12-2020

Cyber Security Evaluation of CentOS Red Hat Based Operating System Under Cyber Attack with Increasing Magnitude

William E. R. Rivas
The University of Texas Rio Grande Valley

CYBER SECURITY EVALUATION OF CENTOS RED HAT BASED OPERATING SYSTEM UNDER CYBER ATTACK WITH INCREASING MAGNITUDE

A Thesis

by

WILLIAM E. R. RIVAS

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

MASTER OF SCIENCE IN ENGINEERING

December 2020

Major Subject: Electrical Engineering

CYBER SECURITY EVALUATION OF CENTOS RED HAT BASED OPERATING SYSTEM UNDER CYBER ATTACK WITH INCREASING MAGNITUDE

A Thesis
by
WILLIAM E. R. RIVAS

COMMITTEE MEMBERS

Dr. Sanjeev Kumar
Chair of Committee

Dr. Mark Yul Chu
Committee Member

Dr. Wenjie Dong
Committee Member

December 2020

ABSTRACT

Rivas, William, Cyber Security Evaluation of CentOS Red Hat Based Operating System Under Cyber Attack with Increasing Magnitude, Master of Science in Engineering (MSE), December 2020, 107 pp., 88 figures, 100 references

There are new internet accessible devices on the market every day, any of which can be used in a network of zombie machines meant to carry out an attack, called a botnet. These botnets are used to flood a system with information, ideally consuming large amounts of resources, such as memory or processing power. If the attack is successful, operation within the target system is effectively halted, often for long periods of time in the more severe attacks. Just like the variety in devices, there is a variety in the software that operates these devices. In this experiment, I focus efforts on comparing the ability of CentOS 15 with Windows Server 2012R to function under attack. I analyze four popular DDoS attacks using simulated network traffic consisting of botnets ranging from of over 16 million systems, 65 thousand systems and 254 systems in a controlled, closed environment.

DEDICATION

I want to credit the fulfillment of Graduate Thesis to my mother, Yolanda Rivas. She was the initial voice that pushed me to pursue further education and it is because of her unending support that I had the motivation to complete my studies. It was her leading by example that inspired me to work as hard as possible. Now I can put my graduate degree next to hers.

ACKNOWLEDGEMENTS

I want to thank Dr. Sanjeev Kumar, for his willingness to lead me down this path of further education, and the use of his knowledge and experience to spearhead my dissertation committee. I also want to thank the rest of my committee, Dr. Wenjie Dong and Dr. Yul Chu, for their time and patience in assisting me.

I would like to thank all the members of the Network Research lab at UTRGV. Every one of my friends and colleagues there gave me input that culminated in this final product and I would be nowhere without them as well.

This research work is based upon work supported in part by the US National Science Foundation under Grant No. 0421585 and Houston Endowment Chair in Science, Math and Technology Fellowship. I want to thank the NSF for providing the funding which allowed this evaluation to be performed.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

CHAPTER I

INTRODUCTION

It is no secret that the number of devices with internet connectivity is at the highest it's ever been and is only going to continue to grow. Especially with the rise of cryptocurrency in recent years [5]. It is because of this that there should be awareness about the DDoS attacks that occur daily. Simply put, a distributed denial of service attack (DDoS) is a coordinated attack made by several computing systems that have been compromised by a user or group of users, sometimes referred to as a "hacker group". There are many methods of acquiring these large banks of computing devices, ranging from phishing, social engineering, exploits, etc. These large computer banks of machines seized by hacker groups are called botnets. Each bot is an individual machine and is referred to as a bot simply because it is meant to be used not by a single user, but instead in conjunction with even more "bots". There machines are used without the knowledge of the owner or administrator, often undetected unless actively inspected for suspicious transmissions. The intention is to disrupt. The intention is often a means to an end. One such example is the attacks on cryptocurrency exchanges [5]. In this example, these attacks are intended to disrupt trading of cryptocurrency, with the possibility of theft of currency. This is just one example of an everyday occurrence. The power and magnitude of these attacks depends on the size of the botnet available to the attacker or attackers. To reiterate, the goal here is to disrupt a target system [2].

With the Internet of Things, or IoT, booming in recent years, available devices have spiked. This led to an increase in DDoS attacks in the last two years. Not only has the frequency increased, but so has the intensity by several hundred percent [1]. In previous years, attacks of a couple hundred Gigabytes were enough to bring a service to a halt. Recently in 2018, Akamai/Prolexic reported the 1.3Tbps attack against Github and shortly after that an attack of 1.7 Tbps was reported by a separate mitigation agency [3]. Fortunately, both attacks were mitigated before any long-term damage could occur. In less than 3 years, the strength of possible attacks has increased by a staggering amount. Kaspersky Lab performs a quarterly DDoS Intelligence Report. The report for the fourth quarter of 2019 shows a clear and strong growth in attacks, especially smart attacks. This report also shows a distinctly popularity in TCP attacks performed in the final quarter of last year, although all types of attacks continue to occur and are just as relevant. This report goes on to show that the vast majority of botnets are registered in the United States.[4].

In order to cement my understanding of these attacks and the significance of preventing these types of attacks, several publications were covered [6-15]. This paper will focus on four commonly Layer 3 and Layer 4, shown in Kaspersky's quarterly report: TCP, UDP, Ping (ICMP), and Smurf (ICMP) [4]. In order to evaluate CentOS's ability to mitigate difference intensity of attack, different Classes of addressing are used in this evaluation; Specifically, Class A, Class B, and Class C addressing schemes are employed.

There is much debate as to which Operating System is best for server-side operations. When deciding on a commercial Operating System, there are typically three choices: Windows, Apple, or one of the various distributions of Linux. However, these options become fewer when deciding what OS to use for server operations. This is due to macOS server being proprietary and

useful only to homes or businesses dedicated to using Mac products, which limits the flexibility and compatibility. This leaves the debate between the final two contenders, Windows and Linux.

Windows is the most widely used operating system since most people are familiar with it and it is very user friendly. Windows server is very popular for back-end operations for the same reason, whether it be business or home use. Most people's first OS is Windows; Since it is quite common, one should have minimal issue with flexibility or compatibility between systems. Windows also comes prepackaged with many utilities which aid runtime and assist users. However, ease of use is not the most important specification when it comes to choosing a server OS, especially for business.

When running servers for business with a large number of users, Distributed Denial of Service (DDoS) attacks are commonplace. Windows may be the most popular operating systems for several reasons, but the question is if this is true from a security standpoint. The ease of use and flexibility mean nothing when the entire system is unusable due to system resources being consumed by dummy traffic. Since Windows comes with many utilities that run in the background, this bloat may also contribute to an attack indirectly, further saturating the systems resources.

This leads to the other end of the debate. With Linux, you have the ability to build your operating system from the kernel up. You can choose what software and utilities run and when. This eliminates any bloat that may come with a Windows system. This is to say that Windows has software that runs by default when using a fresh install that Linux does not. This can create additional overhead that could ideally be used circumvent the attack or continue operation under attack load. Aside from all this, several distributions of Linux, including ones for server operations, can be acquired for free.

The most popular Linux distributions for server-side use are UBUNTU Server, CentOS, and Red Hat Enterprise Linux (RHEL); the latter OS is specifically designed for business use and is very popular amongst large companies. CentOS will be the primary topic of this evaluation. CentOS is based on Red Hat and is often regarded as the free version of RHEL. CentOS by default, requires most utilities to be configured by the user, i.e. the firewall that is included in the OS packages blocks all traffic and protocols need to be enabled. Linux also allows minimalist installations, command line only. In this sense, you can build upwards based on your security needs and resource pool. The variety of options and control allow very flexible fault handling.

Up until this point, my research has consisted of stress testing Operating Systems on fixed hardware, typically Windows Server. I modeled this experiment after previous experiments where we set up a target server flooded it with attack traffic to monitor the effect DDoS attacks have on internet connectivity [15]. In that experiment, we set up a Target/ Victim, supplied it with "normal traffic" and then interrupted the normal traffic with Layer 4 attack traffic, TCPSYN Flood and UDP Flood. Again, this was only performed on Windows Server. While researching I began reading Kaspersky Quarterly Reports and determined I wanted to compare a Windows environment to a Linux environment, since those are the two most attacked Operating Systems [4]. Apple server is not listed in the Quarterly Reports, as most of its components have been stripped and it was ultimately discontinued [99].I chose CentOS as the Linux distribution for this experiment because it is the distribution I had spent the most time practicing with and I wanted to evaluate the operating system as I had evaluated Windows Server in the past [15]. Since this is an evaluation of CentOS as well as a comparison to Windows, I included Layer 3 attacks as well: Layer 3 attacks being Ping Flood and Smurf attack.

Most comparisons I came across opted for Linux as the superior OS when compared to Windows, citing scalability and smaller installation footprint as well as many other points [100]. My hypothesis is that CentOS will outperform Windows Server 2012R under DDoS attack when both Operating Systems are evaluated out-of-the-box, based on the smaller footprint and the difference in CPU scheduling [97,100]. I want to evaluate both OS out-of-the-box because the less configuration the consumer has to perform, the better, and CentOS is packaged with less bloatware than Windows 2012R so I determine it will outperform [100].

Problem Statement

Issues with cyber-security are ever-growing. Internet usage is commonplace today and with that, comes all the problem that are typical for internet usage. The most commonly used Operating Systems may not be best when it comes to ensuring the integrity of your data. Regardless of the Operating System, hardware, or software, computer systems come with certain vulnerabilities. Distributed Denial of Service (DDoS) occur daily and all over the world against every type of service available. These attacks aim to deny the user access and disrupt a target service. Every operating system comes with tools to help ward off threats, but commercial operating systems may come with additional bloatware that can use up resources in the background and add to the resource saturation brought on by traffic floods. This paper will focus on testing two Operating Systems' built-in protection and making a comparison between the two. Windows is a very popular, common OS but this experiment intends to argue that CentOS may be a better out-of-the-box option. This experiment will consist of comparing Windows Server 2012R and CentOS and running four different DDoS attacks of varying magnitudes against these

systems with their most default operational configuration. The four attacks used in this experiment will be Ping Flood, Smurf Attack, TCPSYN Flood, and UDP Flood.

Distributed Denial of Service Attacks and the OSI Model

A Denial of Service attack (DoS attack) is form of a cyber-attack which means to prevent client operation of a target computer system. In a Denial of Service attack, a host floods a target services with false traffic. A notable example used for stress testing DoS attacks is the Low Orbit Ion Canon (LOIC), which floods a target with either TCP or UDP packets. A Distributed Denial of Service attack (DDoS attack) shares the same goal as a DoS attack; the key difference being a DoS attack is carried out by a single host, while a DDoS attack is carried out by a network of hosts, called a botnet. Typically, this botnet is a network of hijacked computers taken over by the party carrying out the attack [2].

Figure I-1 Denial of Service Attack Diagram

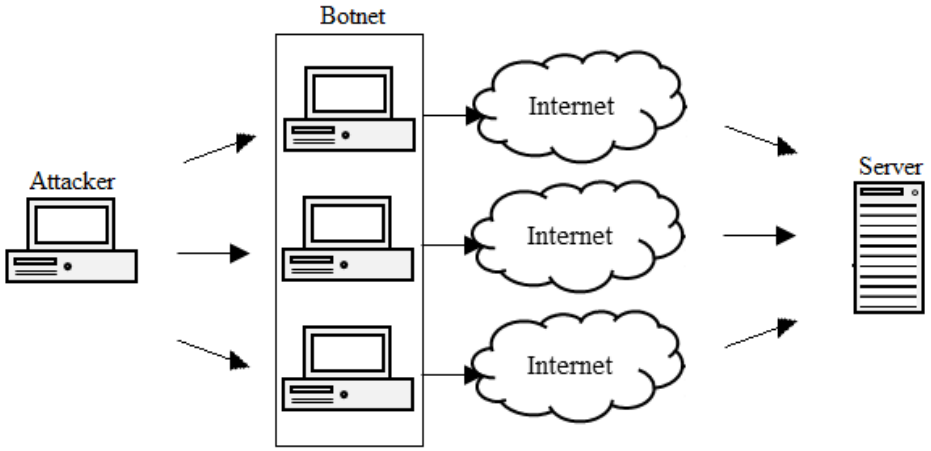

Figure I-2 Distributed Denial of Service Attack Diagram

The OSI, or the Open Systems Interconnections Model, is a very basic concept model of communication between devices. The intention is a standard form of communication adhered by computing devices. This breaks down communication into layers that rely on the previous layer in order to pass information along, also known as encapsulation and decapsulation. Figure I-3 represents an OSI Model that illustrates encapsulation and decapsulation. The OSI model has a total of seven layers. For this evaluation, we are going to focus on two adjacent layers, Network and Transport.

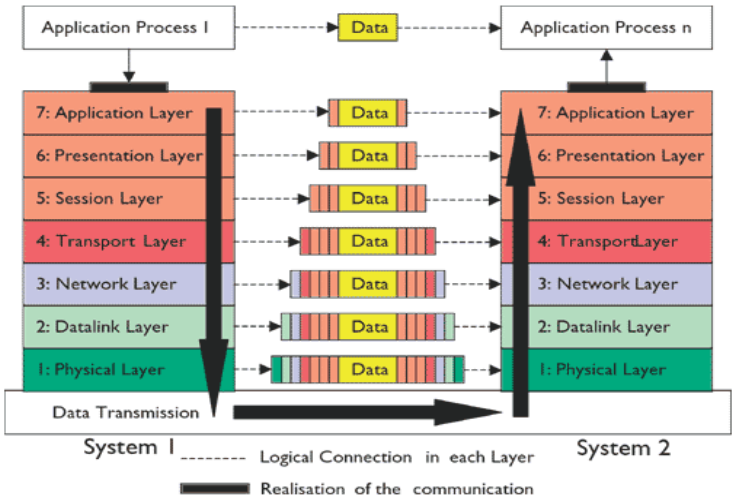

Figure I-3 OSI Model Illustration [98]

The attacks outlined in this paper are primarily ICMP Ping Flood Attack, ICMP Smurf Attack, TCPSYN Flood attack, and UDP Flood Attack. The former two are OSI Network Layer attacks and the latter are OSI Transport Layer attacks; Layers 3 and 4, respectively. Each of these attacks intends to exploit the automated responses to requests in order to render the target device, or devices, inoperable. Albeit in different ways. The key difference is the way each protocol at each layer in the OSI model works. Each attack will be outlined in detail with illustrations for easy understanding.

Ping Flood DDoS Attack

The Internet Control Message Protocol (ICMP) is a feedback oriented protocol designed to run on IP and is outlined in RFC 792; since this is primarily a means of fault communication for hosts and runs on IP, this protocol functions under Layer-3 of the OSI Model. ICMP messages are used to indicate a problem with communication or errors processing packets or datagrams. Although, ICMP is not meant to perfect. There is the possibility of ICMP packet loss

or datagram loss with no ICMP packet indicating the loss. Because of this, other protocol that encapsulate Layer 3, such as Layer 4, have their own means of creating reliable transmission. An example would be the previously mentioned three-way handshake. ICMP is not only used for reporting errors, however. It can also be used to redirect traffic and determine if a host on the system is responding or not active. Since ICMP has so many functions, it can be broken down into several different types, each of which have indicators of their particular purpose. [16-17]

Ping Flooding

ICMP Echo Request, or ICMP Ping, is used to determine if a host is listening and able to communicate. The initial host will send a packet to the second host. The second host will then reverse the source and destination IP addresses included in the ICMP Request when generating an ICMP Reply packet to send back to the initial host. This indicates that the second host is active and listening for communication. There are scenarios where the second host may be offline or set to ignore incoming ICMP Requests. In these cases, the second host may respond with different packet, such as ICMP Destination Unreachable. For this evaluation, it is assumed the target machine is actively listening for ICMP Request packets. [16]

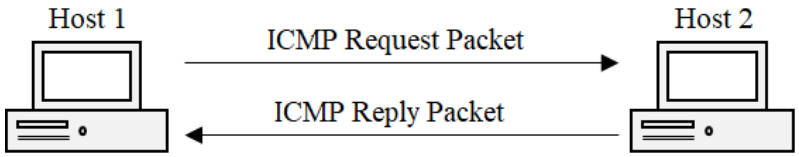

Figure I-4 ICMP Request Process

This automated process leads to another vulnerability that can be taken advantage of by an attacker. For every ICMP Request received, a listening host must create and transmit an ICMP Reply packet back to the original requester. So, the attacker transmits ICMP Ping Request

packets from every host on their botnet to the target system. The target system will receive each of these packets and begin using resources to write ICMP Reply packets and transmit them to the hosts on the botnet, who are simply dropping the Reply packets and sending more ICMP Request packets. This leads to mass consumption of resources and the inability to process legitimate services and disrupt normal traffic to the target system.

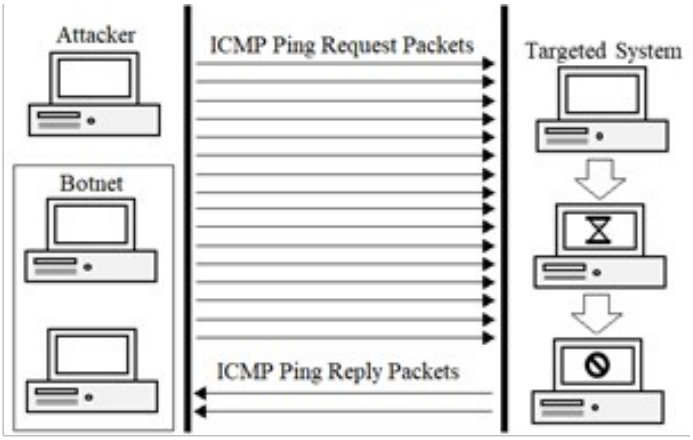

Figure I-5 ICMP Ping Flood, Target System is Flooded with ICMP Request Packets

Smurf DDoS Attack

The Internet Control Message Protocol is also the main protocol for this type of attack as well. Refer to Figure I-4 for details on the function and importance of ICMP. While the previous section outlined and attack using the ICMP Request function, this type of attack utilizes the ICMP Reply function as the means of flooding the target system with data and exhausting the target systems resources. Although they use the same protocol, this type of attack could be considered more sophisticated in comparison to the ICMP Ping Flood attack. This type of DDoS attack utilizes “man-in-the-middle” techniques. [16]

Smurf Attack

As previously stated, for every Request there must be a Reply. In this type of attack, the intention is not simply to mass transmit packets to a target system, but instead to trick the infected network into mass transmitting these packets for the attacker. The attacker spoofs their IP address so the source address in the ICMP header file indicates that the attacker is the target system. The attacker then broadcasts ICMP Request packets to every host on the botnet. Then, by normal function of ICMP, each host then reverses the source and destination fields in the ICMP header and creates ICMP Reply packets to send back to the source of the ICMP Request packets, i.e. the target system. Now the target system is being flooded with ICMP Reply packets and uses resources to process each packet. This type of attack is more complex because it requires the attacker to disguise itself as the target system and this requires specific information about the target system that may not be readily available. [16]

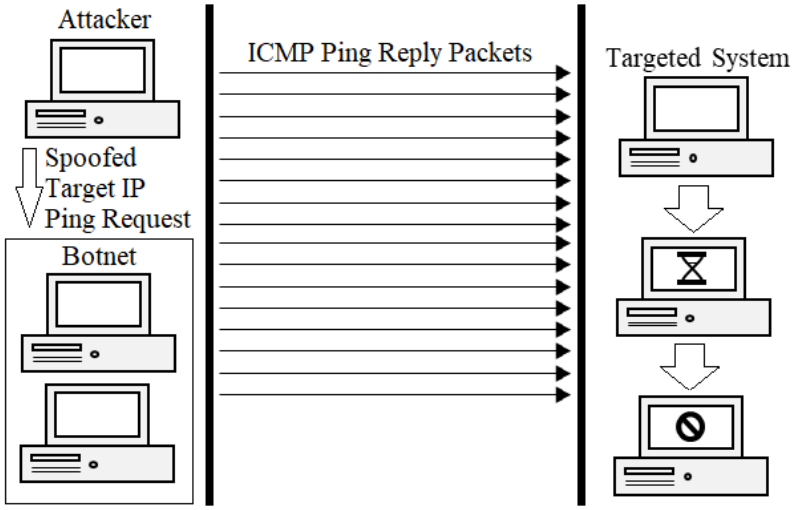

Figure I-6 ICMP Smurf Attack, Target System is Flooded with ICMP Reply Packets

TCP DDoS attack

The Transmission Control Protocol (TCP) is a connection-oriented protocol outlined in RFC 793; one of the current standards for end-to-end communication and functions under Layer-4 of the OSI Model. This documentation states that TCP is reliable for assuring the integrity of the transmission. No transmission occurs until both hosts have confirmed a reliable connection to each other. This method is called the “three-way handshake”. This connection remains open until transmission is complete, then closes. If no secure connection is established, the packets are dropped. This protocol is very commonly used by applications such as e-mail (SMTP), File Transfer Protocol (FTP), Secure Shell (SSH) for remote-access, as well as P2P file sharing and many others that can also be viewed in RFC 793 [18].

TCP-SYN Flooding

As mentioned before, TCP provides a reliable stable connection by implementing a method called the “three-way handshake”. This can also be described as “SYN, SYN+ACK, ACK.”

In order to initiate a new TCP connection, the host requesting the connection transmits a TCP packet containing an active Synchronize (SYN) flag and a sequence number to the second host. The sequence number is randomly generated when the request packet is created by the initial host system. The second host, the receiver, must respond with a packet containing an active Synchronize-Acknowledge (SYN-ACK) flag, the sequence number received in the SYN packet incremented by one, now referred to as the acknowledgment number, and a separate

randomly generated sequence number. To finalize the connection, the initiating host responds to the receiving host with an active Acknowledge (ACK) flag, the sequence number for this packet is the received acknowledgment number also incremented by one. Once this packet is received by the second host, the connection is fully established [18].

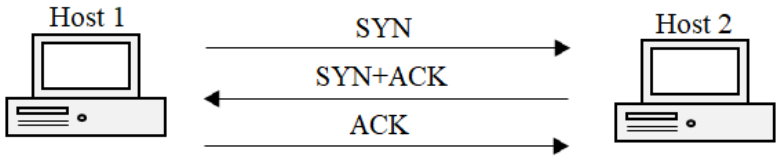

Figure I-7 TCP/IP "Three Way Handshake"

An attack occurs when a third-party takes advantage of this automated process with packets specifically designed to disrupt this handshake. The attacker, usually protected by using spoofed IP addresses and armed with a botnet army, floods a targeted system with TCP-SYN segments without sending ACK packets and fulfilling the 3-way handshake, illustrated in Figure I-7. The result is half-open connections saturating the Transmission Control Block (TCB) and preventing new connections from normal users to be established. The TCB also holds information for all active/half active connections, each of which uses some system resources to maintain their status as half-open. These half-open connections are dropped after a period, however. The intention of the attack is to establish and keep as many half-open connections as possible while the target system keeps them open for the configured period and eventually tries to drop them.

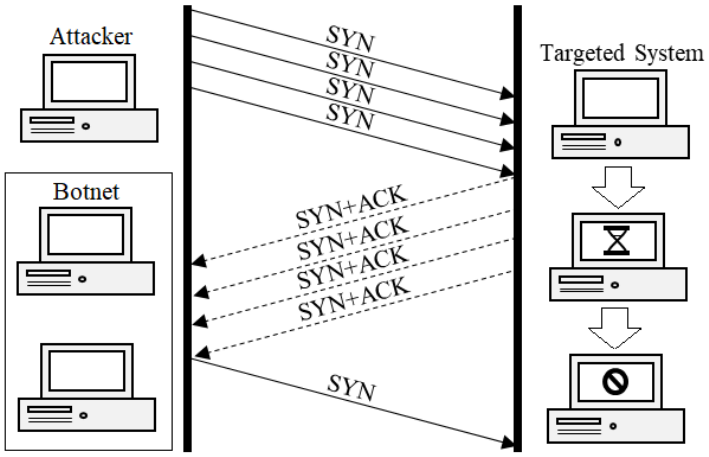

Figure I-8 TCP SYN Flood, Target System filled with half-open connections[15]

UDP DDoS attacks

The User Datagram Protocol (UDP) is a "connection-less" oriented protocol outlined in RFC 768; another current standard for end-to-end communication and functions under Layer-4 of the OSI Model. UDP is an alternative mode of packet transfer that is connection-less in the sense that this protocol does not require a three-way handshake. So, this protocol is quicker and requires less resources, at the expense of integrity and security. UDP does not guarantee packet delivery or packet duplication protection. UDP datagram headers are constructed in a way similar to TCP headers; the key difference much of the transmission data typically included in a TCP header is left out of a UDP header. The only information needed for UDP datagram transmission is Destination Port, Length, and Checksum. UDP can check if integrity has changed in transit, but no indication is transmitted, the datagram is merely dropped. UDP relies on Internet Protocol to be the underlying protocol. UDP is typically used in video streaming and Trivial File Transfer, as well as others listed in the respective RFC [19-20].

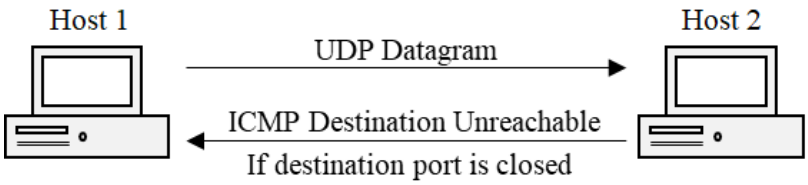

Figure I-9 UDP Transmission Process[15]

UDP Flooding

Since UDP datagrams are “send it and forget it”, these datagrams are sent directly to the destination port listed in the header and no record is kept of receival, or if the packet is dropped. However, if a UDP datagram is sent to port that is closed, Windows will respond with an ICMP Destination Unreachable packet. This is one of the only indicators a packet was received. While enabled in Windows, CentOS has this function disabled by default. This is done to prevent CPU saturation from creating these ICMP packets. However, receiving a large amount of UDP packets may still cause CPU saturation, if the botnet is large enough. So, simply put, a UDP flood typically consists of a large amount of ingress UDP packets and a large amount of egress ICMP packets, if the listening port is closed. However, since the OS being evaluated has these ICMP DU packets disabled and does not respond to UDP datagrams being sent to a closed port, we only need to worry about the former ingress attack traffic.

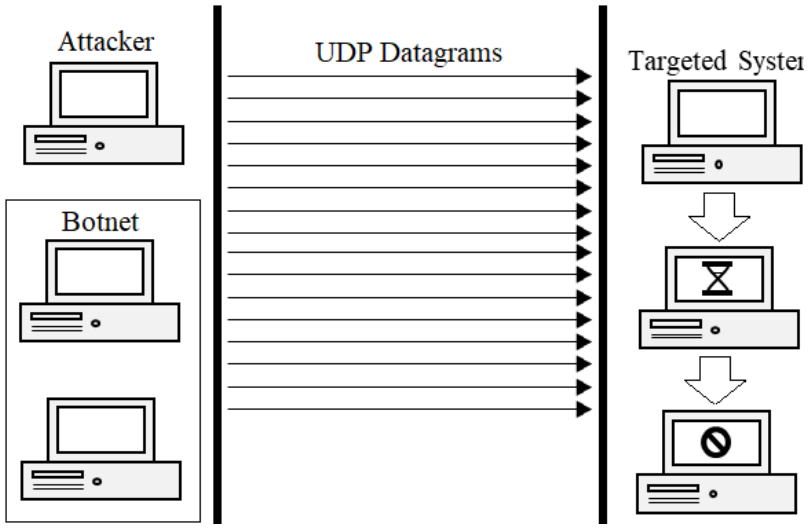

Figure I-10 UDP Flood, Target System is flooded with UDP Datagrams

Experiment Outline

For this evaluation, I performed various DDoS attacks against a target machine that was under HTTP GET packet load in order to test the availability of the web server under legitimate traffic and attack traffic. The legitimate traffic is simulated users attempting to access a web server, while the attack traffic comes from botnets that increase by certain thresholds based on the class of IP Addressing used. The target machine is consistent throughout the experiment and the specifications are listed below.

The testing environment used in this evaluation is illustrated in Figure I-11. This setup can be broken down into three sections: Attacker/Botnet, Normal Traffic, and the intended Victim machine. The Attacker transmits traffic intended to interrupt the Victim machine operation. The Normal Traffic machine transmits HTTP GET packets, intended to simulate a typical website. The Victim Machine records its resources consumption as it is attacked. The Victim Machine is a Dell PowerEdge T320 Server with the following specifications:

- Intel® Xeon® CPU E5-2407 v2 @ 2.40GHz Processor

- 8 GB of RAM.
- CentOS 15 as one Operating System.
    - Apache as the server
- Windows 2012R Server as the other Operating System
    - IIS as the server

The Victim server will receive four types of DDoS attack traffic from three botnets of 254 systems, 65 thousand systems and, 16+ million systems: random Class C, B, and A addresses, respectively. This is to say that for each of these systems an IP Address is generated and packets (dependent on the type of attack) are sent from that address to the Victim before another address is randomly generated and more packets are sent.

