf-dos-attack.html.

[72] “What Is a Smurf Attack?” Usa.kaspersky.com, Kaspersky US, usa.kaspersky.com/resource-center/definitions/smurf-attack.

[73] “Smurf DDoS Attack.” Cloudflare, www.cloudflare.com/learning/ddos/smurf-ddos-attack/.

[74] Mogul, J. C. "RFC 919: Broadcasting Internet datagrams, October 1984." See also STD0005 [30]. Status: STANDARD.

[75] Mogul, Jeffrey. "RFC-922 Broadcasting Internet Datagrams in the Presence of Subnets." Network Working Group (1984).

[76] Postel, Jon. "RFC 793: Transmission control protocol." (1981).

[77] Ramakrishnan, K., Sally Floyd, and D. Black. "The Addition of Explicit Congestion Notification (ECN) to IP: RFC 3168." Internet Engineering Task Force (2001).

[78] Gont, F., and A. Yourtchenko. On the Implementation of the TCP Urgent Mechanism. No. RFC 6093. 2011.

[79] Bellovin, Steven M., and Fernando Gont. "Defending against sequence number attacks." (2012).

[80] Eddy, Wesley. TCP SYN flooding attacks and common mitigations. No. RFC 4987. 2007.

[81] Advisory, C. E. R. T. "CA-1996-21 TCP SYN flooding and IP Spoofing Attacks." (1996).

[82] “SYN Flood DDoS Attack.” Cloudflare, www.cloudflare.com/learning/ddos/syn-flood-ddos-attack/.

[83] “Defenses Against TCP SYN Flooding Attacks - The Internet Protocol Journal - Volume 9, Number 4.” Cisco, 30 Nov. 2017, www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-34/syn-flooding-attacks.html.

[84] Postel, Jon. User datagram protocol. No. RFC 768. 1980.

[85] Eggert, L., G. Fairhurst, and G. Shepherd. "RFC 8085, UDP usage guidelines." Internet Engineering Task Force (IETF) (2017).

[86] “UDP Flood DDoS Attack.” Cloudflare, www.cloudflare.com/learning/ddos/udp-flood-ddos-attack/.

[87] “A Cisco Guide to Defending Against Distributed Denial of Service Attacks.” Cisco, 12 Mar. 2018, www.cisco.com/c/en/us/about/security-center/guide-ddos-defense.html#11.

[88] “Microsoft Lifecycle Policy.” Support.microsoft.com, Microsoft, support.microsoft.com/en-us/lifecycle/search?alpha=Windows Server 2012 R2.

[89] Poggemeyer, Liza, and Takeshi Katano. “Windows Server Release Information.” Microsoft Docs, Microsoft, 11 Dec. 2018, docs.microsoft.com/en-us/windows-server/get-started/windows-server-release-info.

[90] “An Incredible Web Server That's Built around You...” Overview : The Official Microsoft IIS Site, www.iis.net/overview.

[91] Fielding, R., and J. Reschke. "RFC 7230: Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing, June 2014."

[92] “Mac Pro (Mid 2010) - Technical Specifications.” Official Apple Support, Apple, support.apple.com/kb/SP589?locale=en_US.

[93] “BCM5719 - Quad-Port 1GBASE-T.” Broadcom Inc., www.broadcom.com/products/ethernet-connectivity/network-ics/bcm5719-1gbase-t-ic.

[94] “Intel® Xeon® Processor W3530 (8M Cache, 2.80 GHz, 4.80 GT/s Intel® QPI) Product Specifications.” Intel® Xeon® Processor W3530, Intel,

ark.intel.com/content/www/us/en/ark/products/41313/intel-xeon-processor-w3530-8m-cache-2-80-ghz-4-80-gt-s-intel-qpi.html.

[95] Cisco. “WebView Switches User Guide Business Series Model: SRW2048, SRW2024, SRW2016, SRW248G4, SRW224G4”. Cisco, https://www.cisco.com/c/dam/en/us/td/docs/switches/lan/csbms/srw2048/administration/guide/SRW-US_v10_UG_A-Web.pdf.