Normal traffic was a set of simulated users attempting to reach a website hosted by Apache within CentOS and IIS within Windows at 3000 HTTP GET transactions per second. To create a baseline, each trial begins with normal traffic being sent to the web server unimpeded by attack traffic. For the remainder of the trial, simulated attack traffic was generated and pushed to the victim server with an initial rate of 100 Megabits per second (Mbps) increasing by 100Mbps every 5 minutes to a maximum of 1Gbps, the maximum attack load for each trial. Each trial is roughly an hour long. In order to determine the ability of each Operating System to perform under DDoS attack, processor utilization and HTTP transactions per second were measured and recorded using network analyzing software. CentOS firewall blocks all ports and protocols by default, so these had to be enabled. No ports were changed, and all protocols follow their respective RFCs.

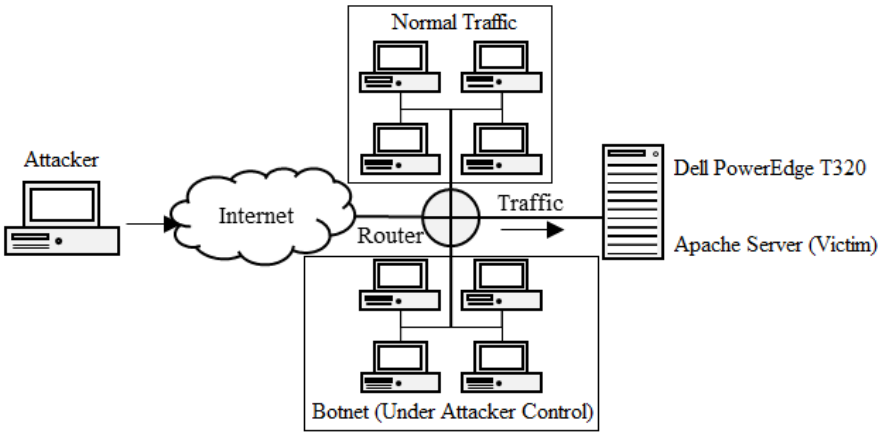

Figure I-11 Experimental Setup[15]

I began by evaluating the operability of Windows Server 2012R under heavy load. I gathered HTTP GET transaction information as well as CPU usage data using the built-in data collector software. I gathered data for each DDoS attack under each size of botnet. This data served as my baseline for the remainder of the experiment. I continued gathering data, now with a CentOS installation. Similar to Windows trials, I gathered HTTP GET transaction data and CPU usage data using a Linux utility called SYSSTAT. For each attack and botnet size, I gathered the HTTP GET transaction data for each OS and plotted it on the same axis to get visual comparison. The same was done with the CPU usage data. Inspection of these plots can determine which operating system is superior in terms of mitigating DDoS attack at varying scales.

CHAPTER II

WINDOWS SERVER OPERATING SYSTEM PERFORMANCE UNDER DDOS ATTACK

This section begins with Layer 3 attacks, or Network Layer attacks. In the OSI model, higher layers encapsulate lower layers so, for example, Transport Layer attacks still include the use of IP. However, these different layers have their own means of security and maintaining communication integrity. Meaning, the methods used in Layer 3 will different from those of Layer 4 as they are meant to work at different stages. So, there should be discernable difference in the results of each attack. The attacks we will focus on first are Layer 3 attacks, followed by Layer 4 attacks.

In order to quantify each Operating System's ability to function under DDoS Attack, we gathered usage data from the CPU and monitored throughput traffic into the target machine. The Data Collector Set was used to gather the CPU data for each of the Windows trials. This is a built-in utility unique to Windows. This utility recorded the CPU percent usage for each individual core of the CPU, as well as the average of all the cores, at 1 second intervals. This data is broken down into two figures for each trial: the behavior of each core under DDoS attack and the Total Average CPU Usage.

Windows Ping Flood Attack

Ping Class A – 16M Hosts

The experiment began the Windows Trials with a Ping Flood Attack from a Random IP Class A network. Figure II-1 shows that CPU usage for each core sits around 20% during the baseline reading. This is normal CPU usage with only normal traffic and no attack traffic being administered. Attack traffic is introduced initially at 100MBps, or 10% load. Each CPU core more than doubles in usage; the initial reading shows roughly 45% for each core. This sharp increase continues into 200 Mbps. Once 30% load attack traffic begins, the usage for each core drops to about 35-40% for the remainder of the trial. Even at max load, 1Gbps, CPU usage sits at about 30-35%. Figure II-2 more clearly indicates this plateau in the CPU usage data. The attack traffic being sent in this attack is ICMP Ping Request packets, so this behavior is likely the Operating System dropping the Request Packets once a certain threshold of CPU usage was met in order to halt the usage of resources toward processing ICMP Packets.

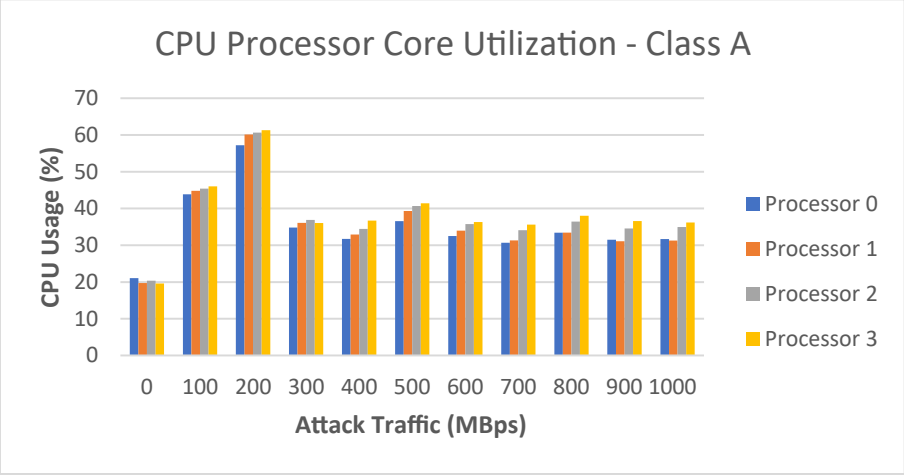

Figure II-1 Ping Flood – Class A – CPU Processor Core Usage

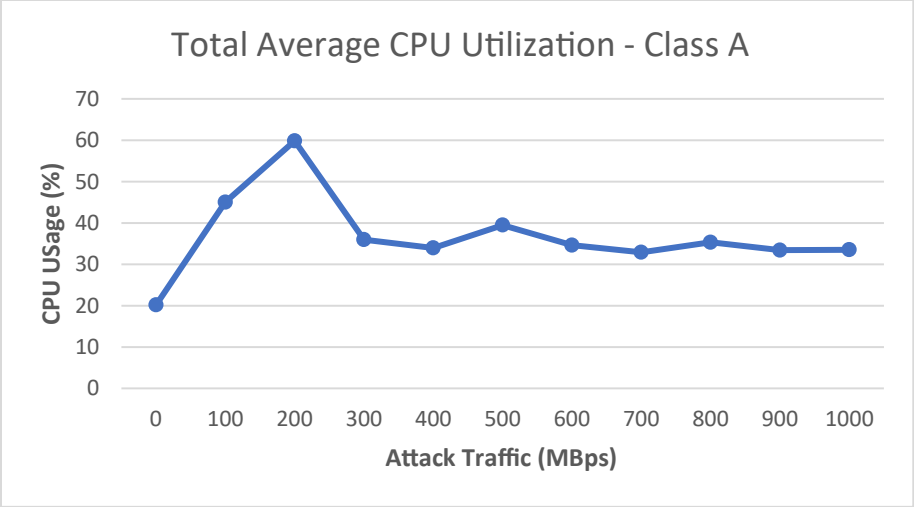

Figure II-2 Ping Flood – Class A – Total Average CPU Usage

Ping Class B – 65k Hosts

The baseline reading for Class B Address trial shows a nominal 20% CPU usage across every core. Once attack traffic has begun with 10% load, the usage rises to 35% across all CPU cores. The CPU usage spikes to over 50% once 200 Mbps attack load begins. Similar to the Class A trial, the CPU usage begins to stabilize once the 30% attack load threshold is met. This is likely for the same reason and the Operating system is dropping the traffic past a certain point. The CPU core usage rises again briefly during the 80-90% attack traffic load, but then drops back down to 25% once max attack load is reached. This is likely due to a background process that briefly increased processor core usage, as it is a consistent rise across all cores.

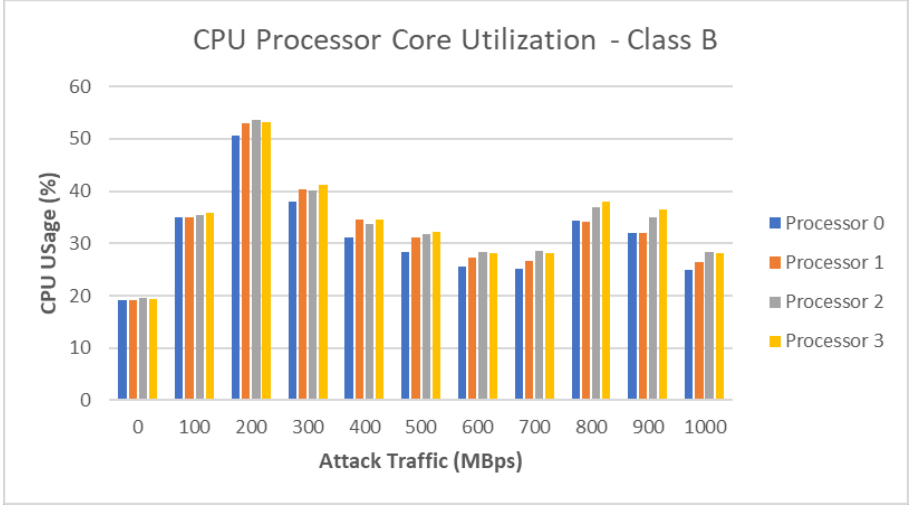

Figure II-3 Ping Flood – Class B – CPU Processor Core Usage

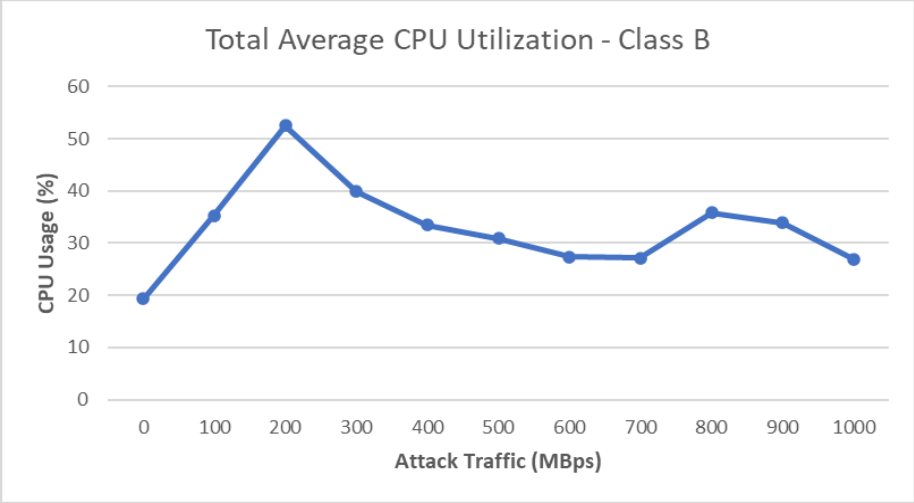

Figure II-4 Ping Flood – Class B – Total Average CPU Usage

Ping Class C – 254 Hosts

Once again, the baseline reading shows the expected 20% CPU core usage. At this point, its safe to indicate that 20% is the 'idle' CPU usage we should come to expect for the remainder of the experiment. Similarly, once the attack traffic begins, there is a large spike in CPU core

usage. This trend continues until the 20% attack load threshold, peaking at 45% CPU core usage. Once the 30% attack load is reached, the CPU core usage has already begun to drop. Once the trial reached 40% attack load, we are sitting at the normal idle traffic, 20%. After this, however, the traffic rises again and plateaus at 30% for the remainder of the trial.

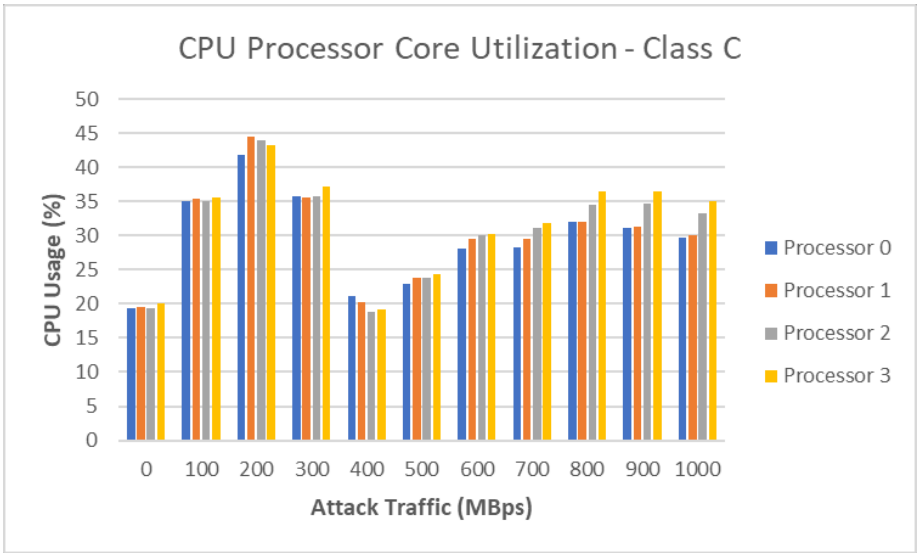

Figure II-5 Ping Flood – Class C – CPU Processor Core Usage

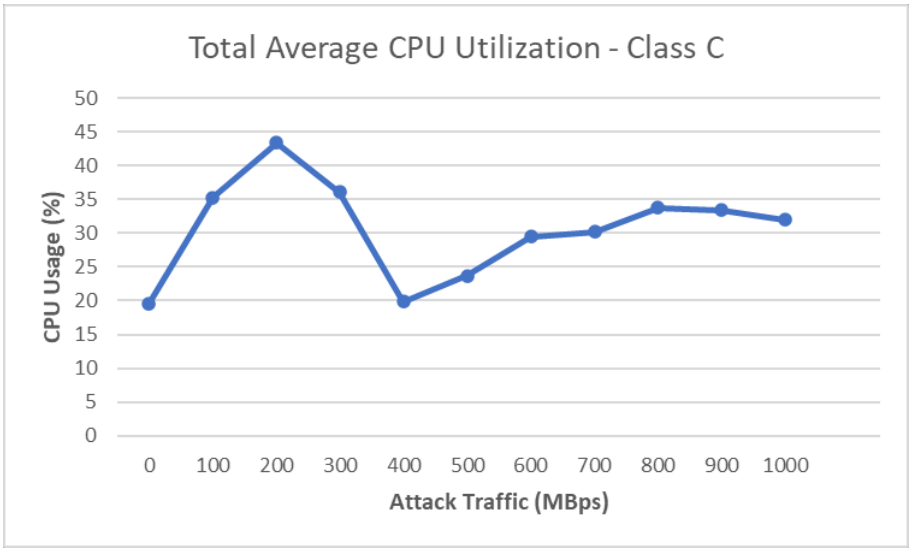

Figure II-6 Ping Flood – Class B – Total Average CPU Usage

Windows Ping Total Average CPU Utilization Comparison

Since Class A, Class B, and Class C all differ so greatly in the number of hosts available to each class, you would expect this certain difference to be reflected in the resulting plots as well. Figure II-7 shows the total CPU usage for each of the Class A, B and C trials, average across each CPU core. In every trial, the CPU usage peaked around the 20% attack load, or 200 Mbps. The Figure illustrates the trend of Class A peaking the highest with 60%, with B in the middle with 50% and C the lowest peak with 40%. Finally, all trials end at max attack load, with 30% CPU core usage. This trend persists throughout the rest of the trials. Although the number of hosts are different, each trial shows similar behavior. This may be indicative of Windows 2012R handles a large amount of ICMP packets in a very specific way; working quickly to process the packets and then drop them just as quickly when there is a very large amount of ingress ICMP traffic. This conclusion is consistent with the plot generated.

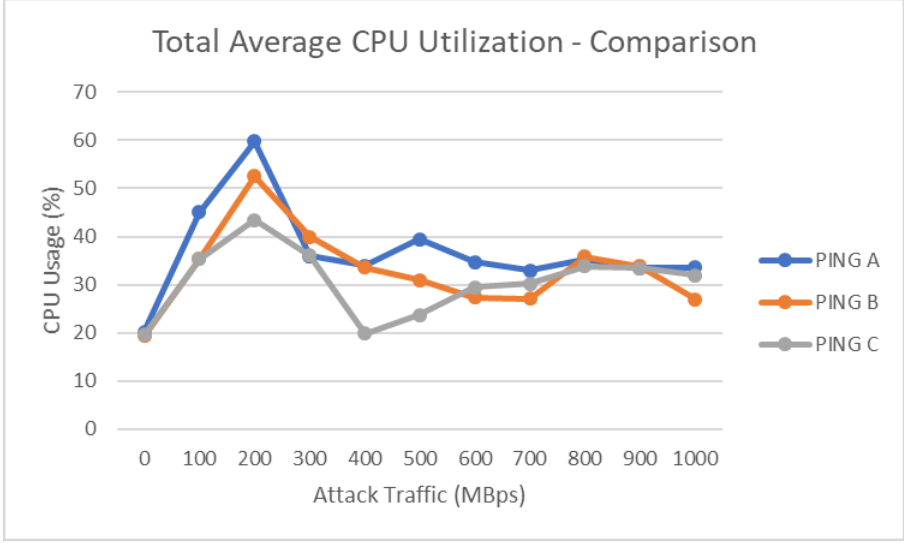

Figure II-7 Ping Flood – All Classes – Total Average CPU Usage

Windows Ping HTTP GET Transaction Loss (Estimated from Output PDF)

The following Figure illustrates the other criteria with which we are comparing Windows with Linux, the ability to process legitimate traffic while under DDoS Attack, in this case Ping Flood attack. The baseline "normal" traffic we have set for every trial in this experiment is 3000 HTTP GET transactions per second. This will be indicated at the start of every trial. Figure 8 shows that across each Ping Flood trial, the server received all transactions until about the 300Mbps or 30% attack load. After, each trial shows a rapid decline in server availability. In this experiment, below 500 HTTP GET transactions per second is considered inoperable. Ping Flood attack reached this point as early as 70% attack load, or 700 MBps, across all class trials. At max attack load, HTTP GET transactions have dropped below 200 transactions per second. So, while the target server is still operable during the attack, many legitimate users are affected by the attack in their loss of availability of the target service.

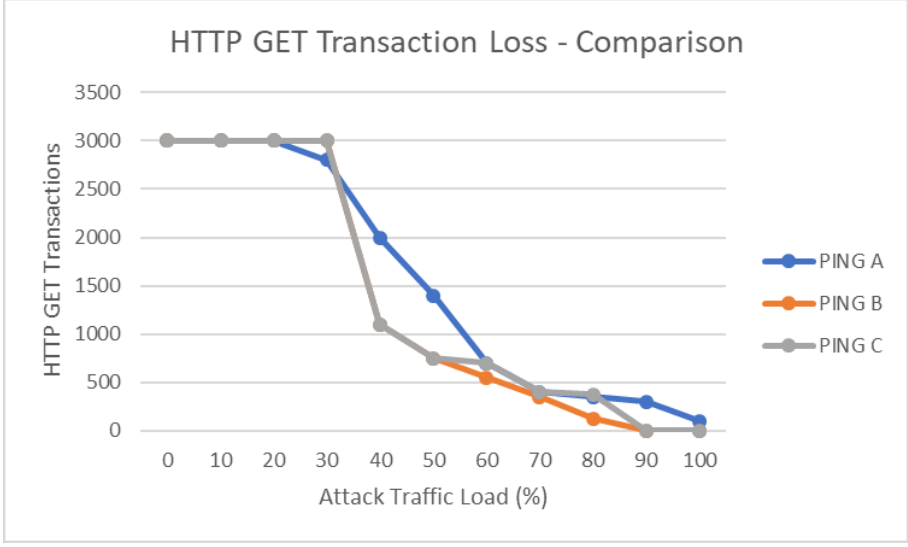

Figure II-8 Ping Flood – All Classes – HTTP GET Transaction Rate

Windows Smurf Attack

Smurf Class A – 16M Hosts

This section of the experiment begins with a Smurf Attack from a Random IP Class A network. Figure II-9 indicates that the target machine is running at a nominal 20% CPU usage in preparation for the attack traffic. Attack traffic is introduced at 10% load which immediately increases the CPU usage to almost 75%. This increase continues into 200 Mbps where CPU usages has passed 80%. Once 30% load attack traffic begins, the usage for each core rises to 90% for the remainder of the trial. This point can be considered total CPU saturation and the target machine is not usable. This occurs as early as 300Mbps, or 30% attack load, under Class A and can be seen in Figure II-10. Compared to Ping Flood, another ICMP based attack, you can see in this trial that Smurf Attacks have a much more severe effect on a target machine. This is because the attack traffic is coming from known devices so the system is less likely to drop excess traffic.

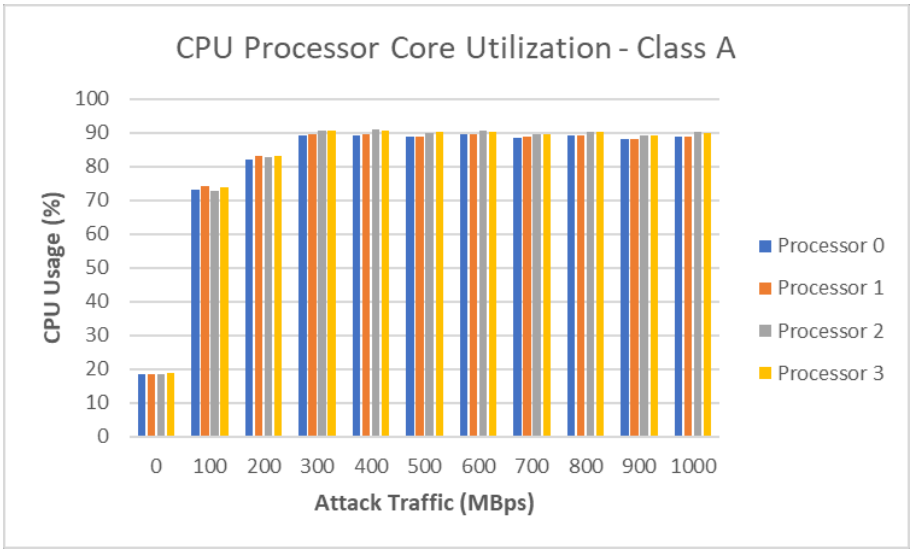

Figure II-9 Smurf Attack – Class A – CPU Processor Core Usage

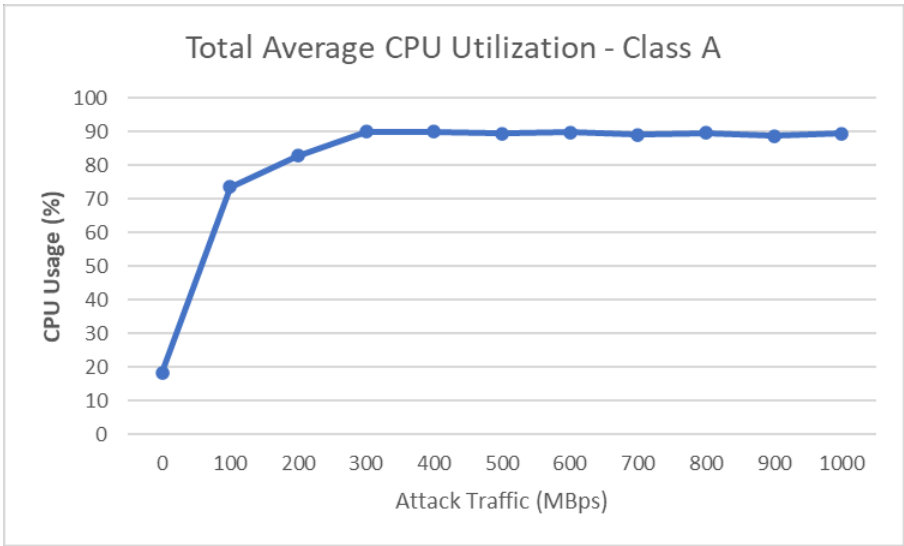

Figure II-10 Smurf Attack – Class A – Total Average CPU Usage

Smurf Class B – 65k Hosts

The baseline reading for Class B Address trial shows 20% CPU usage. Once attack traffic has begun, the usage jumps to 70%. Over the rest of the trial, the CPU usage rises to 90% and remains there. Once the attack reaches 300 Mbps, the system is at full saturation. Similar to the Class A trial, the CPU usage increases to the point of inoperability from a relatively low attack load. Class B still has a significant number of available hosts, but very short of the amount Class A. However, the attack is just as severe. So, under Smurf DDoS attack, a successful attack doesn't require millions of hosts. Several thousand hosts are enough to fully saturate the target CPU.

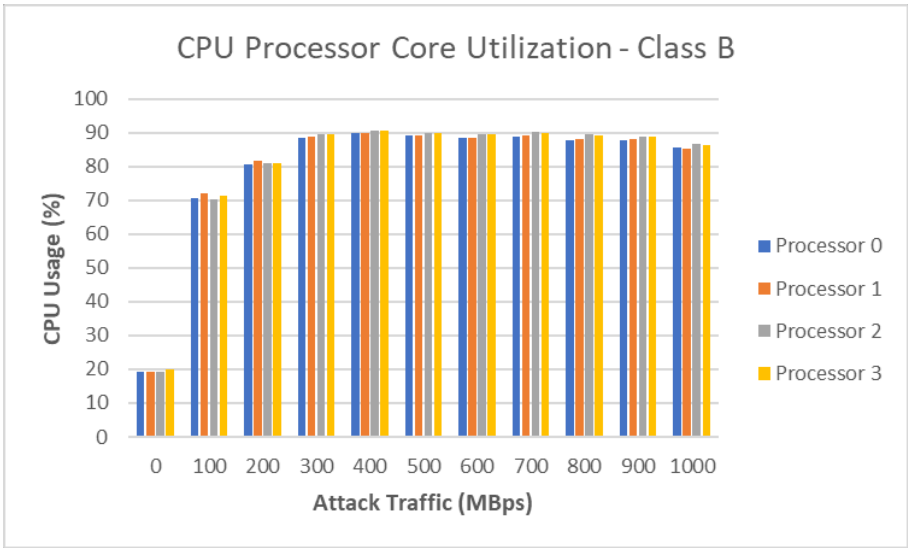

Figure II-11 Smurf Attack – Class B – CPU Processor Core Usage

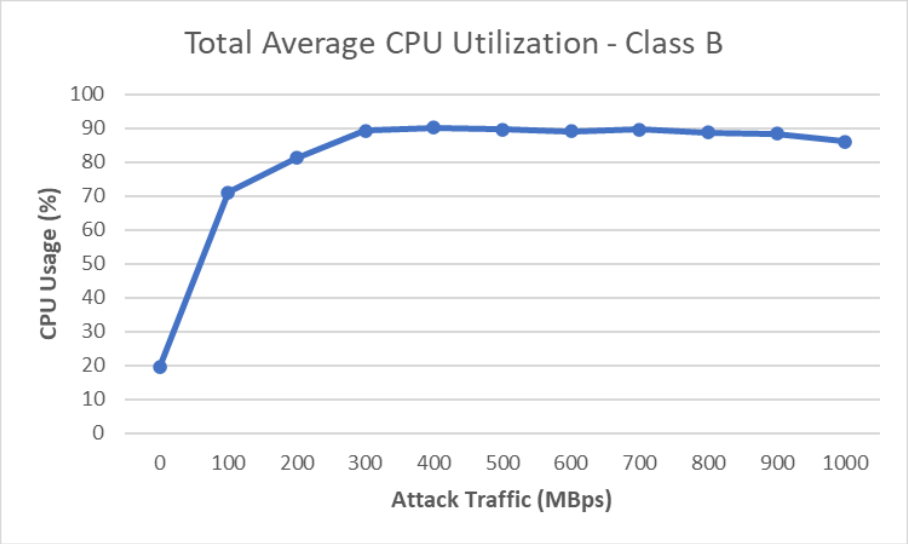

Figure II-12 Smurf Attack – Class B – Total Average CPU Usage

Smurf Class C – 254 Hosts

In this trial the baseline reading shows an above average 30% CPU core usage. However, this change is not large enough to signify anything out of the ordinary so this will be considered normal background process interference. Similar to previous Smurf Attack trials, once the attack traffic begins there is a rapid increase in CPU core usage. The initial jump at 100Mbps is more than double the normal traffic CPU usage. Once 300 Mbps is reached, the CPU usage is at 90%. This trend continues until the max load of attack traffic where we get a small drop to 70% usage. This is likely due to Class C having the smallest number of available hosts and once the attack persists for a period, the attack traffic begins to be dropped. Smurf attacks can be devastating to a target system. As few as 256 hosts can cause CPU usage to reach maximum capacity.

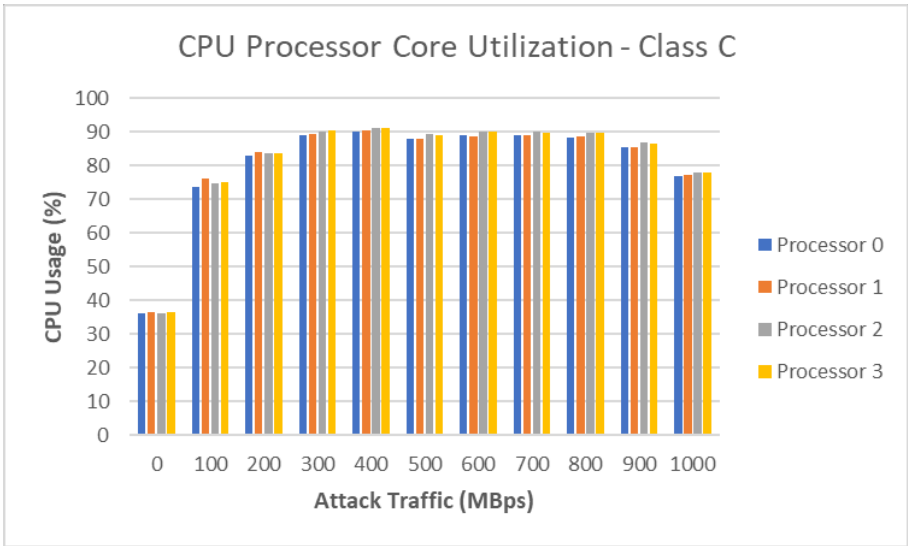

Figure II-13 Smurf Attack – Class C – CPU Processor Core Usage

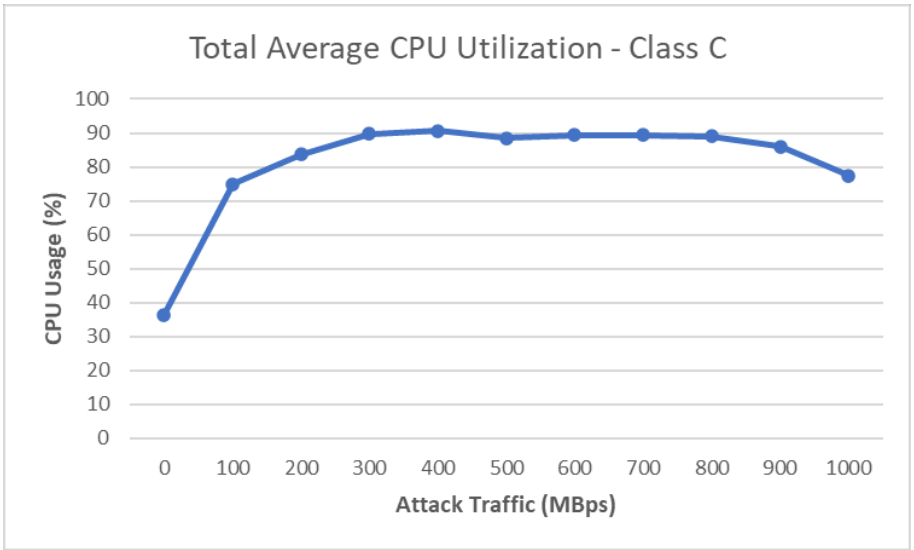

Figure II-14 Smurf Attack – Class C – Total Average CPU Usage

Windows Smurf Total Average CPU Utilization Comparison

Figure II-15 shows a comparison between all Class trials under Smurf Attack. In this comparison, it is not so easy to discern which attack was the most significant. Regardless of the size of the botnet, the results indicated that Smurf attacks can be devastating if performed properly. All trials show normal CPU usage in the baseline phases with the first large jump at 100Mbps and shortly after at 300Mbps, the target device was already at 90% CPU usage. The target remained at 90% for the rest of the trial. However, Class B and Class C show a small decrease once max load of 1Gbps is achieved. Regardless of this, the device is still at 70% under Class C attack and that is still quite high. Although this type of attack can easily lock up a target device, it is simple to mitigate this type of attack by configuring your device to filter out excessive ICMP Requests.

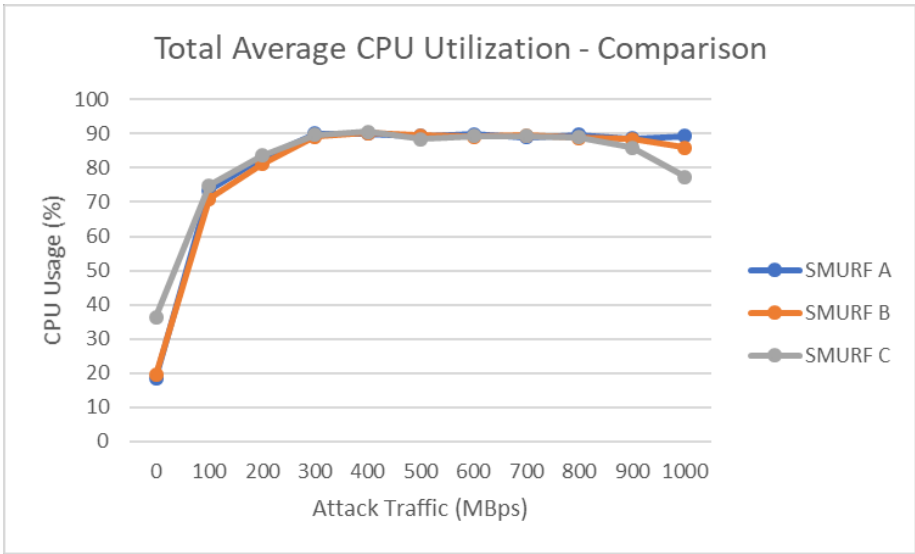

Figure II-15 Smurf Attack – All Classes – Total Average CPU Usage

Windows Smurf HTTP GET Transaction Loss (Estimated from Output PDF)

Figure II-16 shows the HTTP GET transaction rate comparison for the Smurf Flood trials. Initially, we have the normal 3000 HTTP GET packets per second rate of traffic. Figure II-16 shows that across each Smurf attack trial, the server received normal traffic levels until about the 200Mbps mark. It was at this point there was a sharp decline in server availability. By 300 Mbps, half of HTTP connections were lost. By 400 Mbps, the amount of successful transactions was less than 200 and effectively 0 for the remaining time. A Smurf attack was able to prevent availability to the target service, regardless of the botnet size, and at relatively low attack load.

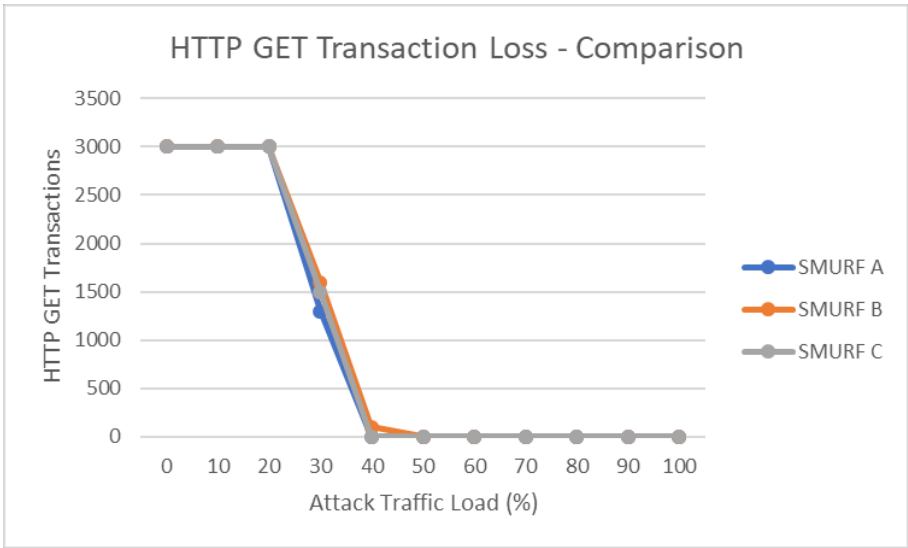

Figure II-16 Smurf Attack – All Classes – HTTP GET Transaction Rate

Windows TCPSYN Attack

TCPSYN Attack Class A – 16M Hosts

This section features Transport Layer attacks, or Layer 4 Attacks. Figure II-17 shows that the target server is running at 20% CPU usage with no attack traffic introduced. This is consistent with previous results and the background interference is now absent. Attack traffic is introduced at 100 Mbps and the target CPU core usage jumps to 80%. This is where the usage level remains for the rest of the Class A trial. At 80%, every CPU core is being utilized almost to max and the target machine is not usable. Under TCPSYN Class A attack, the target system is disrupted as soon as attack traffic is introduced. When compared to the previous Layer 3 attacks, what this attack lacks in severity it makes up for in consistency. This is likely due to Layer 4 protocols relying on Layer 3 protocols, such as TCP utilizing IP. This is why TCP is referred to as TCP/IP. Since TCP is connection oriented, requests for handshake are not dropped as readily as ICMP messages. Since TCP is working off IP, these packets are more resource intensive to process.

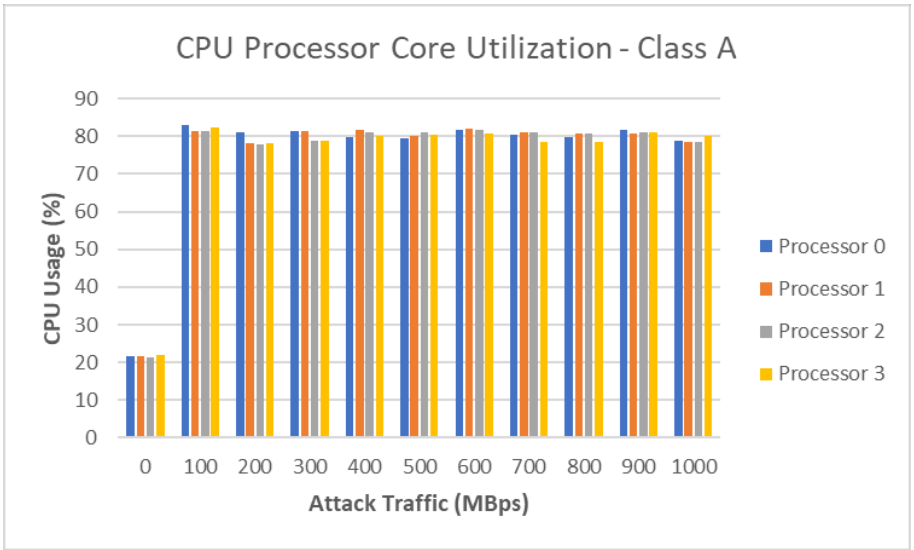

Figure II-17 TCPSYN Attack – Class A – CPU Processor Core Usage

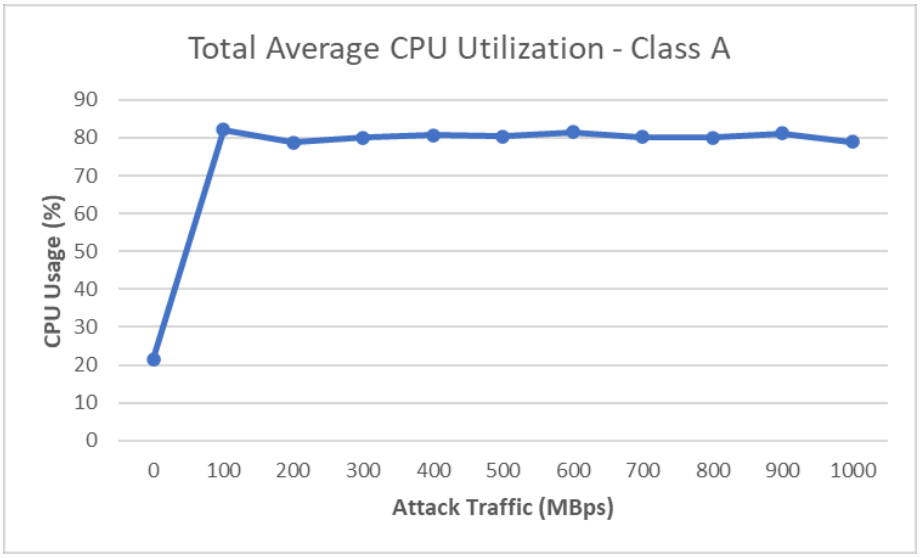

Figure II-18 TCPSYN Attack – Class A – Total Average CPU Usage

TCPSYN Attack Class B – 65k Hosts

The baseline for Class B Address trial shows the expected 20% CPU usage. Once attack traffic in introduced, the usage jumps to 60%. At 200 Mbps attack load, the usage rises to 70%. Once the attack reaches 300 Mbps, the CPU core usage rises over 80% and averages around 85%

for the remainder of the test. This trial showed a similar trend to TCPSYN class A as the saturation point is about 80-85%. However, Class B addressing did not reach this point as quickly as Class A. This trial reached the 80% mark around 300 Mbps while Class A reached it as soon as attack traffic was introduced.

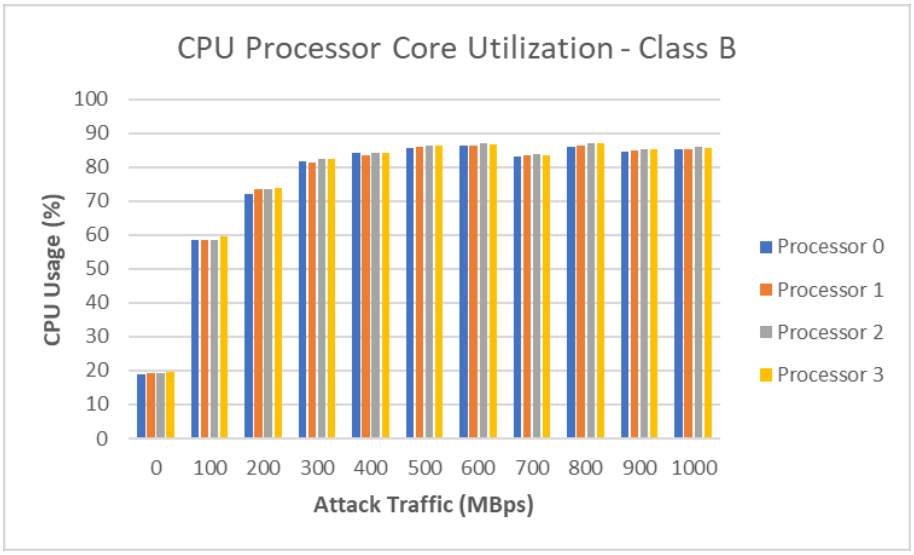

Figure II-19 TCPSYN Attack – Class B – CPU Processor Core Usage

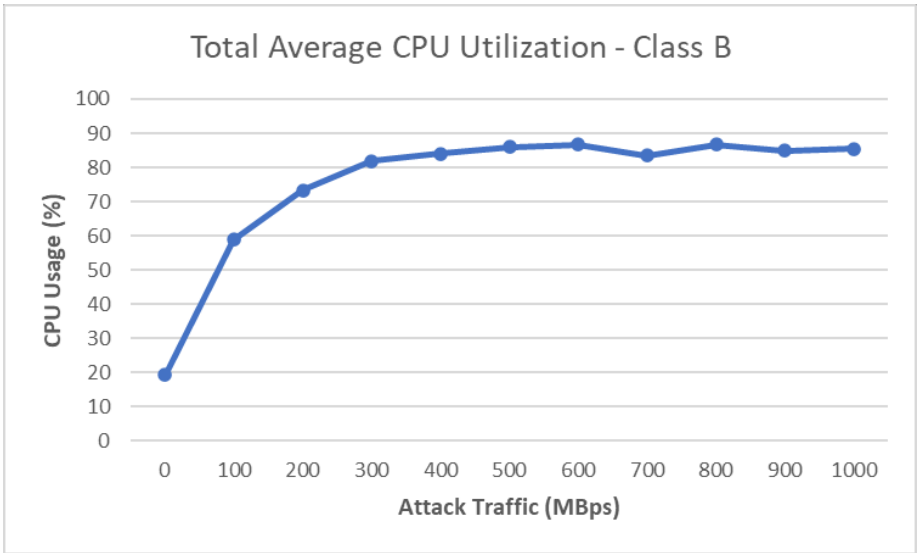

Figure II-20 TCPSYN Attack – Class B – Total Average CPU Usage

TCPSYN Attack Class C – 254 Hosts

Figure II-21 and II-22 show the expected 20% normal CPU usage. As attack traffic is introduced, you see a small increase in the CPU usage. At 100 Mbps, the plot shows a little over double the normal CPU usage levels a 40%. At 200 Mbps, the CPU usage spikes at a little under 60% CPU usage. However, after this initial spike in CPU usage, the levels drop down to 35-40%. For the remainder of the trial, the CPU core usage is sustained at 40% CPU usage, including max load. This trial is indicative of the effect the size of the botnet has on the DDoS attack. The initial test with 16M+ hosts had instant CPU saturation, with Class B functioning briefly before CPU saturation. This trial shows the target machine is still operable for the duration of the attack. So, if the intention is to interrupt the host with a TCPSYN attack, more hosts would yield better results.

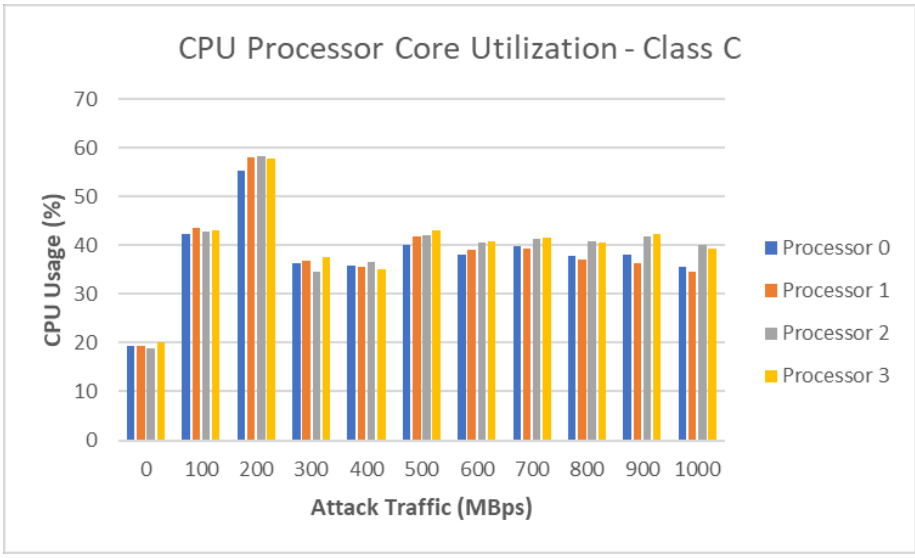

Figure II-21 TCPSYN Attack – Class C – CPU Processor Core Usage

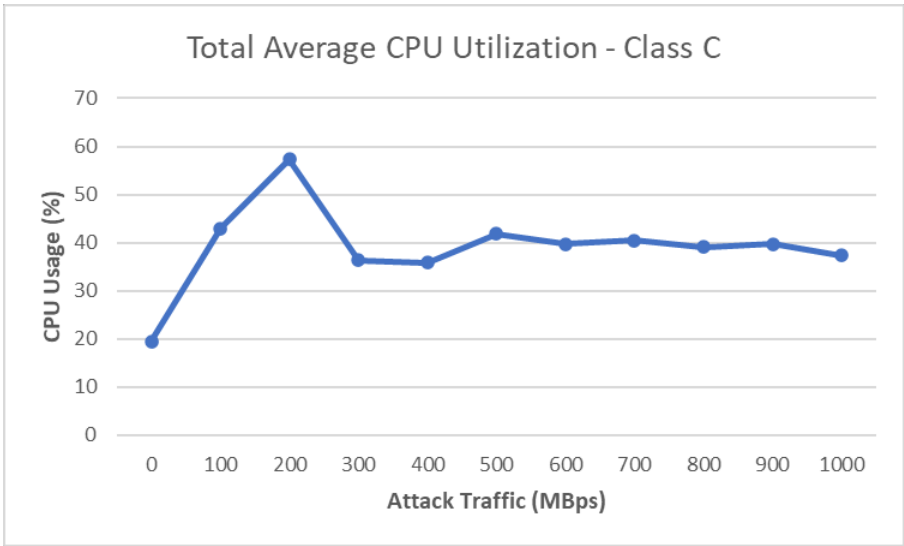

Figure II-22 TCPSYN Attack – Class C – Total Average CPU Usage

Windows TCPSYN Total Average CPU Utilization Comparison

Figure II-23 shows a comparison between all Class trials under TCPSYN Attack. At first glance, you can clearly indicate the individual trend between each class. More so than with the previous Smurf trials. Unlike those trials, the size of the botnet has an effect on the results of the TCPSYN attack for every class. All trials show normal CPU usage in the baseline phases with Class A showing CPU saturation at 10% attack load. Class B follows a similar trend but doesn't reach saturation under 30% attack load. Class C attack doesn't lock up the CPU like the previous trials, however. For the length of the attack, CPU core usage peaks briefly and then maintains a 40% CPU core usage, well within usable parameters.

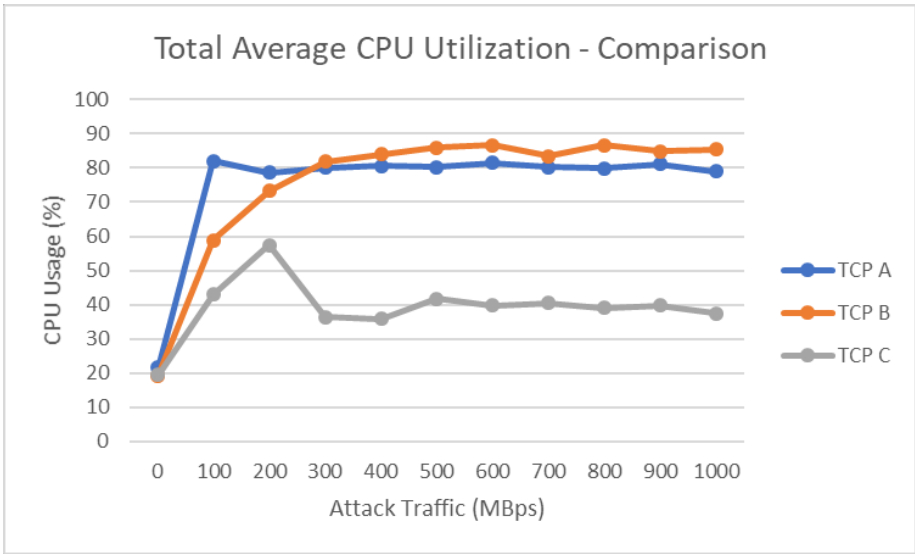

Figure II-23 TCPSYN Attack – All Classes – Total Average CPU Usage

Windows TCPSYN HTTP GET Transaction Loss (Estimated from Output PDF)

Figure 24 shows the HTTP GET transaction rate comparison for the TCPSYN Attack trials. Initially, the normal 3000 HTTP GET packets per second rate of traffic is measured. The Figure shows that during each TCPSYN trial, the server ranged from operable to complete loss of availability. Class A trial shows that once attack traffic was introduced, the server was unable to process any new legitimate connections. Class B shows normal operation until 300 Mbps, where there is a large decline in connections. By the time 400 Mbps is reached, less than 500 connections are being established. Soon after at 500 Mbps, availability is lost. During the Class C trials, the HTTP connection data shows a gradual decline in availability of the server. Once the trial reaches max load however, less than 500 HTTP connections can be made. The results of these trials are an indicator of the effect of botnet size for Layer 4 DDoS Attacks.

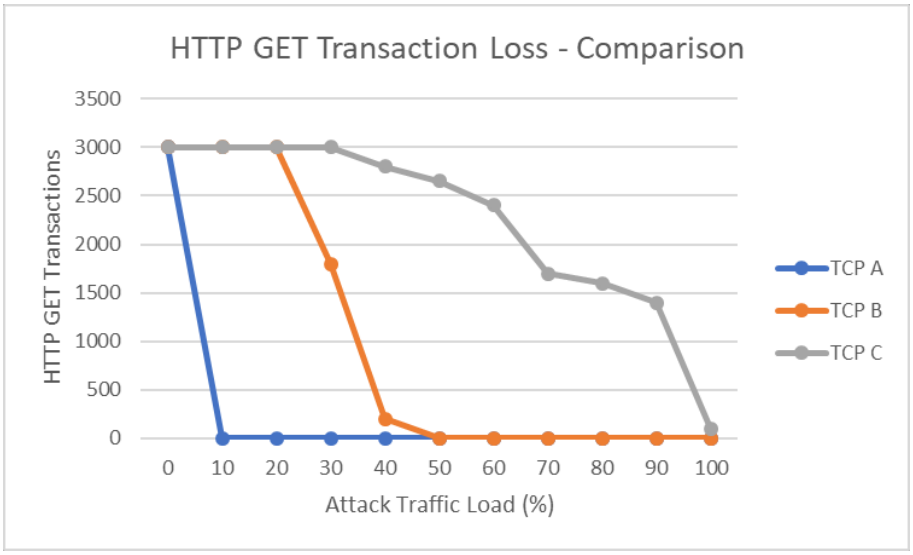

Figure II-24 TCPSYN Attack – All Classes – HTTP GET Transaction Rate

Windows UDP Flood Attack

UDP Flood Class A – 16M Hosts

Typically, a UDP flood involved ICMP Destination Unreachable messages as well. However, this requires the UDP port to be closed. In CentOS, these messages are not sent and UDP datagrams sent to the closed port are dropped. For this reason, these trials will be conducted with the UDP port OPEN.

Figure II-25 shows that the target server is running at normal CPU usage before the attack traffic begins. Attack traffic is initiated starting with 100 Mbps and the target CPU core usage immediately rises to 60%. Once 200 Mbps traffic begins, the CPU usage rises and approaches 70%. At 300 Mbps, the usage drops back down to 60% and stays between 60% and 70% for the remaining time. Even at max load, the CPU core usage is nearly 65%. Under UDP Class A attack, the target system is not entirely disrupted but will exhibit poor performance. Opposed to TCPSYN's handshake method of connection, UDP requires no acknowledgement

and are of lower importance; they may be lost or dropped. This would explain why compared to TCPSYN 80-90% level of CPU saturation, UDP only reaches about 65%. There is likely a specific capacity to process UDP datagrams before they are dropped.

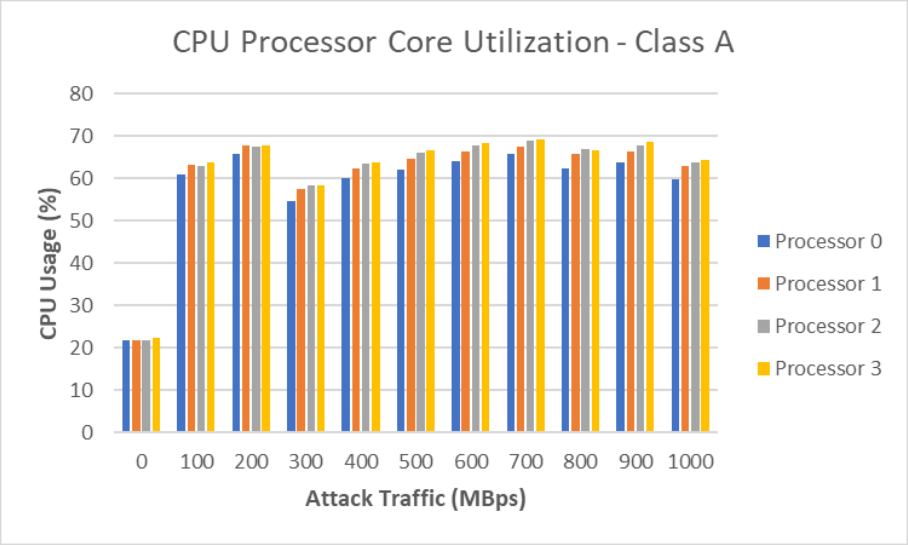

Figure II-25 UDP Flood – Class A – CPU Processor Core Usage

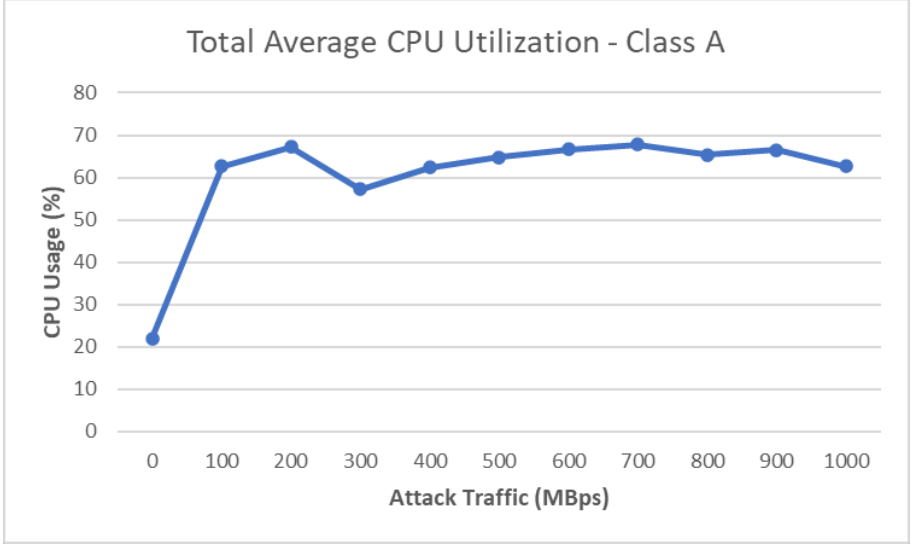

Figure II-26 UDP Flood – Class A – Total Average CPU Usage

UDP Flood Class B – 65k Hosts

The baseline shows the expected 20% CPU usage consistent with each trial. Once the attack traffic begins, the CPU core usage rises to around 60%. The CPU usage stays consistently at 60-65% until the 600 Mbps mark where it peaks at 70%. The CPU core usage stays 60-65% for most of the test time, very similar to Class A. This further indicates the possibly of a capacity as, regardless of the size of the botnet, CPU usage remains unchanged. Must like the previous trial, the moment attack traffic is introduced, the CPU reaches a point where performance would be poor, but availability would continue.