[96] Huculak, Mauro. “How to Use Performance Monitor on Windows 10.” Windows Central, Windows Central, 16 Feb. 2017, www.windowscentral.com/how-use-performance-monitor-windows-10.

[97] Liberman, Arthur. Core Temp, www.alcpu.com/CoreTemp/.

[98] “MATLAB R2016b.” MATLAB - MathWorks - MATLAB & Simulink, 9.1, MathWorks, 15 Sept. 2016, www.mathworks.com/products/matlab.html.

[99] “Microsoft Excel.” Spreadsheet Software - Excel Free Trial, products.office.com/en-us/excel.

[100] Marr, Deborah T., et al. "Hyper-Threading Technology Architecture and Microarchitecture." Intel Technology Journal 6.1 (2002).

[101] “Intel® Hyper-Threading Technology.” Intel, Intel, www.intel.com/content/www/us/en/architecture-and-technology/hyper-threading/hyper-threading-technology.html.

[102] Tarra, Hemanth, and Benoit Jester. “Understanding Processor (% Processor Time) and Process (%Processor Time).” TechNet, Microsoft, 13 June 2014, social.technet.microsoft.com/wiki/contents/articles/12984.understanding-processor-processor-time-and-process-processor-time.aspx.

[103] CompuTronix. “Walkthrough - Intel Temperature Guide.” [Msg 1]. Tom's Hardware Forum, Tom's Hardware, 13 Sept. 2013, forums.tomshardware.com/threads/intel-temperature-guide.1488337/.

[104] Painter, Lewis. “Mac OS X & MacOS Version Code-Names.” Macworld UK, 1 Apr. 2019, www.macworld.co.uk/feature/mac/mac-os-x-macos-version-code-names-3662757/.

[105] “Apple Security Updates.” Apple Support, 11 Apr. 2019, support.apple.com/en-us/HT201222.

[106] “MacOS - What Is MacOS.” Apple, Apple, www.apple.com/macos/what-is/.

[107] “MacOS Server.” Apple, Apple, www.apple.com/macos/server/.

[108] MAMP GmbH. “MAMP & MAMP PRO - Your Local Web Development Solution for PHP and WordPress Development.” MAMP & MAMP PRO - Your Local Web Development Solution, MAMP GmbH, www.mamp.info/en/.

[109] “Premier Fan Control App for MacOS.” TG Pro - Fan Control & Temp Monitoring for MacOS, www.tunabellysoftware.com/tgpro/.

BIOGRAPHICAL SKETCH

Hugo Armando Herrera Martinez earned his Bachelor of Science in Electrical Engineering at the University of Texas Pan American on 2014. He continued his education at the University of Texas Rio Grande Valley and earned his Master of Science in Electrical Engineering on May 2019. During enrollment he was employed as a Teaching Assistant in the Electrical Engineering Department at UTRGV from June 2016 to June 2018 where he also worked as a Research Assistant in the Network Research Lab. Additionally he was employed as a Technology Intern for the Edinburg Consolidated Independent School District throughout his graduate studies.

E-mail: haherrera90@gmail.com

Publications and poster presentations:

- Herrera, H, Rivas W. and Kumar S. "Evaluation of Internet Connectivity Under Distributed Denial of Service Attacks from Botnets of Varying Magnitudes." 2018 1st International Conference on Data Intelligence and Security (ICDIS). IEEE, 2018.
- Herrera, H, and Kumar S. (In Progress) "Assessment of Smurf Mitigation Techniques."
- Herrera, H, Rivas W, and Kumar S. "Impact of Botnet Size on Server Performance." 2017 Poster for HESTEC 2017 competition. UTRGV, 2017.
- Herrera, H, Navarro C, and Kumar S. "Performance Under the Impact of a Large Scale Botnet Utilizing Smurf Attack on Windows 2012 R2." 2018 Poster for HESTEC 2018 competition. UTRGV, 2018.