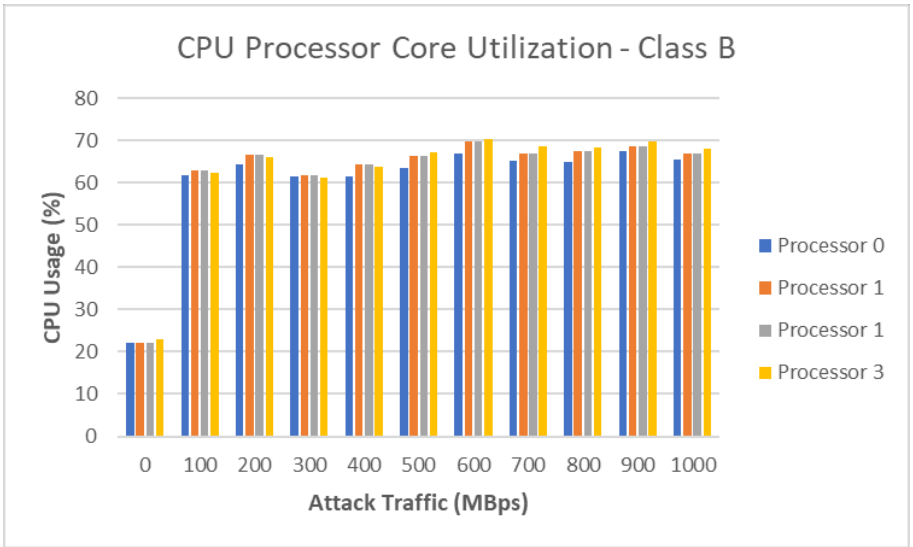

Figure II-27 UDP Flood – Class B – CPU Processor Core Usage

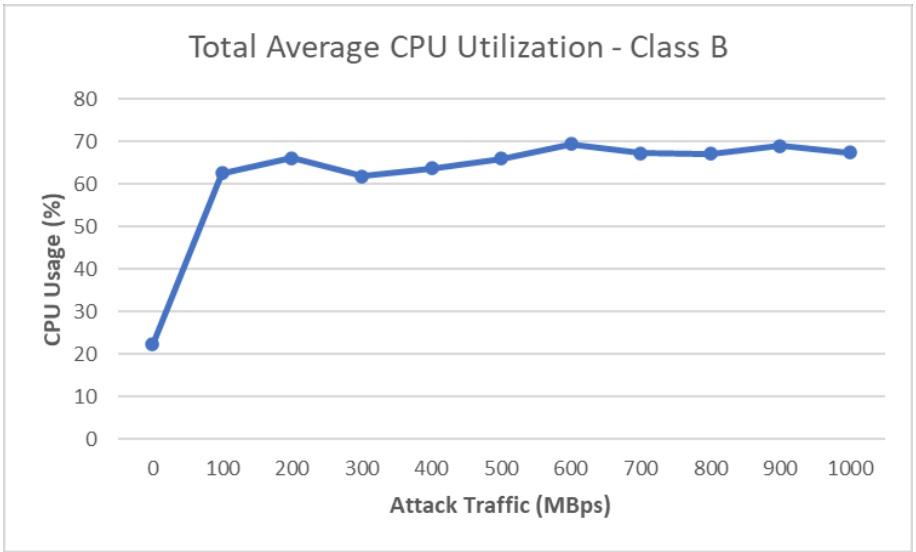

Figure II-28 UDP Flood – Class B – Total Average CPU Usage

UDP Flood Class C – 254 Hosts

The following Figures indicate the expected 20% normal CPU usage initially. Once the attack traffic begins, just like the previous trials, the CPU core usage rises to about 60%. At 200 Mbps, the CPU usage spikes to around 65% CPU usage. After, for the remainder of the trial, the CPU core usage continues to stay at around 60-65%. At max load, the CPU core usage peaks at 70%. The results of Class C confirm that under UDP attack, CPU usage will reach a certain capacity and begin dropping UDP packets to prevent resource overconsumption. Regardless of the botnet size, whether 16 million or 254 hosts, UDP flood attack reach a certain threshold where service performance suffers but availability is not lost.

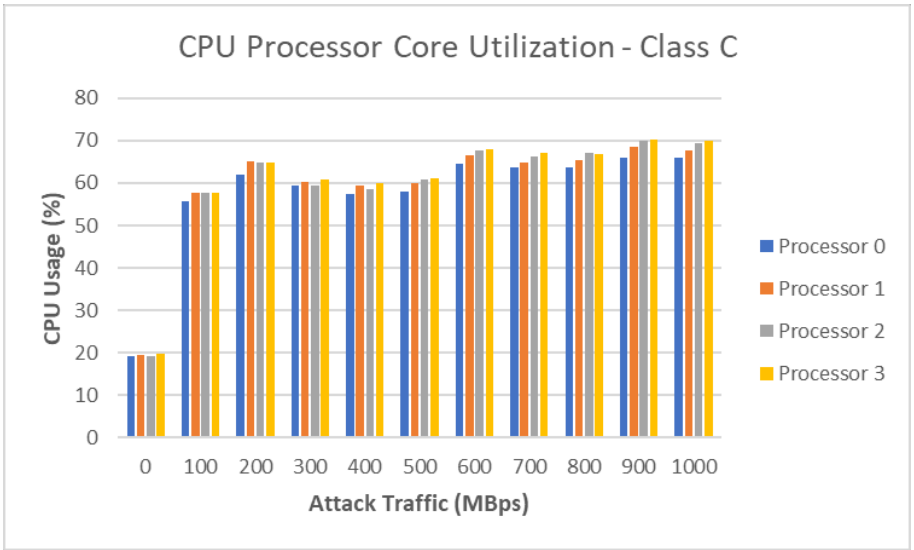

Figure II-29 UDP Flood – Class C – CPU Processor Core Usage

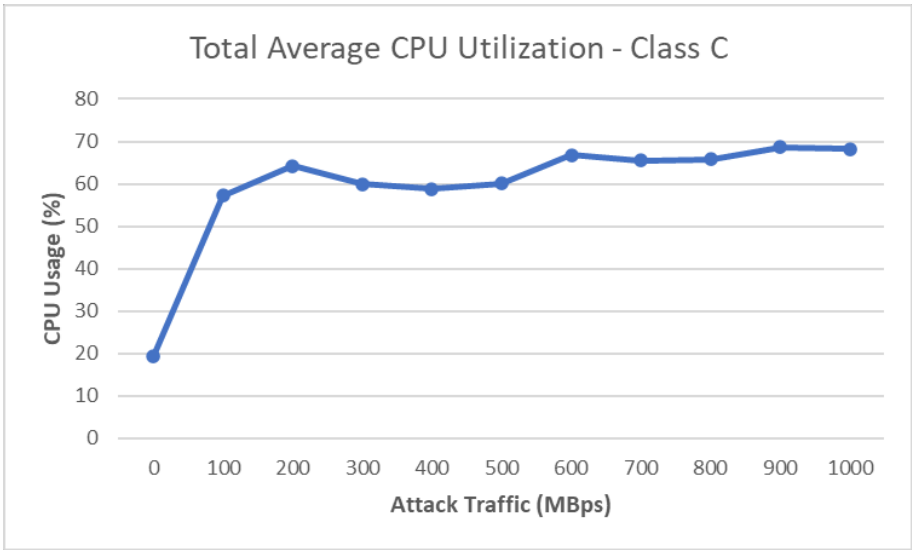

Figure II-30 UDP Flood – Class C – Total Average CPU Usage

Windows UDP Total Average CPU Utilization Comparison

Figure II-31 shows a comparison between all Class trials under UDP Flood. Throughout the UDP trials, it was hypothesized that UDP Flood attacks are handled by limiting the amount

of UDP datagrams that can be processed at once and dropping the rest. This seems likely to be the case according to the Figure below. Performance under UDP Flood attack is very similar, regardless of the size of the botnet carrying out the attack. All 3 Class trials shows CPU usage in the 60-70% range for the duration of the attack traffic introduced in each trial. Although, 70% CPU usage will result in the target service suffering in performance, it is not enough to lock up the target server.

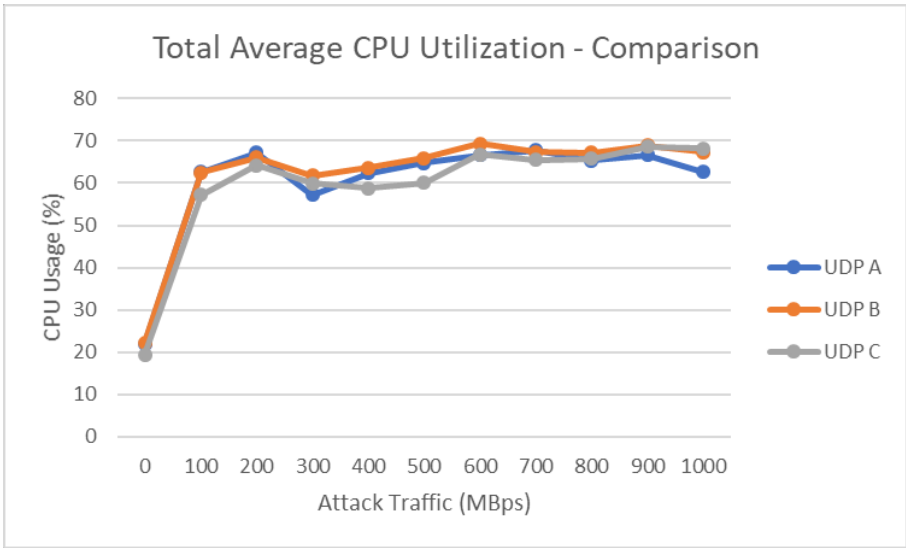

Figure II-31 UDP Flood – All Classes – Total Average CPU Usage

Windows UDP HTTP GET Transaction Loss (Estimated from Output PDF)

Figure II-32 shows the HTTP GET transaction rate comparison for the UDP Flood trials. To begin, the baseline 3000 HTTP GET packets per second rate of traffic is noted. Although the UDP Flood effected the CPU similarly across all classes, the effect on the availability of the service was entirely unique. All class trials show normal traffic until about 20% attack load. Class A trial shows a rapid drop in connections at 30% attack load, a slow decline in available connections during the attack, and finally, complete loss of availability at 80% attack load. Class

C shows similar behavior to Class A only the decline in connections during the attack occurs slower and loss of availability occurs at 90% attack load. Class B trial results indicate uptime throughout the attack with less than 500 connections available only occurring at max load. At 90% attack load, nearly half connections are still available.

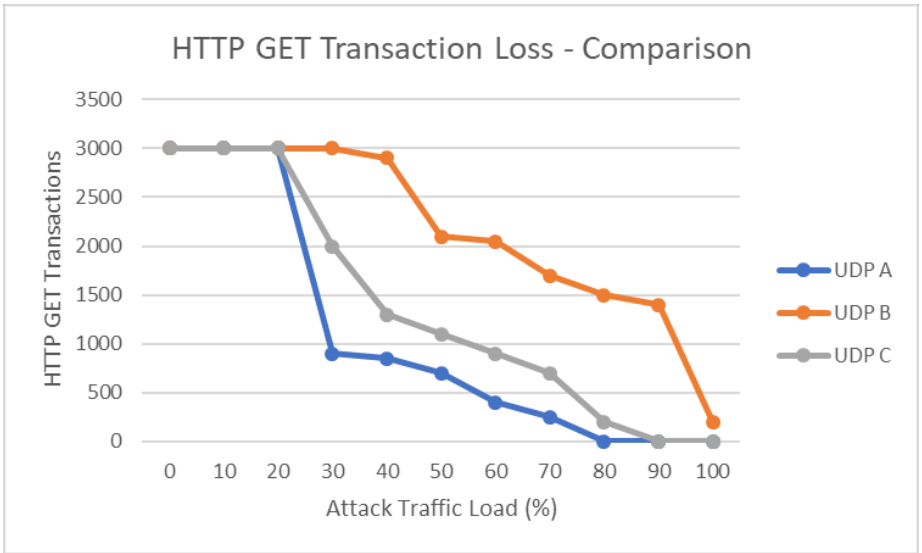

Figure II-32 UDP Flood – All Classes – HTTP GET Transaction Rate

CHAPTER III

CENTOS OPERATING SYSTEM PERFORMANCE UNDER DDOS ATTACK

Once the set of Windows Server trials in the previous section were concluded, the target machine was formatted with a fresh installation of CentOS 15. The version used is readily available on their website as a typical installation or a minimal installation. The typical installation contains many libraries and prepackaged software that you might expect in a typical installation of Windows, such as security software, word procession software, a GUI, etc. The minimal installation is only command line and some basic packages required by the Linux kernel. This version is intended for the seasoned veteran Linux user, or those who may want to keep their footprint as small as possible, while using the same CentOS kernel. The typical installation may come with software some user may not want, so the minimal installation would be preferable as in most cases it is easier to start at the bottom and build whatever it is the task ahead may require. In order to keep the results of this experiment as consistent as possible, the typical installation was used for the following set of trials.

After the Operating System was installed, the firewall needed to be configured to allow the HTTP transaction traffic, as well as the attack traffic. This server software used is Apache, which is readily available in the CentOS software repository. Once this server was created and configured, the final problem was recording the data required by the experiment outline, the CPU core usage data and the HTTP GET transaction data. CentOS does not have a Data Collector Set

software, so this required some creativity. The software ‘htop’ was used in this experiment. For all intents and purposes, this was used as an equivalent to Windows Task Manager; solely to monitor the amount of ingress traffic to determine if any traffic was currently being received by the target machine. This software allowed for each CPU core to be monitored as well. However, this software did not allow me to export this data. In my research I discovered ‘SYSSTAT’, this software was key in outputting the required data for this experiment. With this software, I was able to export data in a comma separated value, semicolon delimited, Excel file. This recorded each CPU Core as well as the average CPU Core usage.

The CentOS section of this experiment follows the same outline of the Windows Server section. The initial trials cover Layer 3 attacks and the later Trials cover Layer 4 attacks.

CentOS Ping Flood Attack

Ping Flood Class A – 16M Hosts

Figure III-1 shows the behavior of each processor core during the attack. The baseline information for each trial remained essentially the same; with no attack traffic introduced, the normal traffic puts CPU Core usage around 20%. Once the attack traffic is introduced, you see an initial jump, followed steady rise in CPU usage. This rise is consistent with each increase in attack traffic, roughly 10-15% at each increase. CPU Core usages maxes at 400MBps or 40% attack traffic. It is here where we first see the CPU core usage reach 80%, the highest point in the trial. It remains at 80% for the rest of the trial. At 100% attack traffic load, or 1000Mbps, the CPU does drop, but only by 5%. This is attributed to the Operating System likely trying to mitigate some of the ingress attack traffic. The initial trial was very promising as it shows the

data collection software is working well in conjunction and data was recovered that is comparable to the Windows Server data.

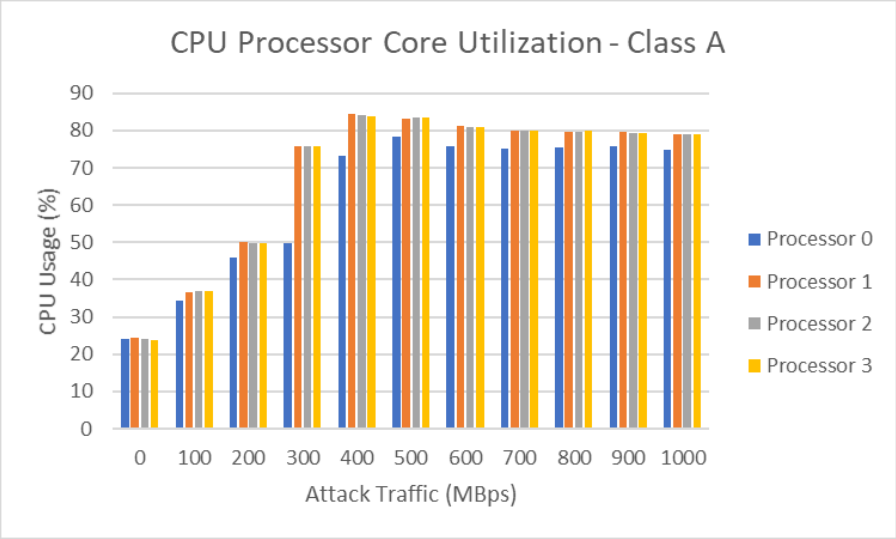

Figure III-1 Ping Flood – Class A – CPU Processor Core Usage

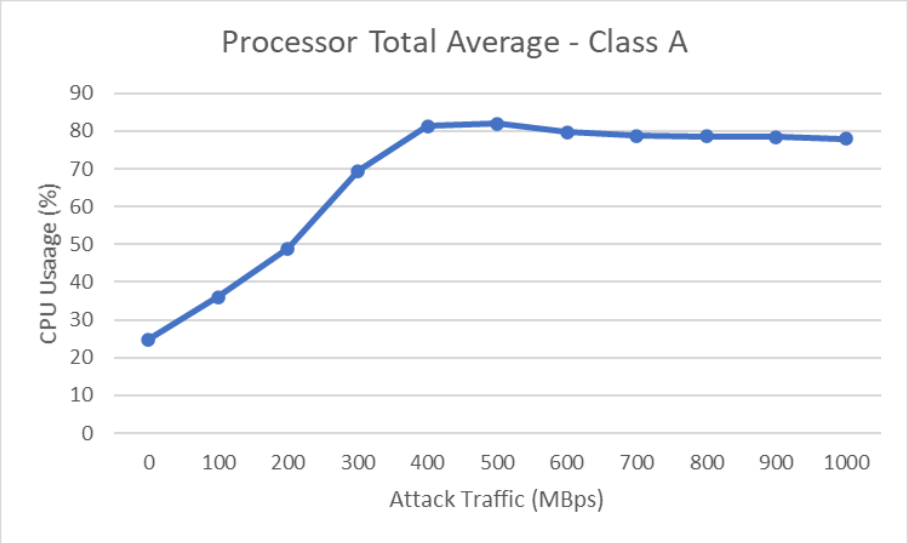

Figure III-2 Ping Flood – Class A – Total Average CPU Usage

Ping Flood Class B – 65k Hosts

Figure III-3 shows the CPU Core usage for the second trial of Ping Flood, Class B. This is the first trial to show an uneven load on the CPU Core usage information. Initially, we see the 20% CPU Core Usage with no attack traffic. The load appears to be marginally uneven but negligible. However, once attack traffic load reaches 200 Mbps, we start to see Processor Core 1 start to fall behind. This becomes more evident in the rest of the experiment. It happens to multiple cores, not just Processor Core 1. This is due to the different scheduling scheme packaged in CentOS, known as CFS, or Completely Fair Scheduling.[97] Processor 1 is much lower than the other processors, which are all higher than they would normally be. The average is still taken to get an accurate sample. This information is more clearly presented at the end of the Ping Flood section, where all classes are compared.

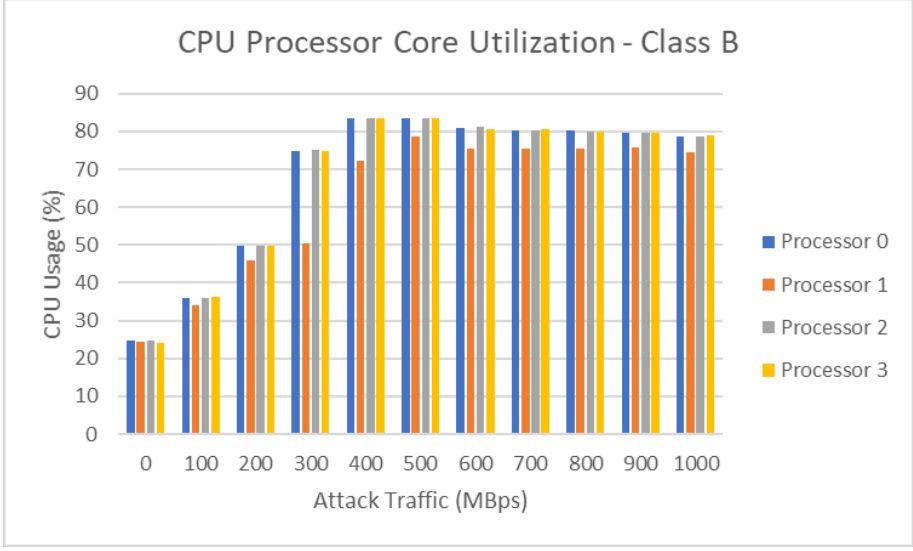

Figure III-3 Ping Flood – Class B – CPU Processor Core Usage

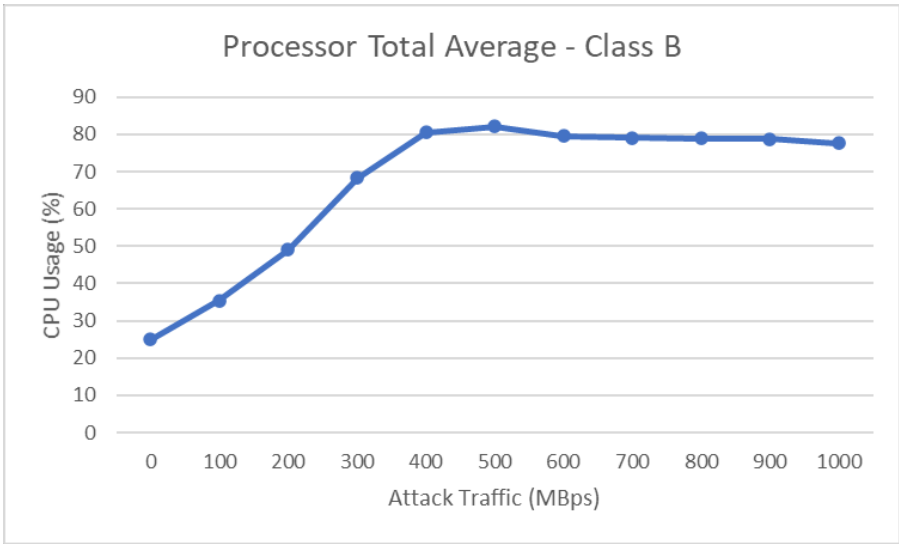

Figure III-4 Ping Flood – Class B – Total Average CPU Usage

Ping Flood Class C – 254 Hosts

Figure III-5 shows the CPU core behavior witnessed in the previous section as well. This further cement the concept that the Linux Kernel distributes the attack traffic load differently than Windows Server. The initial 20% CPU core usage is visible in this trial as well. This trial shows a steady rise in CPU core usage, again, maxing out at nearly 80%. This, however, occurs later in the experiment around the 700Mbps mark, illustrated in Figure III-6.

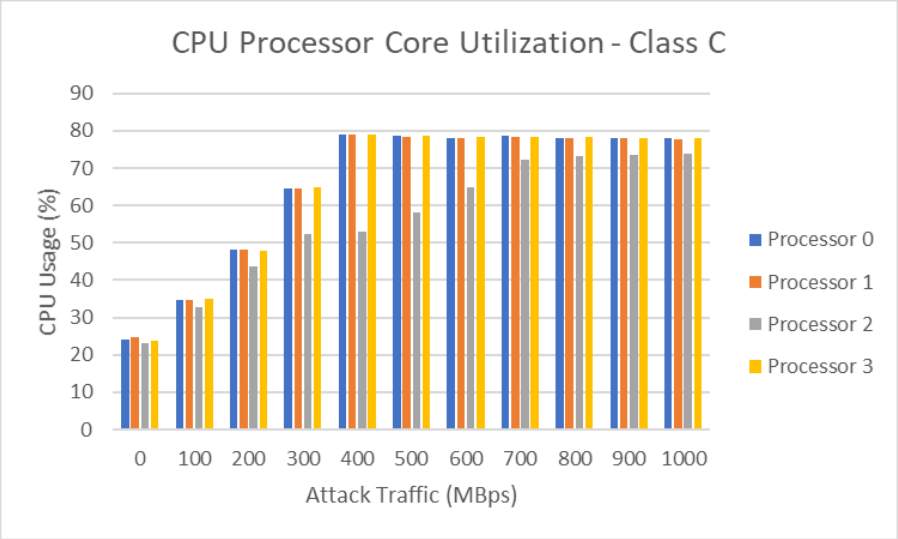

Figure III-5 Ping Flood – Class C – CPU Processor Core Usage

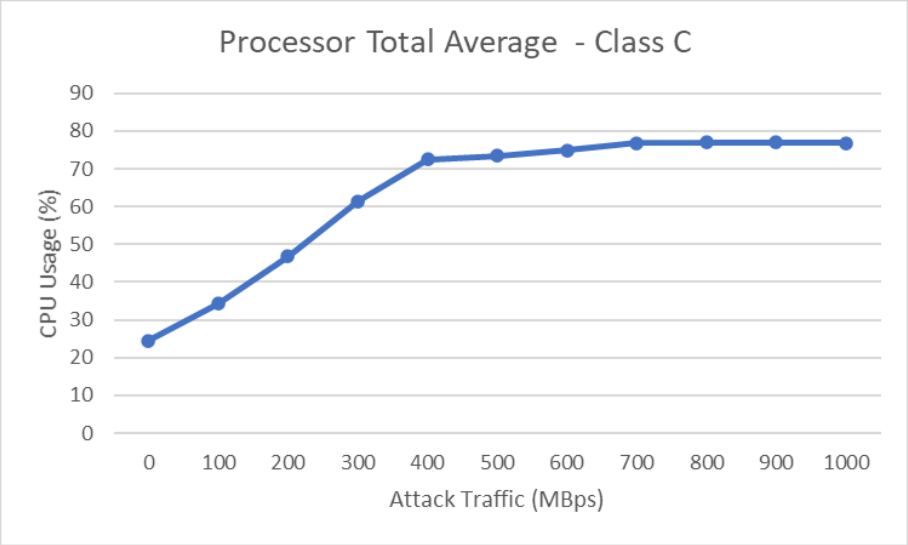

Figure III-6 Ping Flood – Class C – Total Average CPU Usage

CentOS Ping Processor Total Average Comparison

Figure III-7 shows the results from the Ping Flood attack against the CentOS target service. At 0% attack load, the nominal traffic match those of the previous attacks, roughly 25%

of the total average CPU usage. Each Classes' results show no more than a 5-10% difference, so any of these abnormalities are at the fault of other services. In this attack, we reach the saturation point as soon as 400Mbps, 40% attack load. Here we see saturation upwards of 80% for most of the attack, with the higher attack loads dropping to the typical 78%, still quite high and effectively rendering the service unavailable. As previously mentioned, although Class B CPU core usage reflected different levels of usage when compared to Class A, the total average CPU usage results are so close, they seem to be identical. So, although one Core seemed to be processing fewer threads, the other cores seem to process more, so the average remains the same.

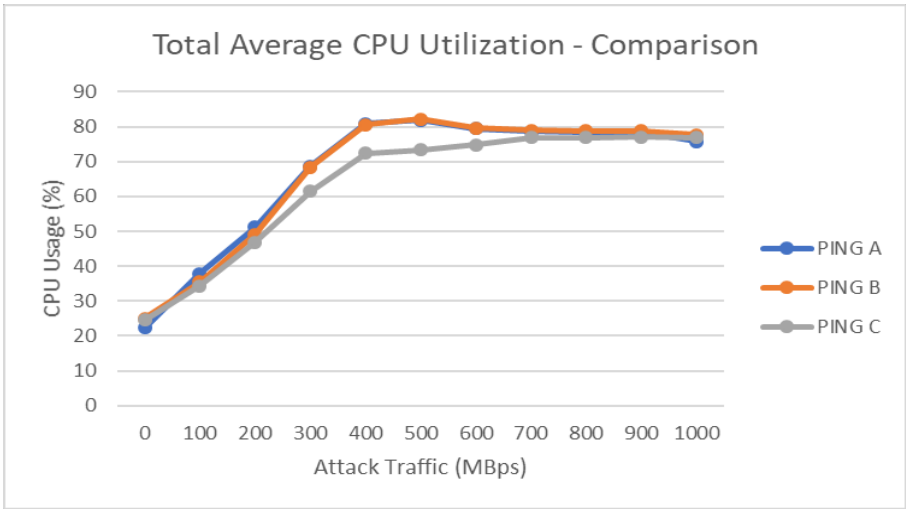

Figure III-7 Ping Flood – All Classes – Total Average CPU Usage

CentOS Ping HTTP GET Transaction Loss (Estimated from Output PDF)

The HTTP GET transactions shown in Figure III-8. The loss of normal traffic begins at about 300Mbps which has been consistent across all attacks. After, we see a steady decline in available connections. As early as 600Mbps, the threshold for total loss is reached and very few legitimate connections can be made under Ping Flood attack. At the max attack load, less than 100

connections can be made under attack. At 500Mpbs attack load, before the point of saturation, the system can maintain around 1300-1400 connections per second. This is more connections that Layer 4 attacks at the same points. Although those protocol take more action to secure the packets being sent, this overhead adds up and leads to more losses due to inability to process legit traffic as well as attack traffic. However, since ICMP attacks require less resources to maintain, these types of attacks could effective be kept up for longer periods of time.

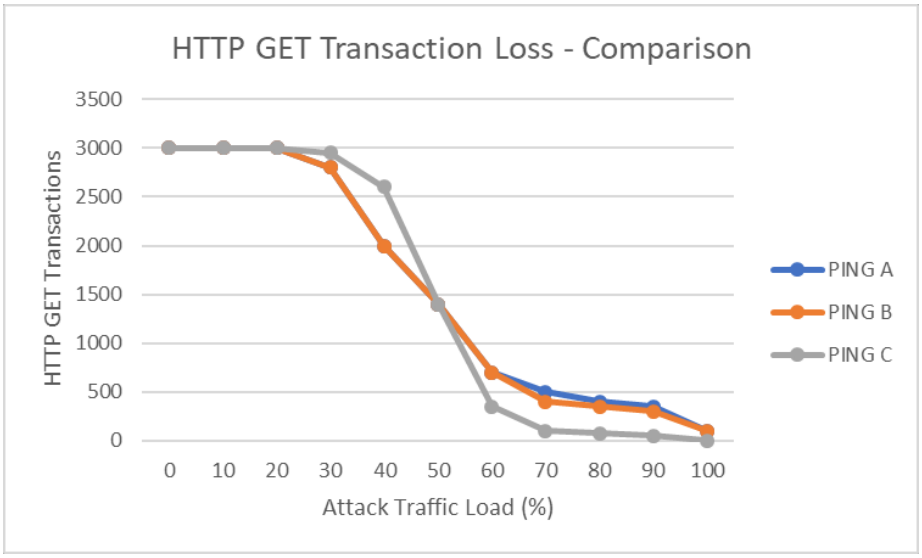

Figure III-8 Ping Flood – All Classes – HTTP GET Transaction Rate

CentOS Smurf Attack

Smurf Attack Class A – 16M Hosts

The next series of trials involves a Smurf attack against the target device. Initial traffic shows the expected 20% CPU core usage. Once the attack traffic is introduced, an initial jump in CPU core usage occurs, to about 30% usage. After, a steady rise is visible; Roughly 10% CPU core usage for each phase of the attack. Shown in Figure III-9, once the 500 Mbps mark is

reached, we start to see the CPU core usage plateau at around 65%. The CPU usage remains at the level for the remaining duration of the trial. Even though this trial utilized a very large botnet, the Operating System is handling the large amount of traffic, because typically the CPU usage would max out at around 80%. However, in this case, the CPU core usage noticeably lower than previous trial and reflects the Operating Systems ability to function under DDoS attack.

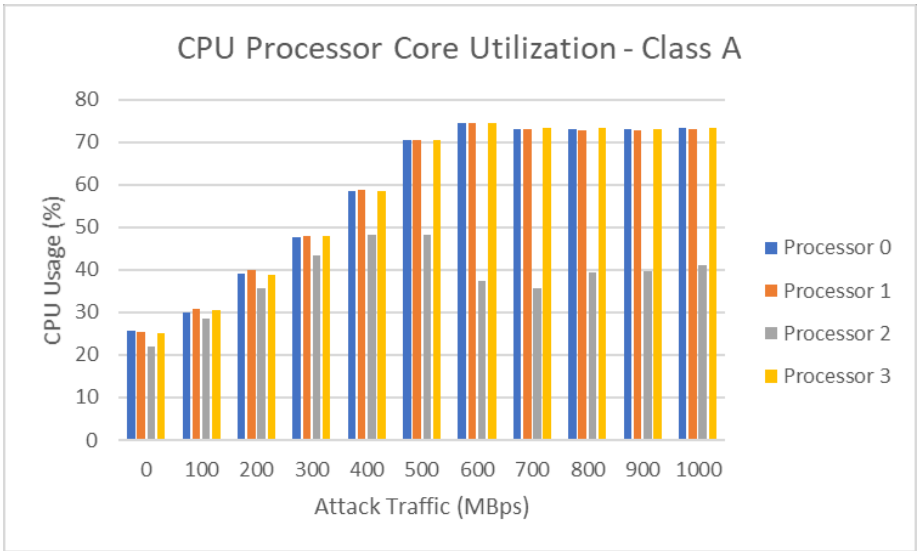

Figure III-9 Smurf Attack – Class A – CPU Processor Usage

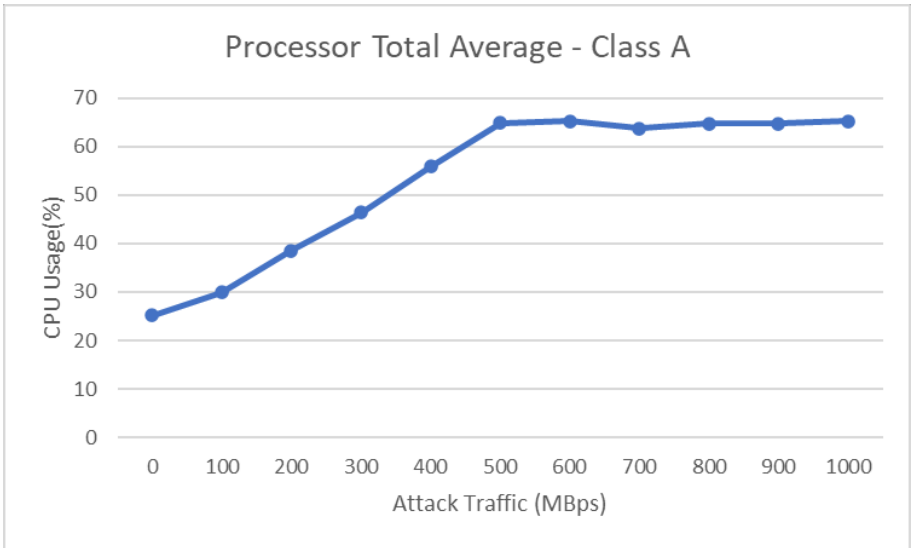

Figure III-10 Smurf Attack – Class A – Total Average CPU Usage

Smurf Attack Class B – 65k Hosts

Figure III-11 shows the next phase of the Smurf Attack DDoS trial. Unlike the previous trial, Class B shows steady levels across each CPU core. This is due to less hosts sending attack traffic to the target machine. The experiment begins with expected CPU core usage with the lack of attack traffic. Similar to the previous trial, we see a steady 10% rise at each load during the attack, shown in Figure III-12. However, this trial shows a much higher max CPU core usage, more in line with previous trials. At about 600 Mbps, the CPU core usage reaches 80% and slowly declines over the rest of the trial. The final CPU core usage at max attack load is 70%. Although this reading is marginally lower than the maximum, its still quite significant and reflects stress on the target machine.

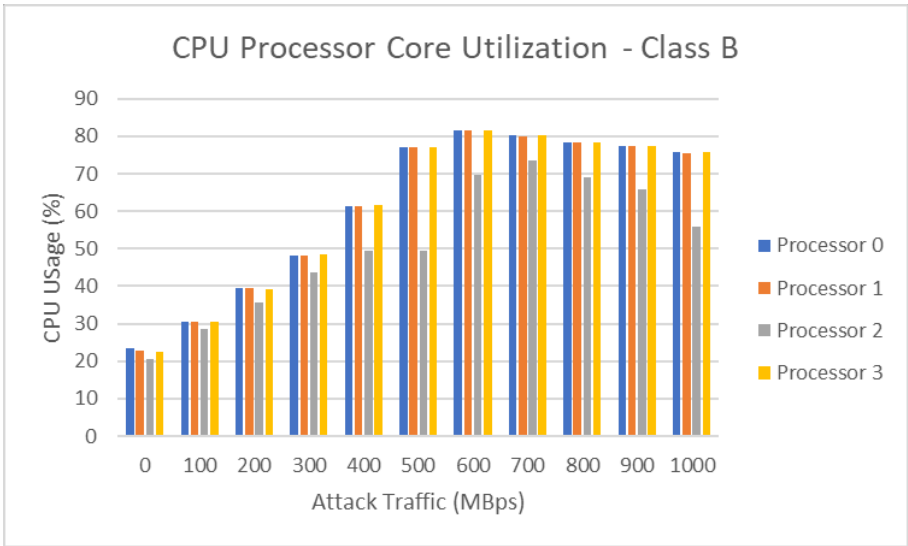

Figure III-11 Smurf Attack – Class B – CPU Processor Core Usage

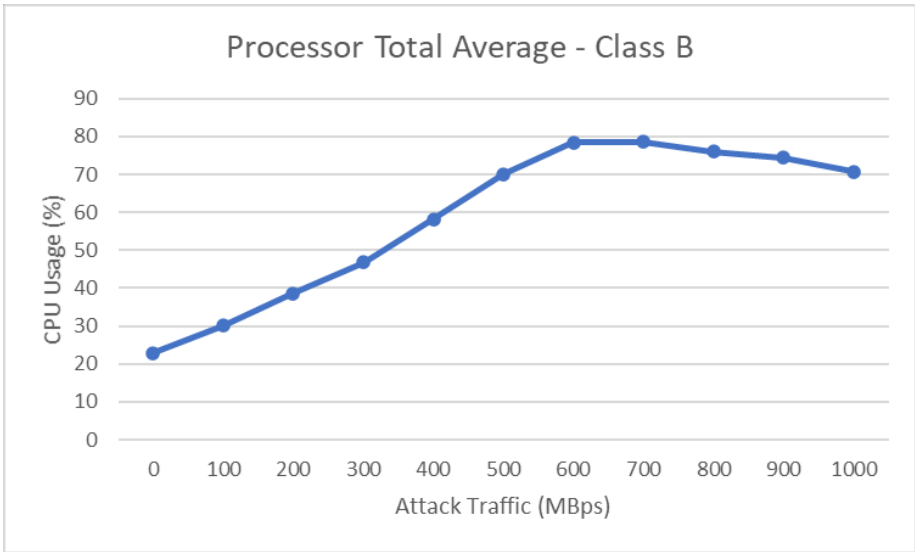

Figure III-12 Smurf Attack – Class B – Total Average CPU Usage

Smurf Attack Class C – 254 Hosts

The final phase of the Smurf Attack trial is the Class C trial. The first thing to note is that although this Class has a much smaller number of hosts than the previous two, the results are still quite significant and goes to show the type of damage a Smurf attack can cause if unmitigated. Initial nominal traffic shows the expect 20% CPU core usage. Once attack traffic is introduced, we see a steady rise over the course of the attack. According to the Figure on the next page, it still evident that a consistent 10% rise occurs at each attack load increase over the course of the DDoS attack. Figure III-14 indicates that the maximum CPU core usage for this trial was 75% at 700Mbps attack load. The max attack load reading shows that the CPU core usage decreased to a final value of about 68%. Just like the previous trial, this trial shows a steady rise, followed by a steady decline after a short peak. However, each trial performs so similarly it's likely that the Smurf attack is handled by the Operating System in way that the size of the botnet is not of significance, but more likely reliant on resource utilization.

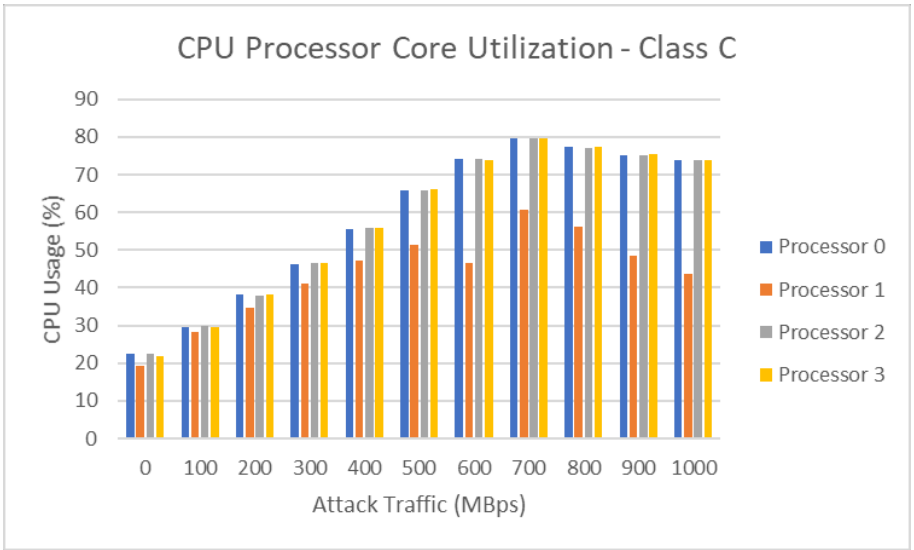

Figure III-13 Smurf Attack – Class C – CPU Processor Core Usage

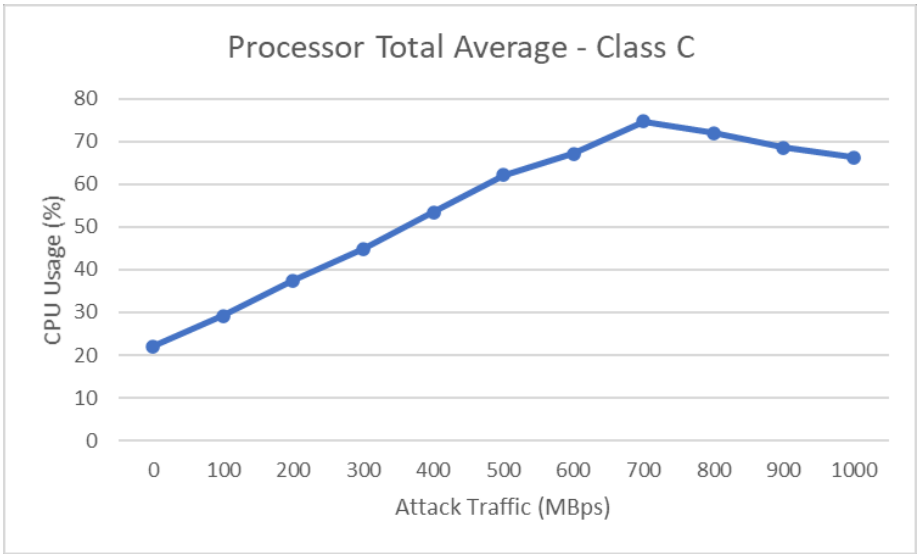

Figure III-14 Smurf Attack – Class C – Total Average CPU Usage

CentOS Smurf Processor Total Average Comparison

Smurf Attack results follow a similar trend to that of the Ping attack, however not quite as severe. The initial stage shows normal traffic with a slower increase over the length of the DDoS Attack. Here we see smaller, 5-10% increased as the attack load increases. The peak of the attack

occurs at 700Mbps whereas the peak for Ping Flood occurred sooner. There is also much variation across each class, with Class B attack being the only to reach the typical 78% CPU usage saturation threshold. At 800 Mbps, the trials start to plateau at a number lower than the saturation threshold. Class B, the highest performing of the attacks, remains at 70% average usage, even at max load. Theoretically, this attack could be considered less effective than the "less sophisticated" Ping attack, at least in regard to saturating the CPU cores.

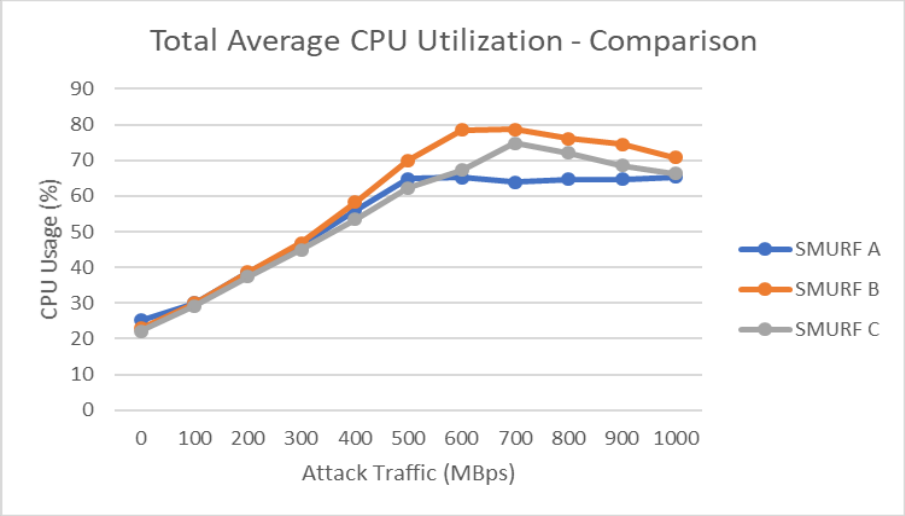

Figure III-15 Smurf Attack – All Classes – Total Average CPU Usage

CentOS Smurf HTTP GET Transaction Loss (Estimated from Output PDF)

The HTTP GET transaction losses reflect the previously state hypothesis of a Smurf attack being less effective against a CentOS hosted server. The system is able to maintain normal traffic until 500 Mbps with less than 500 transactions per second lost across all classes. At 400 Mbps, the first changes being to occur, but they are nearly negligible as they appear to be 50-100 transaction losses, compared to the 500 or so lost during the next phase of the attack load

increase. The target system is able to maintain a suitable amount of connections to remain above the saturation threshold until 90-100% attack load. At max attack load, we see total saturation and maximum connection losses.

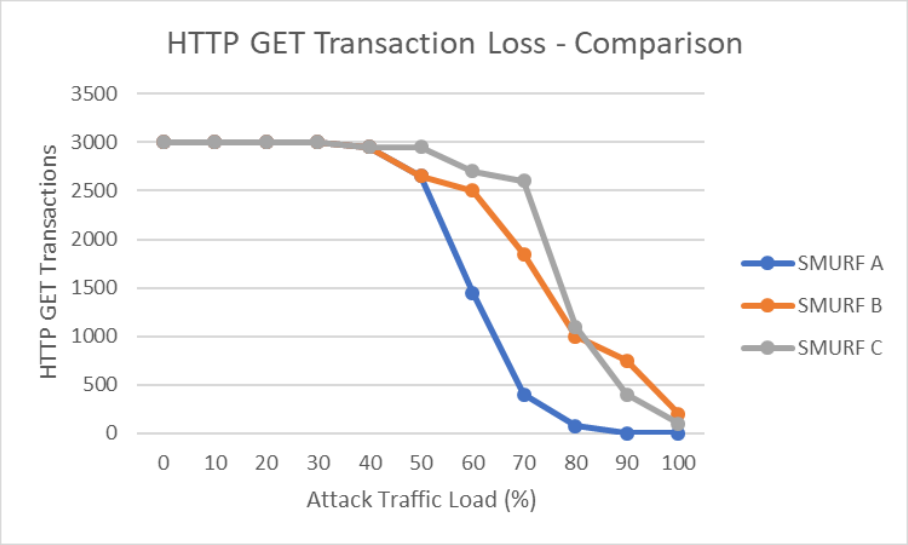

Figure III-16 Smurf Attack – All Classes – HTTP GET Transaction Rate

CentOS TCPSYN Attack

TCPSYN Attack Class A – 16M Hosts

This set of trials is when we begin the Layer 4 attacks. First, see Figure III-17 below shows how damaging a Layer 4 attack can be. The first trial shows the expect normal traffic before attack traffic is introduced. Once attack traffic begins, we see a large spike in the first load of the attack, 100 Mbps. The CPU core usage doubled. Unlike the previous trials, the steady rise is at a much large rate, almost 20% CPU core usage. At around 300 Mbps, the CPU core usage has reached 70% and continues to rise for the remainder of the trial. The final reading at max attack load shows the CPU core usage sitting at 80%.

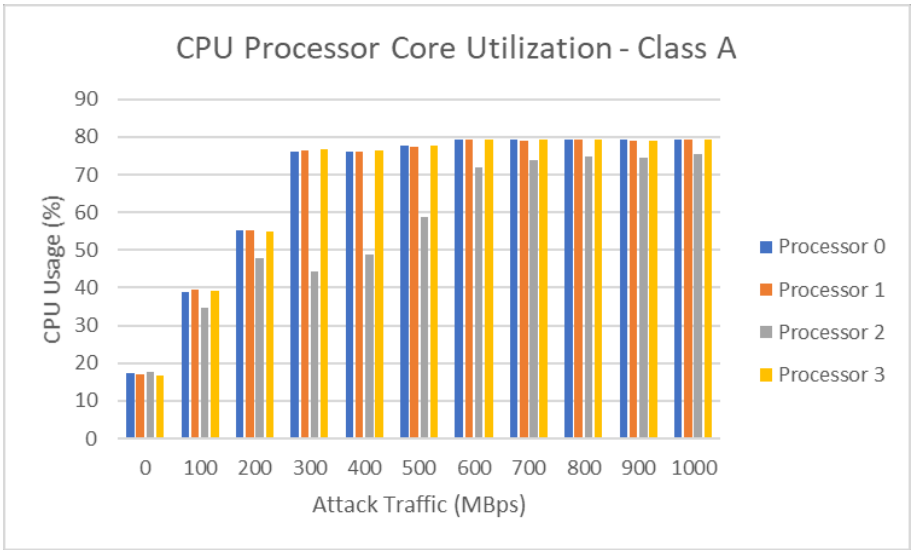

Figure III-17 TCPSYN Attack – Class A – CPU Processor Core Usage

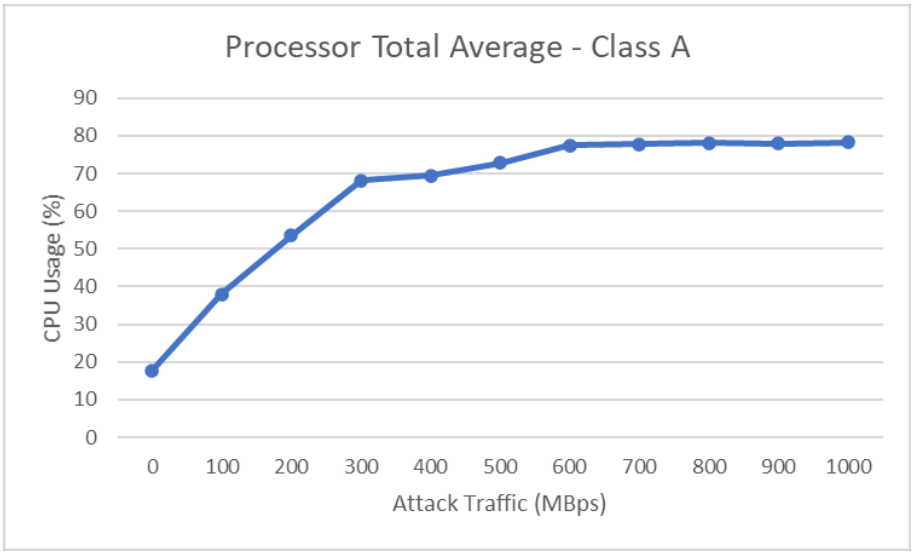

Figure III-18 TCPSYN Attack – Class A – Total Average CPU Usage

TCPSYN Attack Class B – 65k Hosts

Figure III-19 shows the next trial in the experiment. Immediately it is apparent that TCPSYN attacks are more dependent on resource utilization rather than botnet size. This concept

will be confirmed in the next phase of the trial if the pattern continues. Class B shows the initial normal traffic. Once attack traffic begins, you the 20% jumps that occur over the increasing attack load. At 100 Mbps, the CPU core usage has doubled the nominal traffic levels. Again at 300 Mbps, the CPU core usage has reached 70%. This value slowly increases over the remainder of the trial, ending up at 80% CPU core usage at 1000 Mbps attack load. These results are very similar to the previous trial and indicate that Layer 4 DDoS attacks, even against Linux OS, can still be quite formidable.

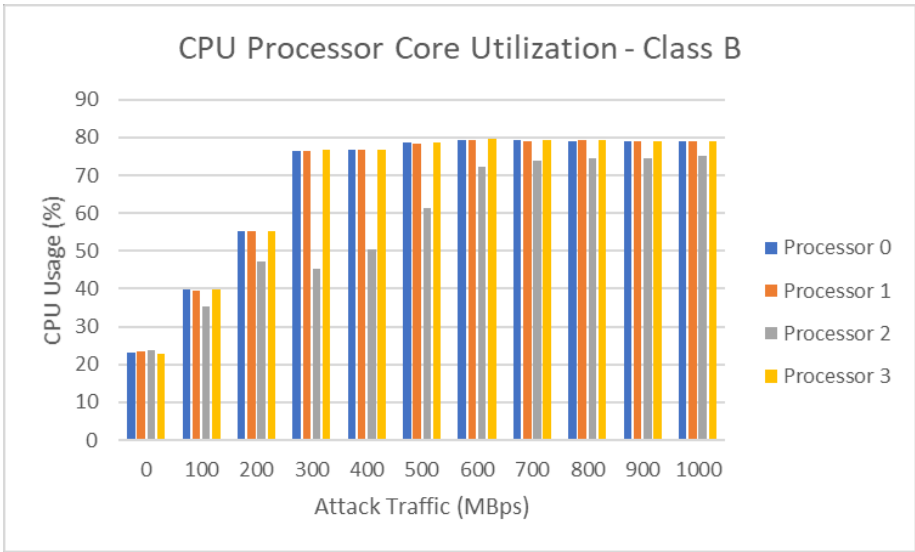

Figure III-19 TCPSYN Attack – Class B – CPU Processor Core Usage

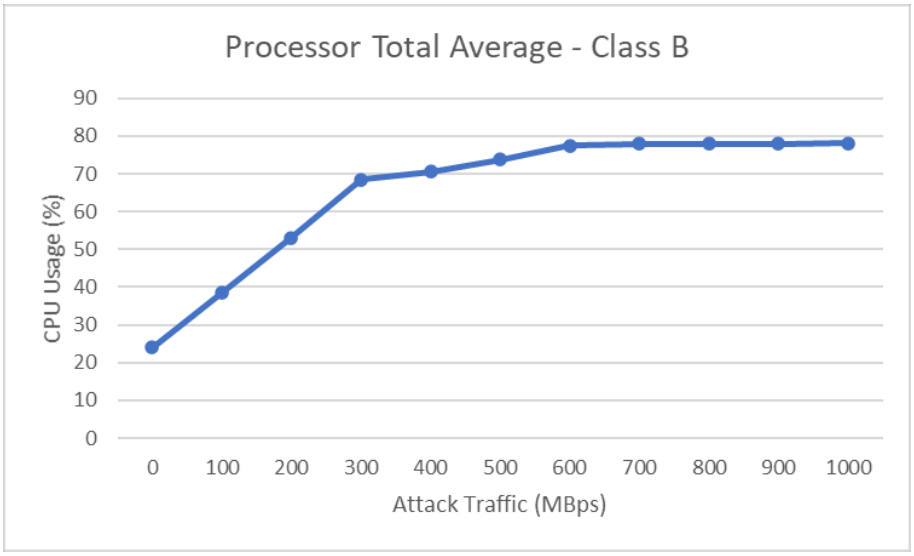

Figure III-20 TCPSYN Attack – Class B – Total Average CPU Usage

TCPSYN Attack Class C – 254 Hosts

Finally, Figure III-21 shows the Class C trial for the TCPSYN attack against CentOS. As expected, Figure III-22 shows that regardless the size of the botnet, a TCPSYN Attack can effectively render a target device useless if the attack traffic is not properly handled and allowed to be processed by the CPU. Initially, normal traffic levels are shown to be 20%, as expected. Once attack traffic is introduced, we see the familiar spikes at each increase of attack load. At 300 Mbps, the CPU core usage is a little under 70%. After, the CPU core usage steadily rises to 80% over the remaining trial. Since Class C is so much smaller than the other classes, this is the evidence needed to claim that, under Layer 4 attack, the size of the botnet is not as important as the ability for the operating system to utilize its resources when handling large amounts of Layer 4 attack traffic.

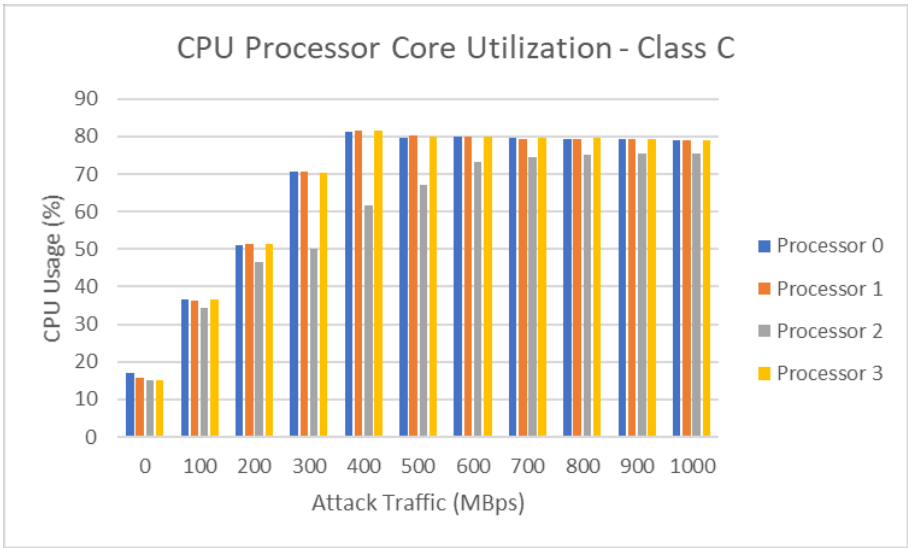

Figure III-21 TCPSYN Attack – Class C – CPU Processor Core Usage

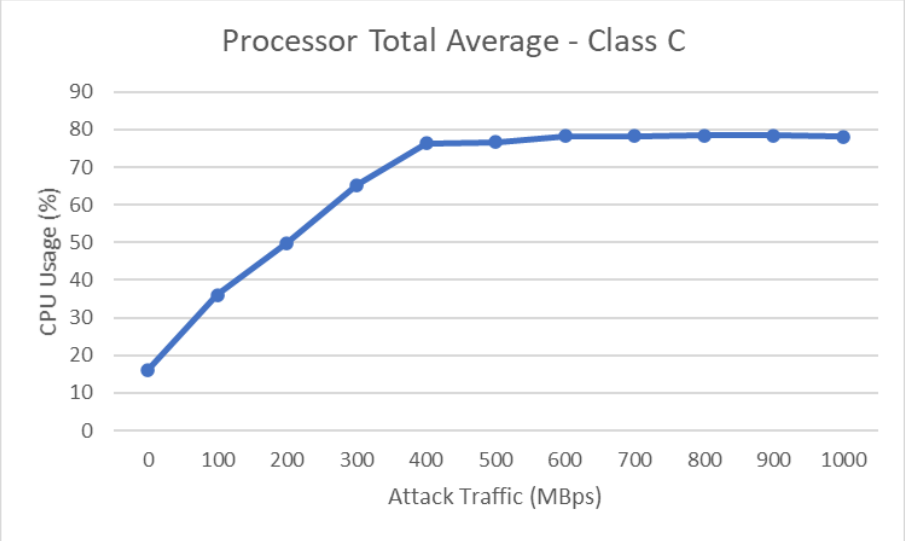

Figure III-22 TCPSYN Attack – Class C – Total Average CPU Usage

CentOS TCPSYN Processor Total Average Comparison

Initially, no attack traffic is sent into the target device. Nominal operation with no load shows as 10-20% average CPU usage in the Apache server. Class B trials peak over 20%, but

Class A features significantly more addresses, so I attribute this to background processes that may have spiked initially. The discrepancy soon disappears, as once the DDoS attack begins Class B evens out with the other Classes. This is due to the nature of using random addressing, as a fixed address leads to only one thread being created and only one core of the CPU being saturated. 80% CPU consumption seems to be the saturation point, as this is consistent across all trials, illustrated in all Figures in this section. Figure III-23 indicates that around 500-600 Mpbs attack load (50%-60%) is when CPU saturation occurs. After that point, all Classes peak at roughly 78% CPU Usage Total.

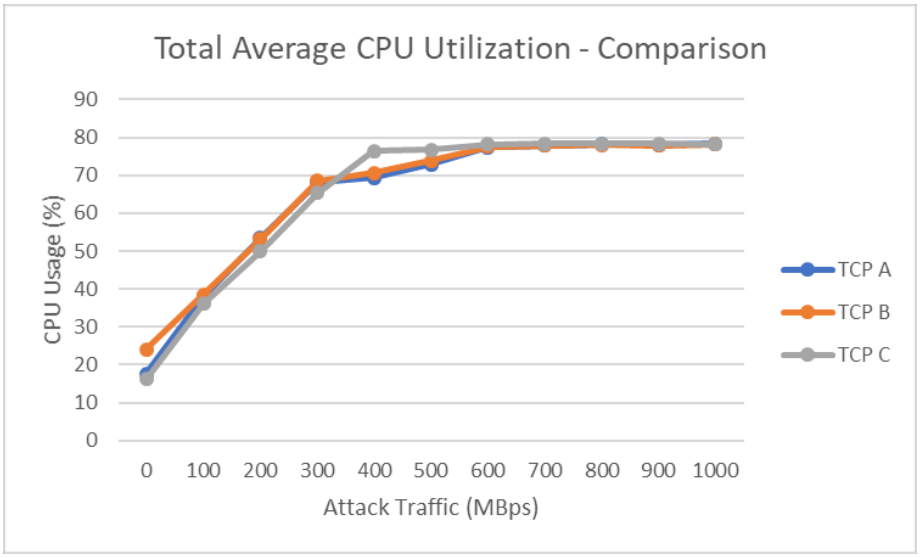

Figure III-23 TCPSYN Attack – All Classes – Total Average CPU Usage

CentOS TCPSYN HTTP GET Transaction Loss (Estimated from Output PDF)

Figure III-24 shows the HTTP connections being made under TCP-SYN Flood DDoS attack. During the initial phase, no attack traffic, you can clearly see the 3000 HTTP GET per second baseline nominal traffic. This persists until about 300 Mbps (30%) attack load, when we get a sharp decline in legitimate traffic; nearly 1000 HTTP GET per second. However, this is only

with Class B addressing at this point. Once we move into the next phase, 40% attack load, the other Classes begin to decline as well. Although, Class B seems to lose connections quicker and sooner, this may be due to background processes. Class A should, in theory, decline the quickest, as it is the largest number of hosts, however it is second to Class B. Each class declines at different attack loads, however, each attack declines at a steady rate. At about 70% attack load, each class has dropped below 400 HTTP transactions per second. At this point, the attack is successful and access to the server is throttled. After this point, all legitimate traffic is lost in the remaining attack load trials.

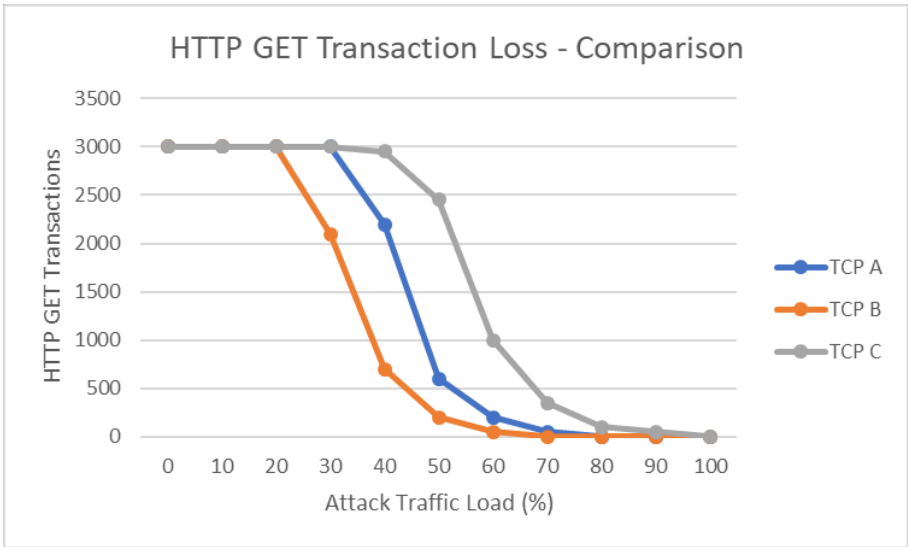

Figure III-24 TCPSYN Attack – All Classes – HTTP GET Transaction Rate

CentOS UDP Flood Attack

UDP Flood Class A – 16M Hosts

The final phase of these trials is the remaining Layer 4 attack, the UDP Flood. At first glance, you see similar results to the Class A trial for TCPSYN. Figure III-25 shows the initial

expected level of CPU core usage in regards to normal traffic. Once attack traffic is introduced, a steady increase is visible, roughly 10% for each level of attack load increase. At 300 Mpbs, the CPU core usage reached 70% and shortly after rises to 80% where it remains for the rest of the experiment. Already we see similar results to the previous trial. So, regardless of the type of attack, the operating system handles the ingress traffic very similarly, with little effect coming from the size of the botnet. This is further proven as this trial continues.

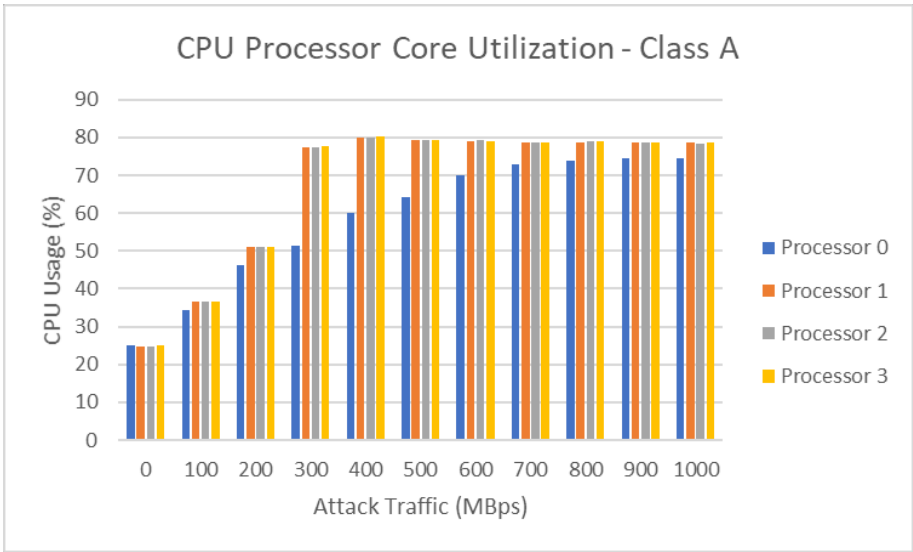

Figure III-25 UDP Flood – Class A – CPU Processor Core Usage

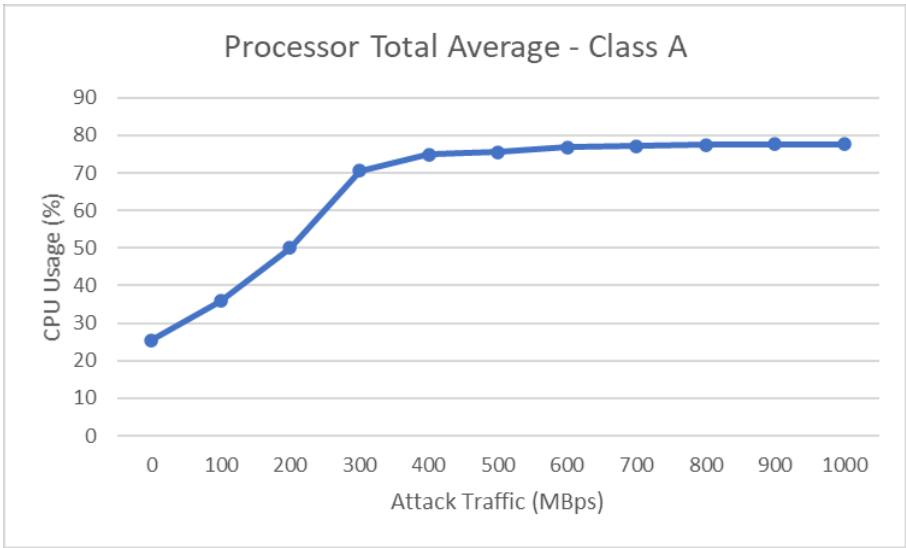

Figure III-26 UDP Flood – Class A – Total Average CPU Usage

UDP Flood Class B – 65k Hosts

Class B trial shows similar results to the Class A trial. The initial 20% is marginally higher in this trial, closer to 25% CPU core usage with no attack traffic. The steady rise occurs again once attack traffic is introduced. At 300 Mbps, the target machine sits at 70% CPU core usage. This is expected at this point. Afterward at 400 Mbps, the CPU cure usage rises to 80%, where it remains. The next phase, Class C, is expected to show similar results to Class A and Class B. At this point, CentOS has handled Layer 4 attacks very similarly, with little regard to the protocol being utilized.

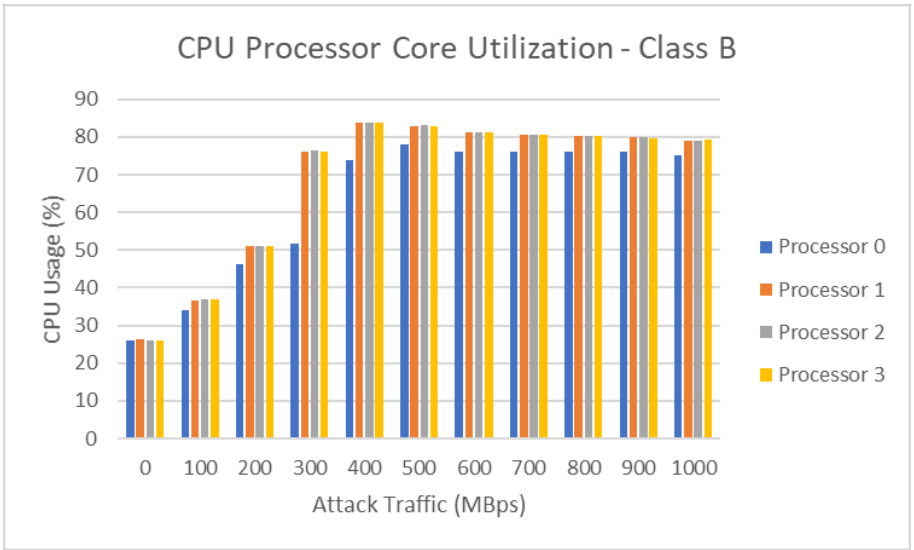

Figure III-27 UDP Flood – Class B – CPU Processor Core Usage

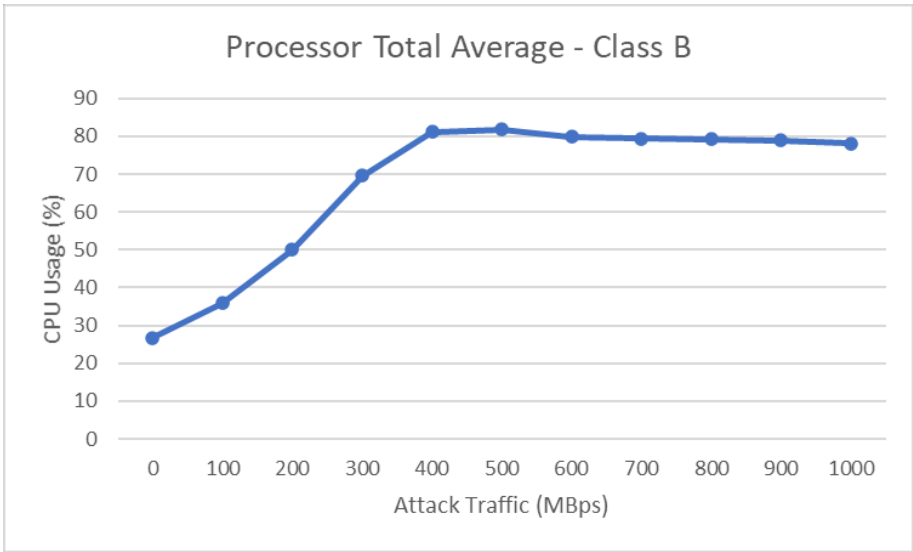

Figure III-28 UDP Flood – Class B – Total Average CPU Usage

UDP Flood Class C – 254 Hosts

The final trial in this Chapter is the Class C UDP Flood trial. As previously mentioned, Class C is much smaller than either of the previous Classes. However, despite this significant

difference in botnet size, Figures III-29 and III-30 indicate that the size of the botnet is not the only significant part of a DDoS attack. It is also up to how the Operating System handles said attack. The normal attack traffic is consistent with the rest of the experiment. The attack traffic shows a 10% increase in CPU core usage at each level of attack traffic load. At 400 Mbps, the CPU core usage has reached 70%, a little later than previous trials. This is due to the smaller scale of the attack. The core usage continues to rise by small amounts as the trial continues. Once 700 Mbps threshold is reached, the CPU core usage reaches just under 80%.

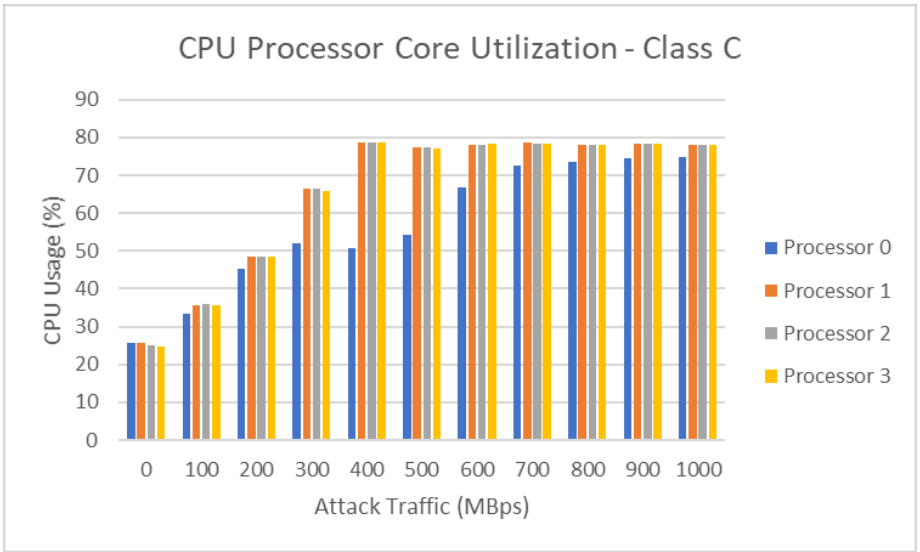

Figure III-29 UDP Flood – Class C – CPU Processor Core Usage

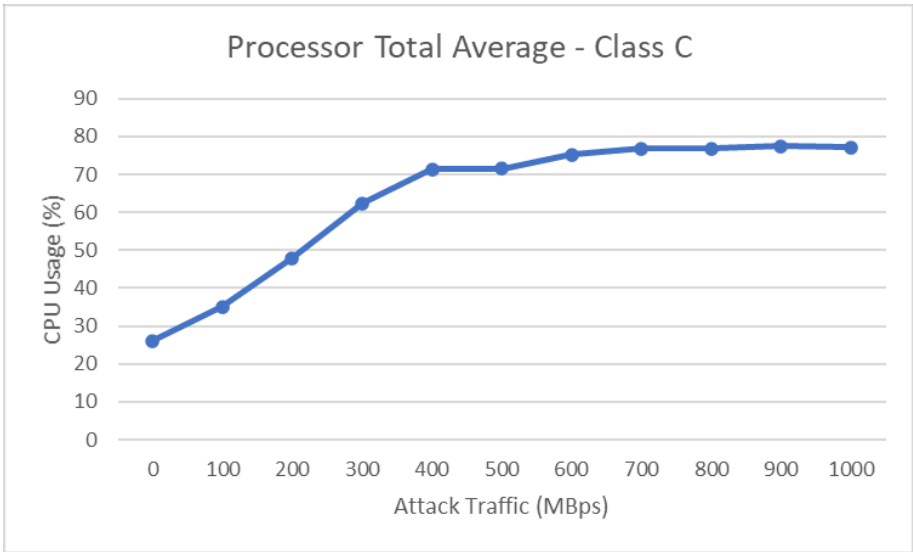

Figure III-30 UDP Flood – Class C – Total Average CPU Usage

CentOS UDP Processor Total Average Comparison

In this experiment, the destination port on the target server is left open. The trials with the port closed lead to normal traffic since no ICMP Destination Unreachable were generated.

Figure III-31 shows the results of the average CPU usage across all cores under UDP Flood DDoS attack. Again, no attack traffic load shows nominal function at around roughly 25% CPU usage. These results show a steady climb into the saturation point of the target system. However, this attack shows saturation of the target as early as 400 Mbps (40%) attack load, 10% earlier than TCPSYN. During this attack, it can be noted that some trials reached 80% CPU saturation, higher than the TCPSYN attack, although marginally. At max attack load, however, the saturation point remains the same figure as the TCPSYN trials, around 78%.

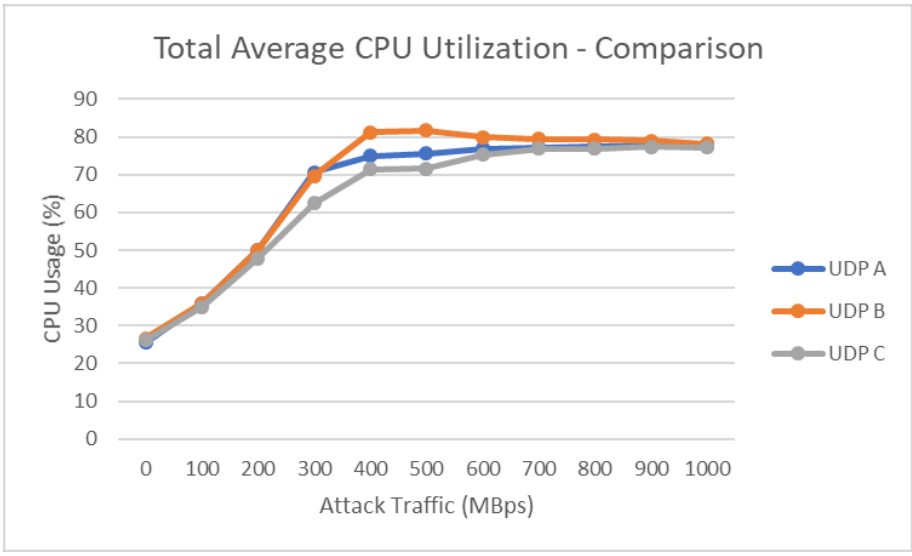

Figure III-31 UDP Flood – All Classes – Total Average CPU Usage

CentOS UDP HTTP GET Transaction Loss (Estimated from Output PDF)

The corresponding HTTP GET trials can be seen in Figure III-32. The loss of transactions begins at 300 Mbps, with Class B showing a marginally quicker decline, which falls behind once the 400Mbps attack load mark is reached. Once 600Mbps is reached, Class A and C have dropped below the saturation point, with Class B just slightly above 500 HTTP GET transactions per second. Past this, total loss of legitimate traffic is achieved with Class B allowing only a fraction of the nominal traffic to continue to the target system. The marginal discrepancy in the decline of each trial can once again be attributed to various background tasks as no configurations, other than changing the addressing class, is done between trials.

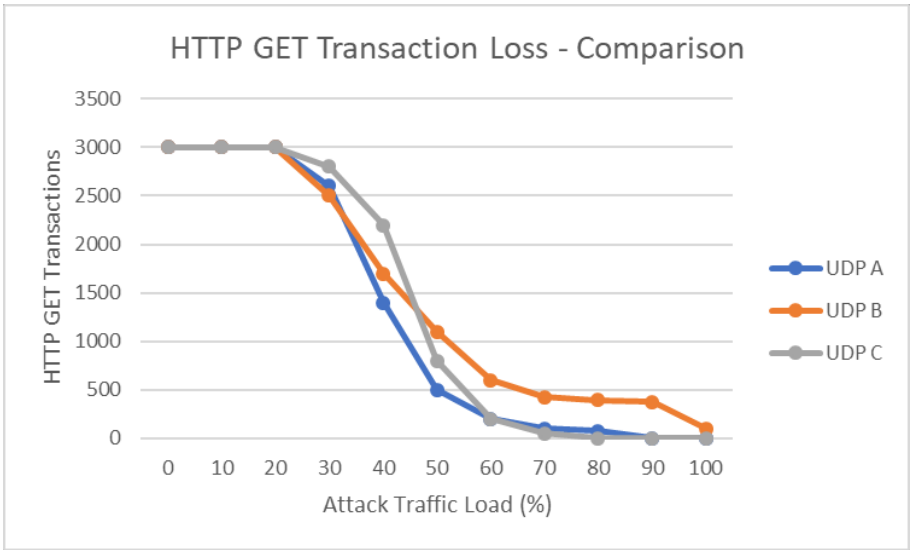

Figure III-32 UDP Flood – All Classes – HTTP GET Transaction Rate

CHAPTER IV

CENTOS V WINDOWS COMPARISON

In this final chapter, the evidence gathered in all the trials will be plotted on the same axis so that a clear comparison could be made between the performance of Windows Server and CentOS. This chapter is divided into sections just like the previous entries. Layer 3 attacks will be covered first with Layer 4 attacks coming in at the end. This chapter will compare the Total Average CPU Utilization and HTTP GET Transaction rate loss between the two Operating Systems. These two fields compared will show not only the differences in performance of the two operating systems, but also show that server resources and server availability are impacted differently under DDoS attack.

Ping Class A – 16M Hosts

Figure IV-1 shows the shows that the CPU usage for each operating system begins to rise quickly once attack traffic is introduced. Windows Server rises at a higher rate, which would indicate that more resources are being used to process the traffic, however after 200 Mbps, the CPU usage falls to a level not much higher than the baseline reading, averaging about 35% for the rest of the trial. Meanwhile, CentOS continues to rise all the way to the 80% CPU usage mark and remains there. This is a clear indication that CentOS is affected more adversely under Ping attack than Windows Server, at base installation. However, this is only server resources.

Figure IV-2 indicates that regardless the effect on machine resources that the attack has, the amount of available HTTP connections remains the same for both operating systems. For the duration of the trial, they fall at the same rate.

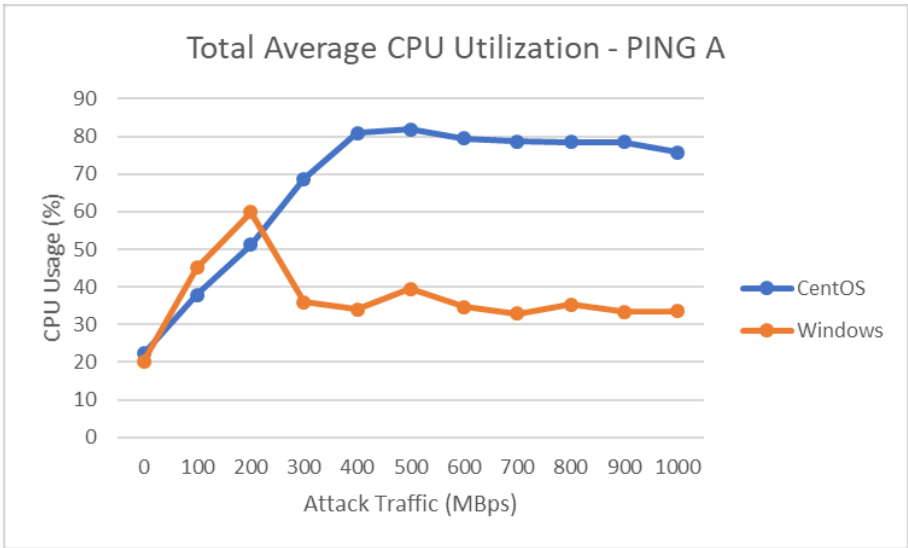

Figure IV-1 CentOS v Windows – Ping Flood – Class A – Total Average CPU Usage

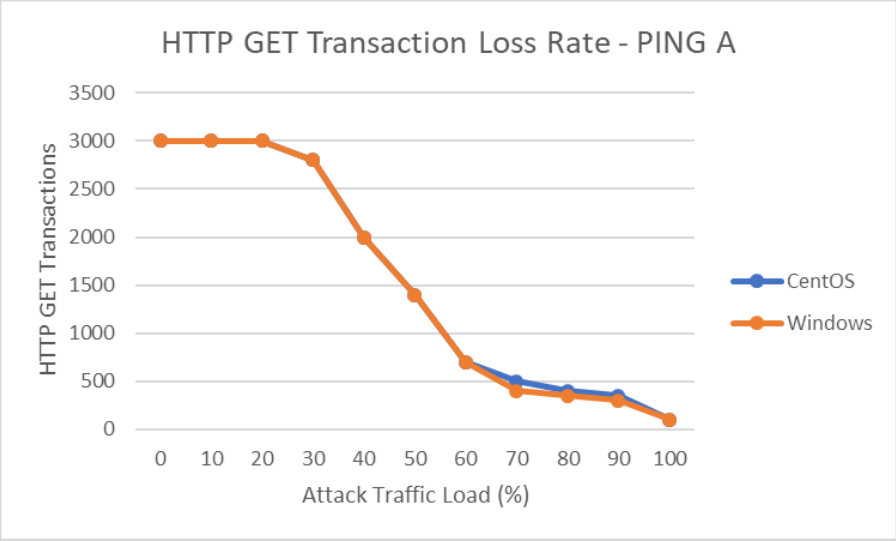

Figure IV-2 CentOS v Windows – Ping Flood – Class A – HTTP Transaction Rate

Ping Class B – 65k Hosts

Figure IV-3 shows very similar results to those of the Class A trials. The previously established theory that CentOS is more adversely affected by a Ping attack is evident in this trial as well. The CPU usage quickly and steadily rises to 80% at an attack load of 400 Mbps and remains there. The Windows performance is similar to Class A; there is a quick rise CPU usage up to 200 Mbps attack load, where the usage falls again to much less resource intensive levels. Of note here is that, again, Windows rose at a larger rate initially, before falling.

The HTTP transaction rate shows in this trial, CentOS can maintain more HTTP transactions than its Windows counterpart. Initially, Windows can maintain more connections for a longer period, however the most drastic change occurs when the attack load reaches 40% and the available connections drop by over 2000 transactions. For the remainder of the trial, CentOS is able to maintain more connections. Other than the large drop on Windows Server, both show steady decline in transactions, which is expected under this type of attack.

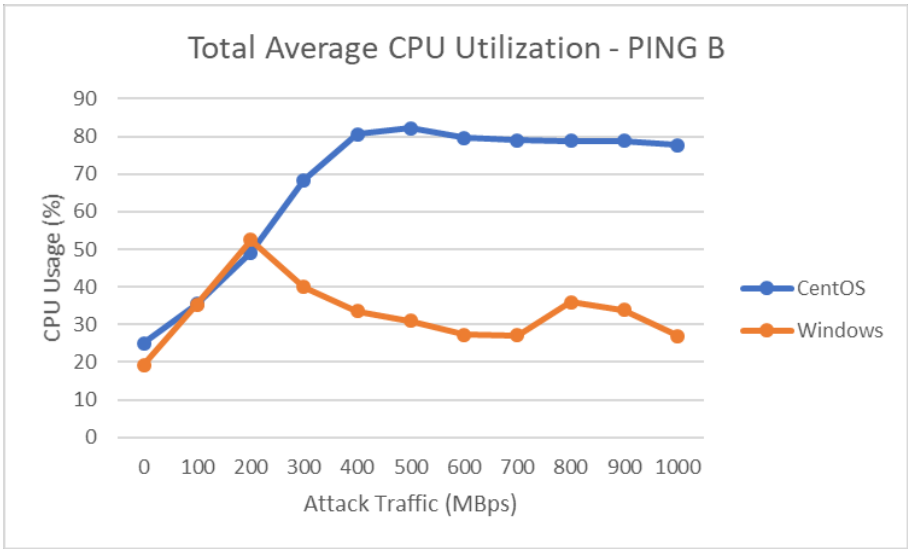

Figure IV-3 CentOS v Windows – Ping Flood – Class B – Total Average CPU Usage

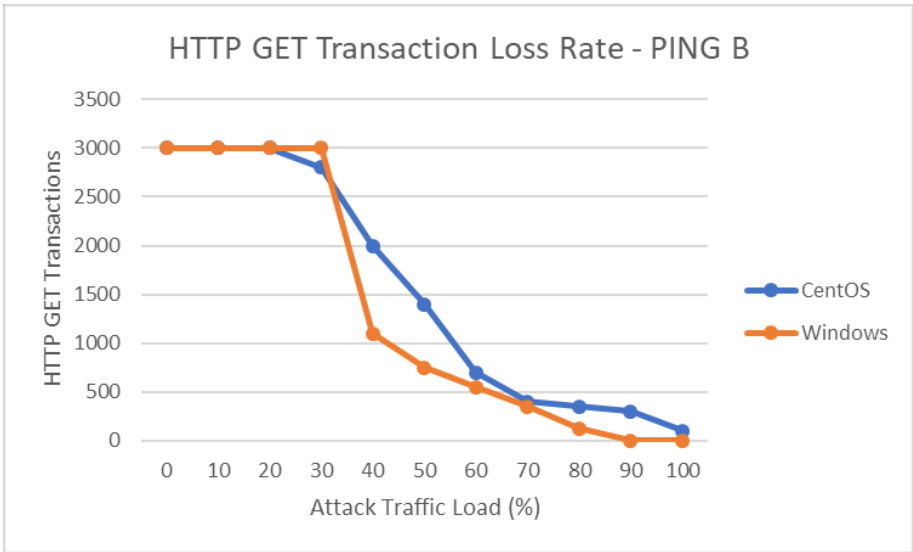

Figure IV-4 CentOS v Windows – Ping Flood – Class B – HTTP Transaction Rate

Ping Flood Class C – 254 Hosts

In the final trial of Ping attack, Figure IV-5 shows results consistent with the previous two trials. CentOS is less able to function under Ping attack than Windows; CentOS rapidly rises to 70% CPU usage by 400 Mbps attack load and slowly rises to 80%, while Windows repeats the previous trials by quickly rising like CentOS and then falling to manageable levels

The transaction rate, shown in Figure IV-6, shows that in the case of Class C, Windows is able to maintain more connections over a longer period. CentOS slowly loses all its connections over the course of the trial. When it comes to ICMP Ping attacks, both operating systems are affected in very different ways. CentOS is unable to manage resources for long, more intensive attacks, however is able to maintain connections as priority. However, as the botnet sizes shrink, Windows is more capable of maintains connections. This is an indicator that Windows may be more beneficial to a smaller scale network. Whereas CentOS might be more useful in a larger scale environment.

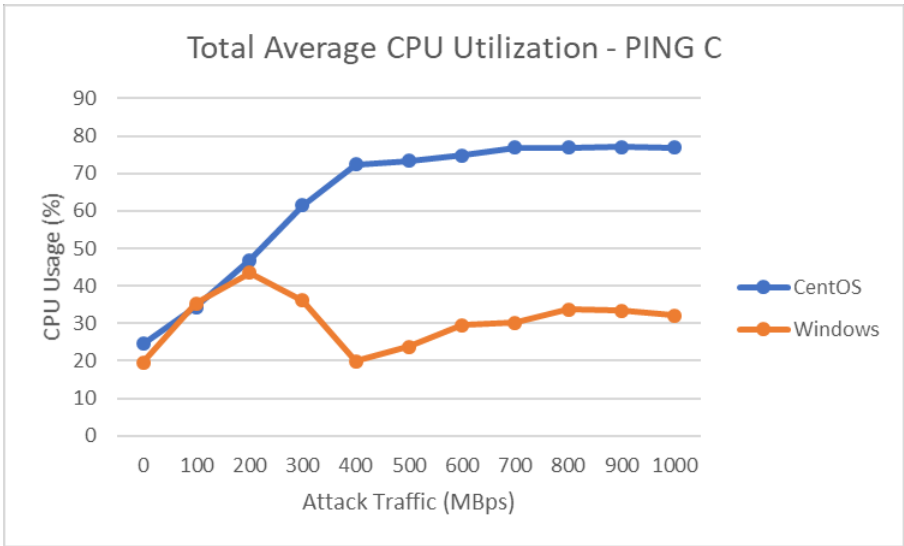

Figure IV-5 CentOS v Windows – Ping Flood – Class C – Total Average CPU Usage

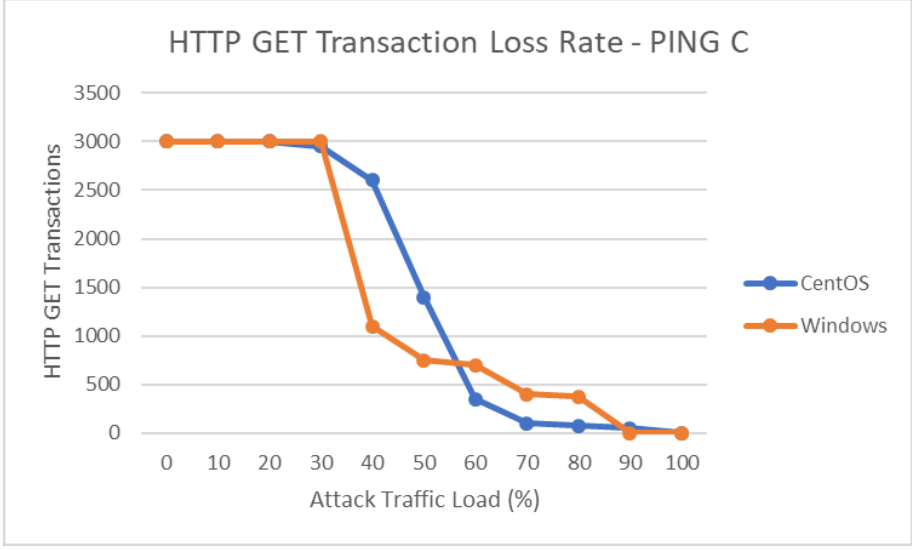

Figure IV-6 CentOS v Windows – Ping Flood – Class C – HTTP Transaction Rate

Smurf Attack Class A – 16M Hosts

The next round of ICMP attack, the Smurf attack, is shown in Figure IV-7. Immediately, Window rises to high CPU usage levels, peaking at 90% CPU usage. With such a large botnet sending attack traffic, Windows is unable to maintain resources and keep connections open. By 300 Mpbs, the device is fully locked up and has the largest decrease in available connections at the corresponding point in Figure 72. By 400 Mbps, or 40% attack load, there are no available connections.

CentOS trials show a steady rise to a peak of about 65% CPU usage. This shows that CentOS is able to manage its resources better under a Smurf attack. CentOS is also able to maintain more connections for a longer period under attack; however, CentOS still loses all connections at the 90-100% attack load mark.

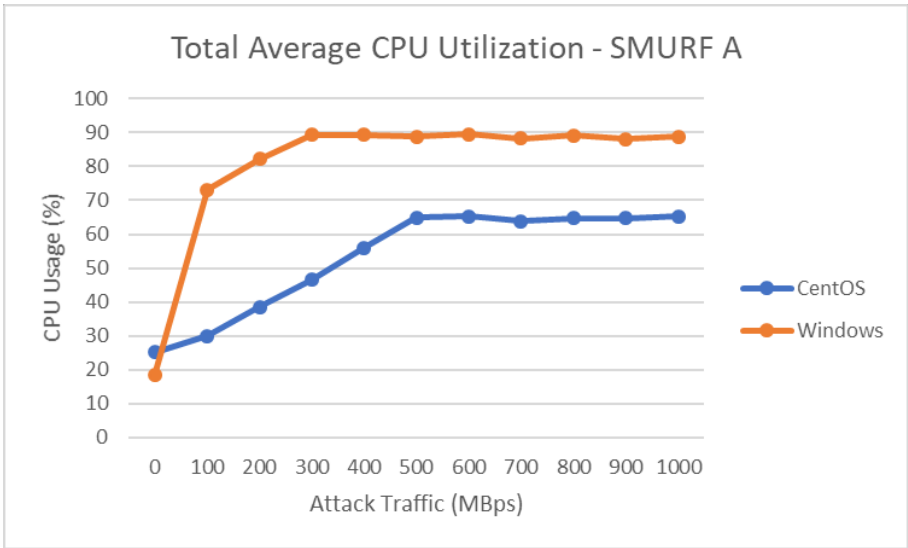

Figure IV-7 CentOS v Windows – Smurf Attack – Class A – Total Average CPU Usage

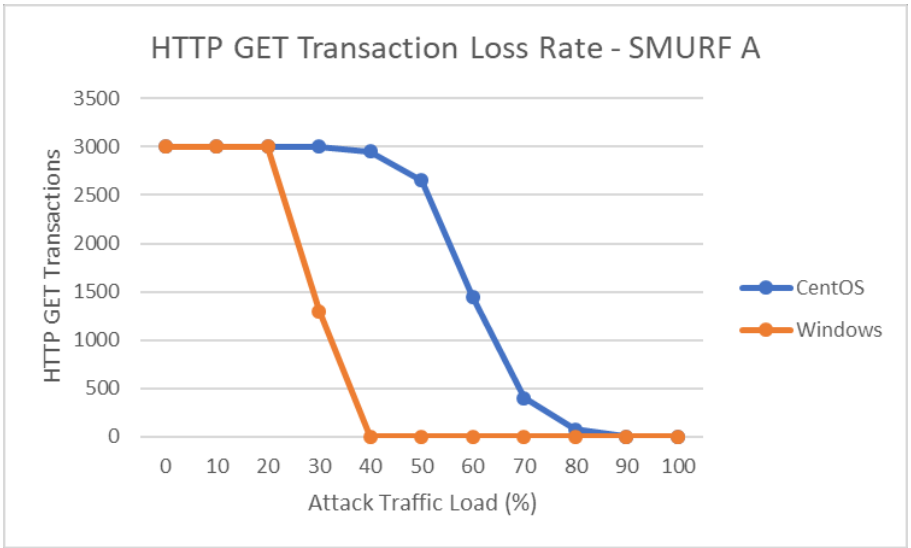

Figure IV-8 CentOS v Windows – Smurf Attack – Class A – HTTP Transaction Rate

Smurf Attack Class B – 65k Hosts

In the next set of results, it is even more clear that Windows has trouble with a Smurf attack as the botnet size is drastically smaller, but the results are very similar to the Class A results. There is a rapid rise once attack traffic is introduced; 70% CPU usage at only 100 Mbps attack load. Soon after at 300 Mbps, the victim machine has reached 90% CPU usage where it remains. The transactions rate also has a large drop at 300 Mbps, or 30% attack load, and no connections are available by the next phase of the attack, 40% attack load.

CentOS performs similar to the Class A trial; there is a steady rise, about 10% usage per attack load. At 600 Mbps, the server has reached 80% CPU usage. Over the rest of the trial, the usage drops to 70% at max attack load, 1000 Mbps. The transaction rate also outperforms the Windows counterpart in this Class, according to Figure 74. At 900 Mbps, or 90% attack load, no more connections are available. This shows the significance of a large scale Smurf attack on any system.

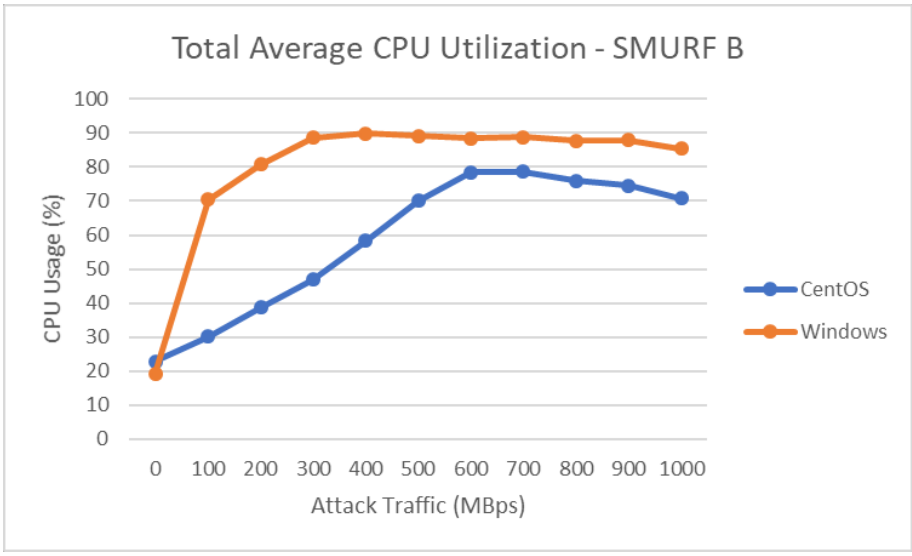

Figure IV-9 CentOS v Windows – Smurf Attack – Class B – Total Average CPU Usage

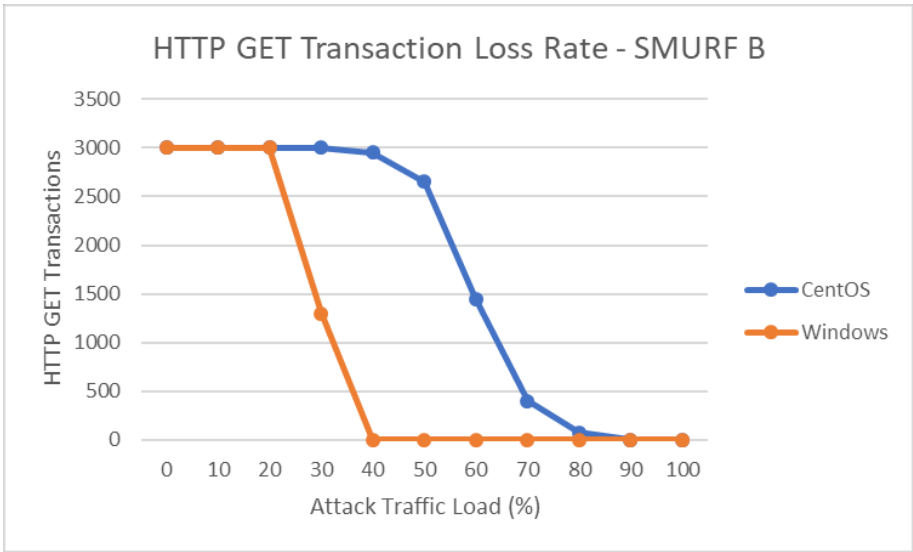

Figure IV-10 CentOS v Windows – Smurf Attack – Class B – HTTP Transaction Rate

Smurf Attack Class C – 254 Hosts

In the final section of the Smurf attacks, the Class C attack, it is clear that Windows Server has difficulty performing under this type of Layer 3 attack. Under Class C Smurf attack,

Windows Server reaches 90% CPU usage by the 300 Mbps, or 30% attack load, mark. Although at max attack load, the CPU usage falls by a little over 10%, but still remains quite high. HTTP connections are also rapidly lost under Class C attack. By 30% attack load, or 300 Mbps, over half the connections have been lost. At 40%, all connections have bene lost.

CentOS shows a steady rise in Figure IV-11, to a peak of 75% CPU usage at 700 Mbps. After, CPU usage drops to about 65% usage. Figure IV-12 shows, under Class C attack, CentOS is able to keep many more connections open over a long period. It is not under the 70% attack load mark that shows the most significant drop in connections. At the 90% attack load mark, less than 500 connections are available. Although we had varying results in the previous Layer 3 attack, these trials show a much clearer comparison between the two operating system at all Classes.

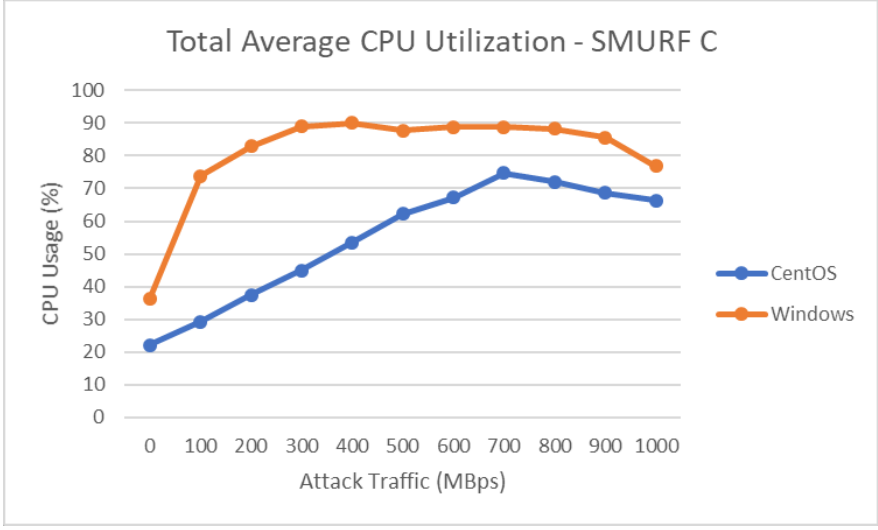

Figure IV-11 CentOS v Windows – Smurf Attack – Class C – Total Average CPU Usage

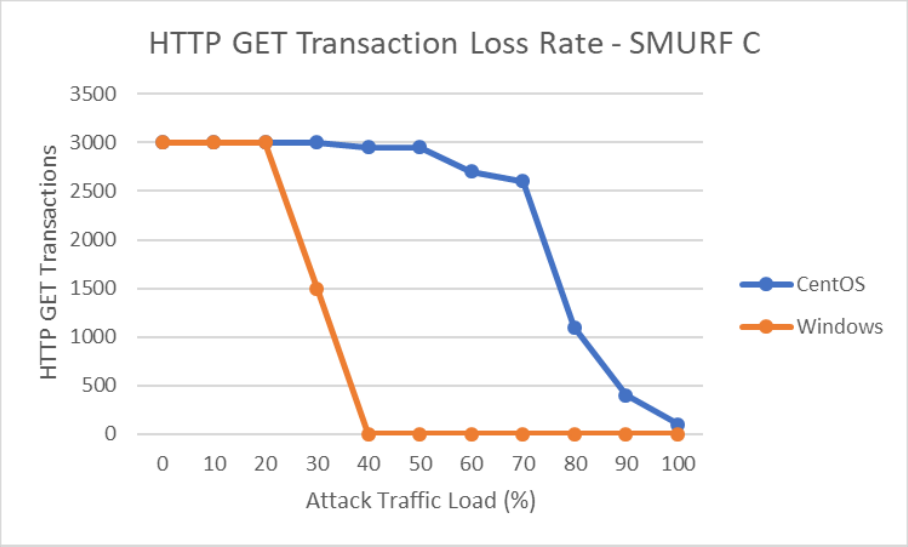

Figure IV-12 CentOS v Windows – Smurf Attack – Class C – HTTP Transaction Rate

TCPSYN Attack Class A – 16M Hosts

This section will cover the first of the Layer 4 attacks, the TCPSYN Flood attack. Class A results show that once attack traffic is introduced, Windows CPU usage quickly rises to a little above 80% CPU usage and remains in that threshold. Figure IV-14 shows that once attack traffic is introduced, all connections are immediately lost, as early as 10% attack load. Class A TCPSYN attack causes Windows to saturate the CPU and lose all connections.

Figure IV-13 shows CentOS is still able to function under this form off attack. The results show a steady rise over the course of the trial and plateaus at about 80% CPU usage at the 700 Mbps mark. At higher loads, this form of attack shows strong results against the operating systems. Figure IV-14 shows that at the same mark, 700 Mbps or 70% attack load, all HTTP connections are lost. The beginning of the attack retains several connections but as the attack ramps up, the connections are quickly lost: The biggest difference at the 40-50% attack load mark.

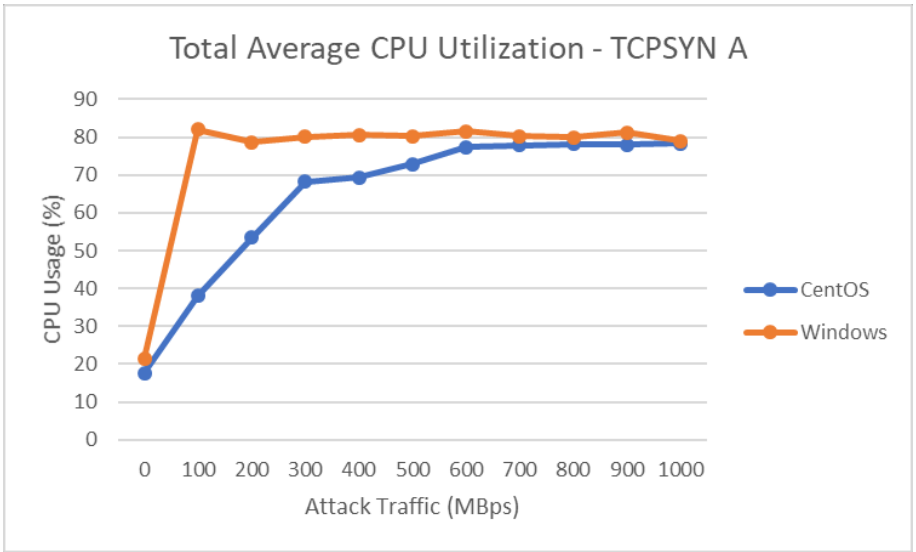

Figure IV-13 CentOS v Windows – TCPSYN Attack – Class A – Total Average CPU Usage

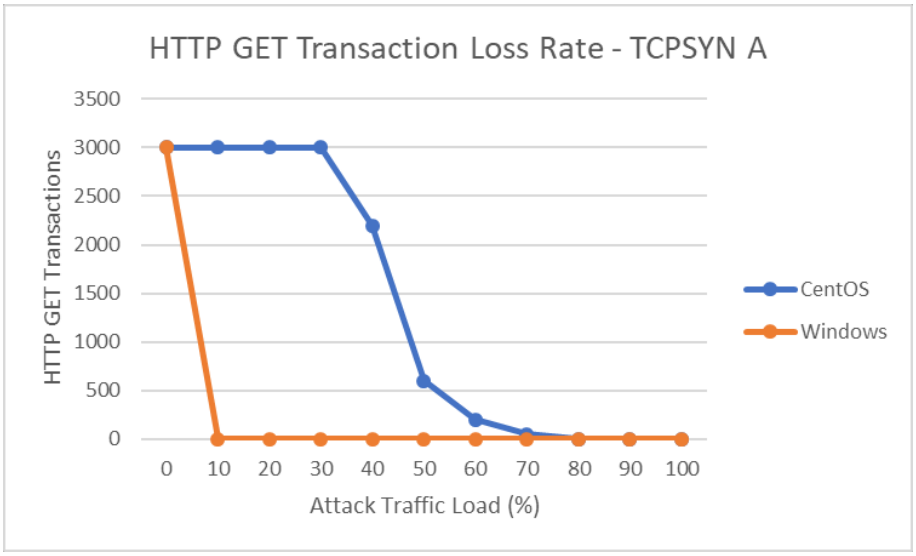

Figure IV-14 CentOS v Windows – TCPSYN Attack – Class A – HTTP Transaction Rate

TCPSYN Attack Class B – 65k Hosts

Once the botnet size shrinks, Windows is able to perform under this type of attack, but not for long. Figure IV-15 shows here is a rise in CPU usage once attack traffic is introduced and

a steady climb to 80% CPU usage by the 300 Mbps mark. The system CPU usage remains at 80% for the rest of the trial. More connections are also available under this type of attack. The target system is able to maintain all 3000 connections until the 20% attack load mark, where connections established drops by almost half. By 40% attack load, less than 500 connections are available and the number continues to drop. By 50%, all connections are lost.

Even under Class B TCPSYN attack, CentOS continues to show superior performance to Windows Server. Figure IV-15 shows a steady rise in CPU usage, about 15% per phase of the attack. At 300 Mbps, the CPU usage has reached 70% and plateaus just under 80% at 600 Mbps. Both operating systems are able to maintain all connections for the initial phases of the attack and remain relatively similar in performance for the duration, shown in Figure IV-16. Although CentOS loses connections at a marginally smaller rate, by the later stages of the attack, neither operating system is able to maintain any connections. By 50% attack load, both operating systems maintain less than 500 connections.

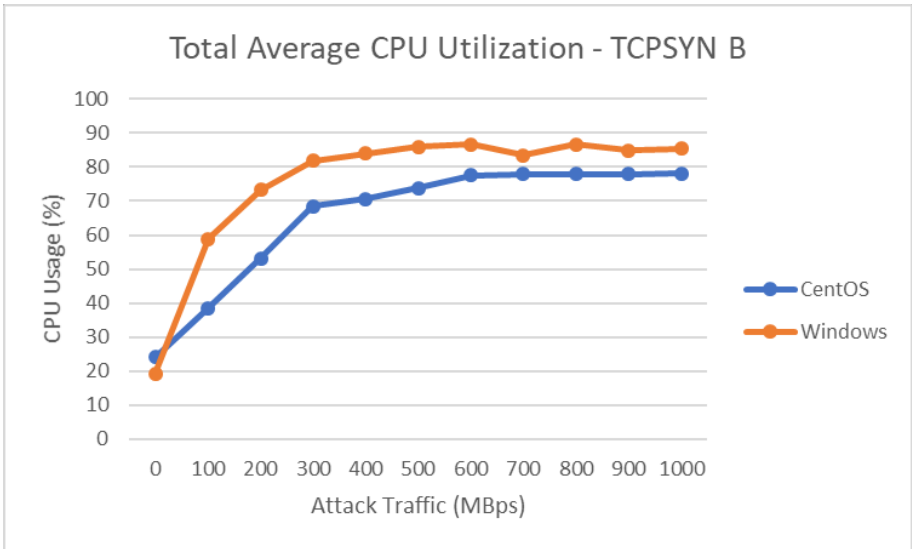

Figure IV-15 CentOS v Windows – TCPSYN Attack – Class B – Total Average CPU Usage

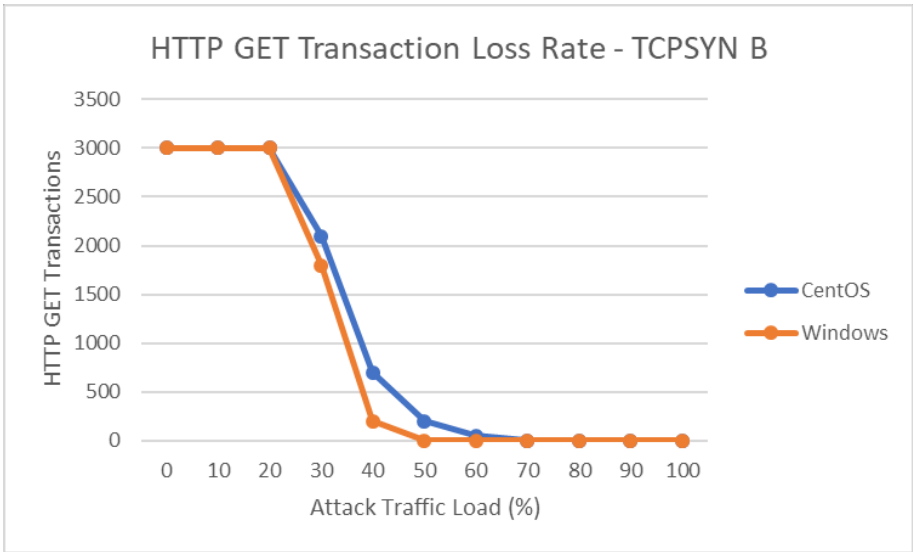

Figure IV-16 CentOS v Windows – TCPSYN Attack – Class B – HTTP Transaction Rate

TCPSYN Attack Class C – 254 Hosts

The Class C TCPSYN attack results show very different results than the previous trials. It seems that Layer 4 attacks are more dependent on the botnet size than Layer 3 attacks. Figure IV-17 shows Windows Server CPU usage rising at a higher rate then CentOS, before dropping down to a manageable 40% CPU usage for the remainder of the attack. The HTTP transaction rate in Figure IV-18 indicates that under Class C attack, Windows is still able to maintain many of its HTTP connections. These connections are retained under max attack load where we have a large drop, below 500 transactions per second.

CentOS appears to process the traffic the same way, regardless of the size of the botnet. There is a steady rise, about 10% per phase, up to 75% at 400 Mbps. After, the CPU usage rises to 80% at 600 Mbps where it remains. Like Windows Server, CentOS was able to maintain all connections until about 400 Mbps, or 40% attack load. The connections per second begin to

decline, with a large drop at 50% attack load. By 70% attack load, less than 500 connections are available, and the amount continues to fall after that.

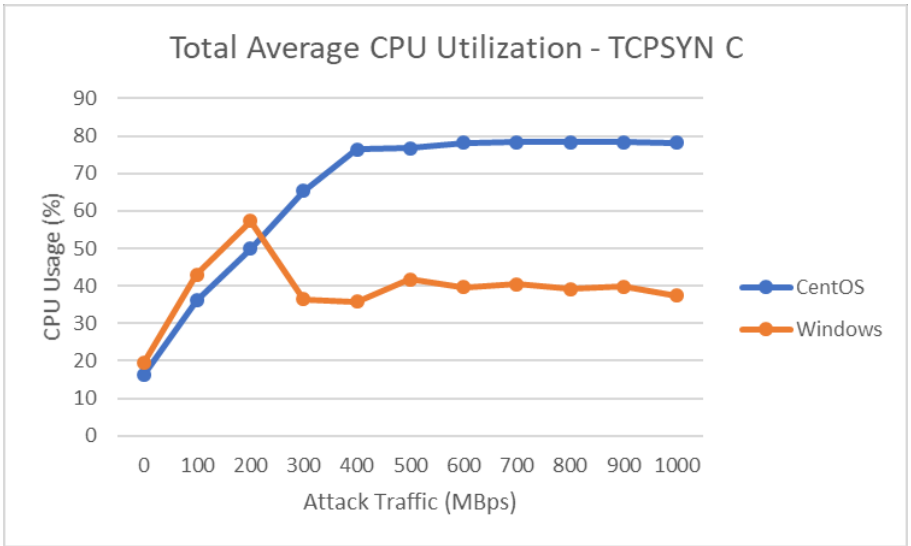

Figure IV-17 CentOS v Windows – TCPSYN Attack – Class C -Total Average CPU Usage

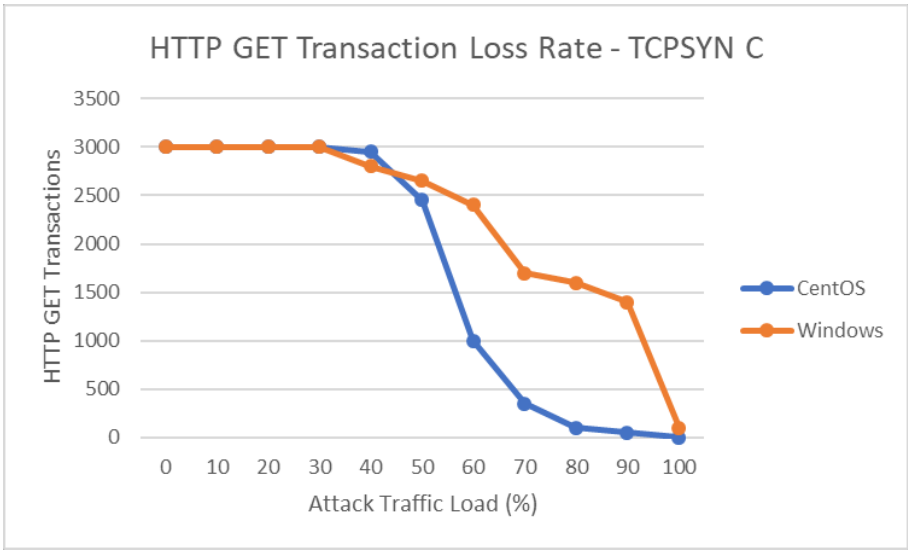

Figure IV-18 CentOS v Windows – TCPSYN Attack – Class C – HTTP Transaction Rate

UDP Flood Class A – 16M Hosts

The last major section of this chapter is the comparison of the Layer 4 attack, UDP Flood. Figure IV-19 shows a rapid increase in Windows CPU usage once attack traffic introduced. At 100 Mbps, the CPU usage jumps to 60%. From here, the value begins to rise and fall as the experiment continues, however the CU usage remains at an average 65%. According to Figure IV-20, Windows can keep connections open for the initial stages of the attack. Then there is a large drop in connections. However, after this large drop, the decline becomes steadier as the attack ramps up. This large drop, over 2000 connections per second, occurs relatively early in the attack, at 30% attack load. The amount of connections drops below 500 at the 60% attack load mark.

Under Class A UDP Flood, CentOS suffers more resource usage as well as lost HTTP connections. By 300 Mbps, CPU usage has reached 70% and slowly rises up to 80% at max attack load. Figure IV-19 shows a roughly 15% CPU usage difference in the later stages of the attack. The early stages of the attack also boast minimal connection losses, according to Figure IV-20. At the 30% attack load mark, we see the first signification decrease in connections available. After this we see the largest changes in transactions per second; by 50%, there are less than 500 connections available. CentOS HTTP connections dropped more steadily than Windows, but Windows was able to maintain more connections for a longer period.

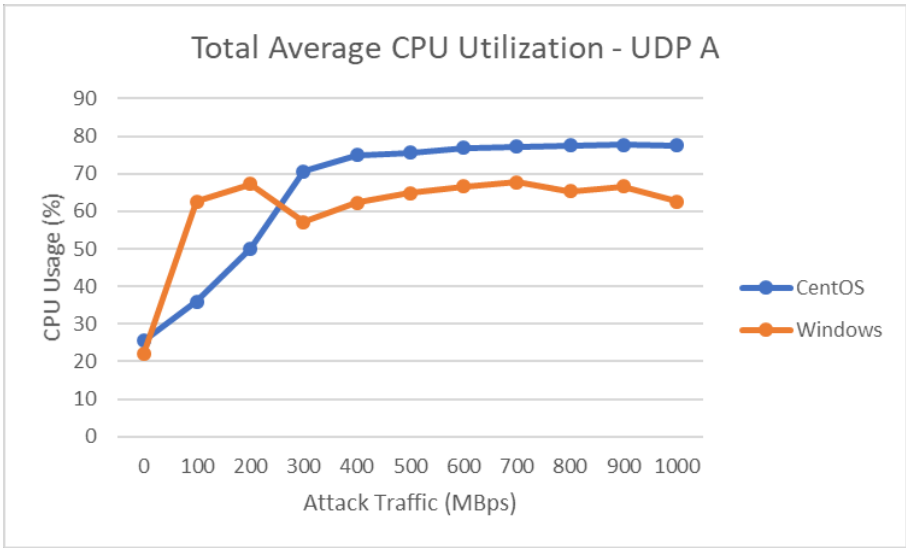

Figure IV-19 CentOS v Windows – UDP Flood – Class A – Total Average CPU Usage

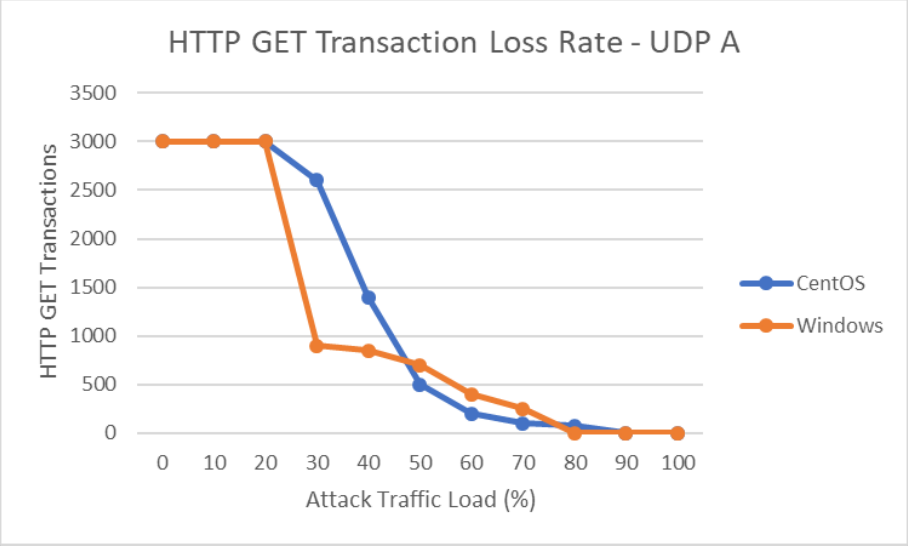

Figure IV-20 CentOS v Windows – UDP Flood – Class A – HTTP Transaction Rate

UDP Flood Class B – 65k Hosts

Like the previous section, Figure IV-21 shows an initial rapid rise in CPU usage.

However, after this jump, Windows is able to maintain this level of CPU usage. The CPU usage

remains in this 60-65% region for the remainder of the trial. Windows is able to maintain HTTP get connections for longer in this trial as well. A significant drop does not occur until the 40% attack load mark. After, it is a slow decline, about 5-10% per phase of the attack. Connections are maintained until the final phase of the attack where connections drop below 500 per second.

CentOS continues to be outperformed in this trial as well. Figure IV-21 shows a 10% rise in CPU usage per phase of the attack until about 400 Mbps, or 40% attack load. Here, the CPU usage peaks at 80% where it remains steady for the rest of the trial. The connection loss rate in Figure IV-22 shows that both operating systems are able to maintain all connections at the initial stages of the attack. CentOS begins to decline at 20% attack load, opposed to Windows at 30%. At this point, the connection loss for CentOS is very steady. It falls below 500 connects at 70% attack load and continues to fall from there.

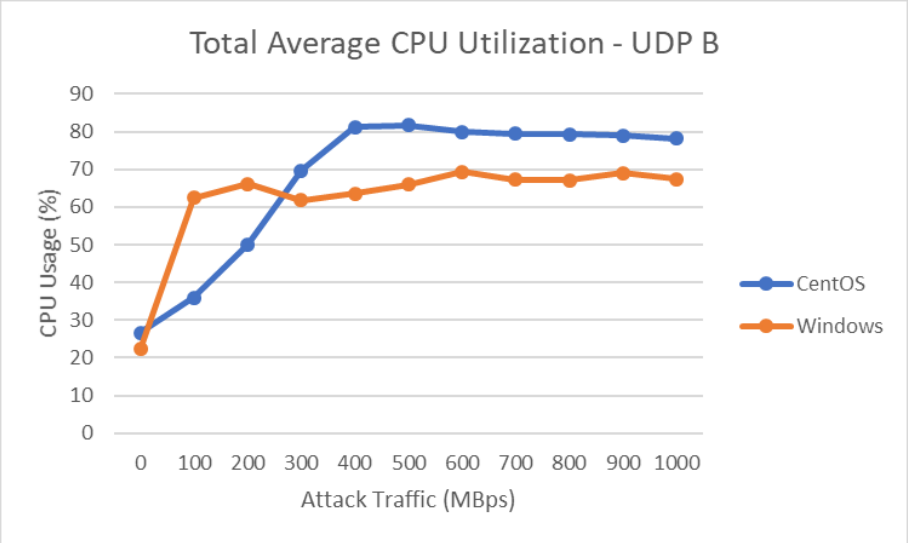

Figure IV-21 CentOS v Windows – UDP Flood – Class B – Total Average CPU Usage

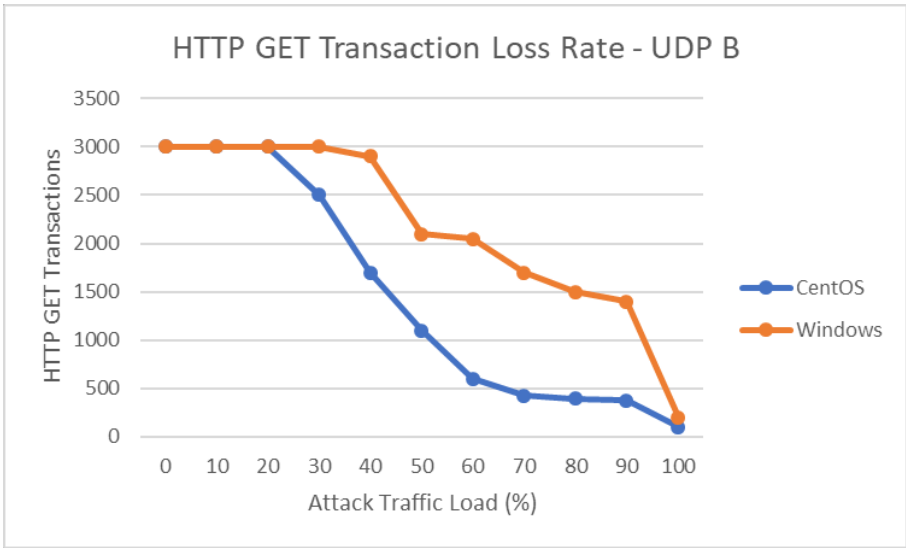

Figure IV-22 CentOS v Windows – UDP Flood – Class B – HTTP Transaction Rate

UDP Flood Class C – 254 Hosts

In this last section, there are similar results to the previous sections. Windows shows a significant jump initially, but the CPU usage remains around this level for the remainder of the attack. The CPU usage peaks at around 70% at max attack load. Windows is able to maintain all 3000 connections per second in the early stages of the attack here as well, Figure IV-23. However, after the 20% attack load mark, the connections are lost at a faster rate than CentOS. The rest of the attack shows the connections being lost steadily until the 80% mark where they drop below 500 and max load where they are all dropped completely.

Under UDP attack, it seems that Windows outperforms CentOS as the attacks go on for longer and the load increases. After 400 Mbps, the CPU usage value is above 70% and nearly 80% by the end of the attack. Initially it appears as though CentOS will maintain more connections, however as the attack continues, the losses increase at a larger rate than that of Windows Server. At 60% attack load, there are already less than 500 connections available. By

80% attack load, all connections are lost. When it comes to Layer 4 attacks, CentOS seems to perform better against lower load attacks. All the initial phase results indicate than as the attack load increases, the CentOS performance drops.

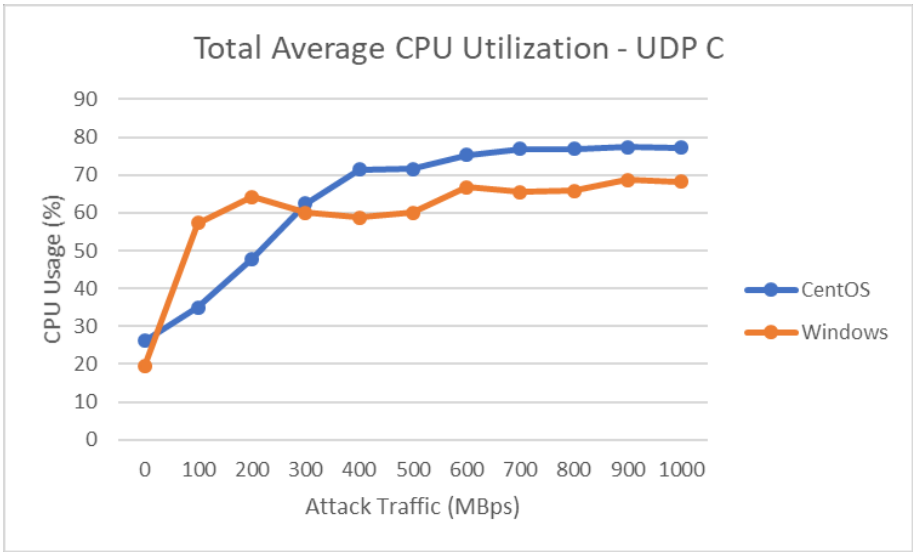

Figure IV-23 CentOS v Windows – UDP Flood – Class C – Total Average CPU Usage

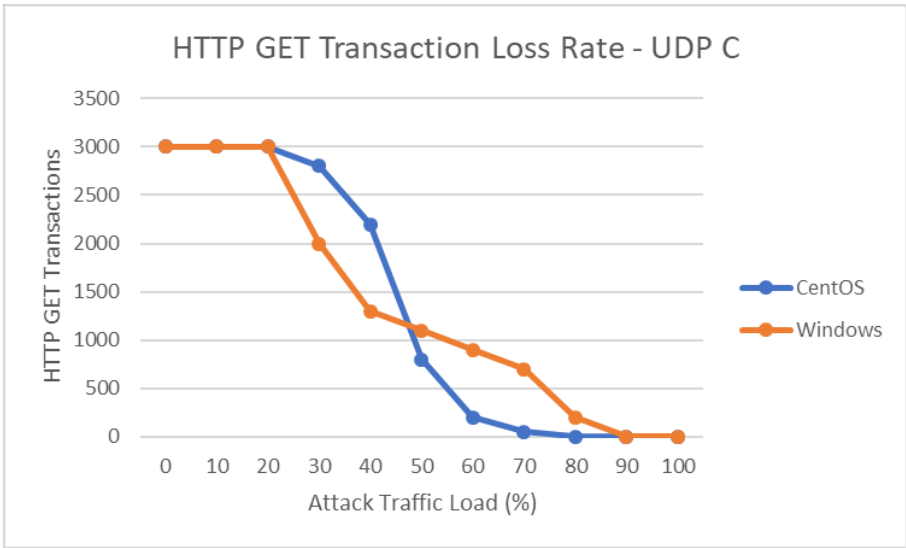

Figure IV-24 CentOS v Windows – UDP Flood – Class C – HTTP Transaction Rate

CHAPTER V

CONCLUSION

The final results of this experiment are varied, and it is not easy to say definitively which Operating System is "better". Each Operating System performed better or worse depending on the DDoS Attack being carried out. Regarding Layer 3 Attacks, Windows 2012R CPU Utilization outperformed CentOS CPU Utilization during all trials of Ping Flood attack while CentOS CPU Utilization outperformed Windows 2012R CPU Utilization during all trials of the Smurf Attack. The amount of HTTP Transactions with the Victim machine favor CentOS however, as more connections are available during every instance of Smurf Attack under Class C Ping Flood. Class A and B Ping Flood attack favor CentOS in HTTP Transaction rate as well. Regarding Layer 4 attacks, Windows 2012R CPU Utilization outperformed CentOS CPU Utilization during all UDP Flood trials. CentOS CPU Utilization outperformed Windows 2012R CPU Utilization for the duration of the Class A and Class B trials of TCP-SYN Flood. Class C TCP-SYN Flood trials show Windows 2012R CPU Utilization performing better under attack than CentOS. HTTP Transaction rates for Layer 4 attacks show Windows 2012R has more available connections under UDP Flood attack, as well as Class C TCP-SYN attack, which reflects the results of the CPU Utilization for the Layer 4 attacks. Overall, the results show that Windows Server 2012R CPUP Utilization typically held up well under Ping and UDP flood but was outperformed by CentOS CPU Utilization in Smurf Attack and TCPSYN Flood, apart from one trial where Windows did better. CentOS managed to maintain more HTTP Transactions

during nearly every attack except UDP Flood where Windows 2012R showed more available connections at the end of the trial for all Classes. Class C trials for Ping Flood and TCP-SYN Attack also show a larger amount of HTTP Transactions on Windows 2012R. The results did not reflect what was hypothesized, however new insight was provided. It is clear that one Operating System might be better suited for handling a specific task when compared with the other, from a performance perspective. Depending on the need or experience of the user, either Operating System can be applied.

REFERENCES

[1].Abrams, Lawrence. "Dramatic Increase of DDoS Attack Sizes Attributed to IoT Devices" Bleeping Computer, 12 Sept. 2018, https://www.bleepingcomputer.com/news/security/dramatic-increase-of-ddos-attack-sizes-attributed-to-iot-devices/.

[2].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.

[3].Goodin, Dan. “US service provider survives the biggest recorded DDoS in history.” Ars Technica, 5 Mar. 2018, https://arstechnica.com/information-technology/2018/03/us-service-provider-survives-the-biggest-recorded-ddos-in-history/.

[4].Kupreev, Oleg. Badovskaya, Ekatrina. Gutnikov, Alexander. “DDoS attacks in Q4 2019.”, Kaspersky Labs, 13 Feb. 2020, https://securelist.com/ddos-report-q4-2019/96154/.

[5].Shevchenko, Andrey. “Crypto Exchanges OKEx and Bitfinex Suffer Simultaneous DDoS Attacks.” CoinTelegraph, 28 Feb. 2020 https://cointelegraph.com/news/crypto-exchanges-okex-and-bitfinex-suffer-simultaneous-ddos-attacks.

[6].Boro, Debojit, et al. "UDP Flooding Attack Detection Using Information Metric Measure." Proceedings of International Conference on ICT for Sustainable Development. Springer, Singapore, 2016.

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

[8].B. Korniyenko and L. Galata, "Implementation of the Information Resources Protection Based on the CentOS Operating System," 2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON), Lviv, Ukraine, 2019, pp. 1007-1011..

[9].P. Blazek, T. Gerlich, Z. Martinasek and J. Frolka, "Comparison of Linux Filtering Tools for Mitigation of DDoS Attacks," 2018 41st International Conference on Telecommunications and Signal Processing (TSP), Athens, 2018, pp. 1-5.

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

[11].S. S. Kolahi, K. Treseangrat and B. Sarrafpour, "Analysis of UDP DDoS flood cyber attack and defense mechanisms on Web Server with Linux Ubuntu 13," 2015 International Conference on Communications, Signal Processing, and their Applications (ICCSPA'15), Sharjah, 2015, pp. 1-5.

[12].K. Treseangrat, S. S. Kolahi and B. Sarrafpour, "Analysis of UDP DDoS cyber flood attack and defense mechanisms on Windows Server 2012 and Linux Ubuntu 13," 2015 International Conference on Computer, Information and Telecommunication Systems (CITS), Gijon, 2015, pp. 1-5.

[13].N. Gupta, A. Jain, P. Saini and V. Gupta, "DDoS attack algorithm using ICMP flood," 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, 2016, pp. 4082-4084.

[14].R. R. Zebari, S. R. M. Zeebaree and K. Jacksi, "Impact Analysis of HTTP and SYN Flood DDoS Attacks on Apache 2 and IIS 10.0 Web Servers," 2018 International Conference on Advanced Science and Engineering (ICOASE), Duhok, 2018, pp. 156-161.

[15].H. A. Herrera, W. R. Rivas and S. 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), South Padre Island, TX, 2018, pp. 123-126.

[16].Postel, J., Information Science Institute, "Internet Control Message Protocol", RFC 792, University of Southern California, Los Angeles, September 1981. http://tools.ietf.org/html/rfc792.

[17].Postel, J., Information Science Institute, "Internet Protocol", STD 5, RFC 791, University of Southern California, Los Angeles, September 1981. http://tools.ietf.org/html/rfc791.

[18].Postel, J., Information Science Institute, “Transmission Control Pro-tocol” RFC 793, University of Southern California, Los Angeles, September 1981. http://tools.ietf.org/html/rfc793.

[19].Postel, J., Information Science Institute User Datagram Protocol RFC 768, University of Southern California, Los Angeles, September 1981. http://tools.ietf.org/html/rfc768.

[20].Eggert, L., Fairhurst, G. and Shepherd, G. UDP Usage Guidelines RFC 8085, NetApp, University of Aberdeen, Cisco Sytems, March 2017. https://tools.ietf.org/html/rfc8085.

[21].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.

[22].Kumar, Sanjeev, and Orifiel Gomez. "Denial of Service due to direct and Indirect ARP storm attacks in LAN environment." Journal of Information Security 1.02 (2010): 88.

[23].Kumar, Sanjeev, and Raja Sekhar Reddy Gade. "Experimental evaluation of juniper network's netscreen-5gt security device against layer4 flood attacks." Journal of Information Security 2.01 (2011): 50.

[24].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.

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

[26].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.

[27].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.

[28].Vellalacheruvu, Hari Krishna, and Sanjeev Kumar. "Effectiveness of Built-in Security Protection of Microsoft's Windows Server 2003 against TCP SYN Based DDoS Attacks." Journal of Information Security 2.03 (2011): 131.

[29].Michael Perez and Sanjeev Kumar, "A Quick Survey on Cloud Computing and Associated Security, Mobility and IoT Security Issues," Journal of Computer Communications, Vol.5, no 12, pp. 80-95, Oct 2017 – citations tracked by Web of Science, Google Scholar.

[30].Rodolfo Baez Jr., Sanjeev Kumar, "Apple's Lion Vs. Microsoft's Windows 7: Comparing Built-In Protection against ICMP Flood Attacks," Journal of Information Security, vol. 5, no.3, pp. 123-135, July 2014, 4807 Text views; citations tracked by Web of Science, Google Scholar, Impact Factor = 2.23

[31].Sirisha Surisetty and Sanjeev Kumar, "Microsoft vs. Apple: Resilience against Distributed Denial-of-Service Attacks." IEEE Security and Privacy, Vol.10, Issue 2, pp. 60-64, April 2012. DOI: 10.1109/MSP.2011.147 (1601 Text Views) Indexed by Web of Science, IEEE Xplore, Google Scholar.

[32].Leonel Aguilera, S. Kumar, "Web Performance of Windows Vs. Linux Servers under DDoS Attack," HESTEC Science Symposium, Sept. 2011. – Best Poster Award

[33].Einar Petana and S. Kumar, "TCP SYN-based DDoS attack on EKG signals monitored via a wireless sensor network," Wiley Journal of Security and Communication Networks, Jan 2011 (Online), Sept. 2011 (in print)

[34].Sirisha Surisetty and Sanjeev Kumar, “McAfee SecurityCenter Evaluation under DDoS Attack Traffic,”Journal of Information Security, vol. 2, no.3, pp. 113-121, July 2011; DOI: 10.4236/jis.2011.23011; ISSN Print: 2153-1234; ISSN Online: 2153-1242; 9298 Text views since publication- citations tracked by Web of Science, Google based Impact Factor = 2.23

[35].R. Gade, H. Vellalacheruvu, and S. Kumar "Performance of Windows XP, Windows Vista and Apple's Leopard Systems under a DDoS Attack," International Conference on Digital Society (ICDS'10), Feb. 2010. Available from IEEE online library Xplore

[36].R. Gade, H. Vallalacheruvu, S. Surisetty, and S. Kumar, "Impact of Land Attack Compared for Windows XP,Vista and Apple Leopard," Poster presentation, Science Symposium, HESTEC'09, Sept. 2009.

[37].S. Kumar and E. Petana, "TCP protocol attacks on Microsoft's Windows XP-based Computers," International Conference on Networking, pp. 238-242, April 2008. Available from IEEE online library Xplore

[38].S. Kumar, "PING Attack – How Bad Is It?" Computers & Security Journal, Vol.25, July 2006

[39].S. Kumar, M. Azad, O. Gomez, and R. Valdez, "Can Microsoft's Service Pack 2 (SP2) Security Software Prevent Smurf Attacks?" Proceedings of the Advanced International Conference on Telecommunications (AICT'06), Feb 2006. Available from IEEE online library Xplore

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

[41].Graham, Alan. “Bigger Online Super Series Cancelled Due to DDoS Attacks.” Bigger Online Super Series Cancelled Due to DDoS Attacks, Poker News Report, 5 Sept. 2017, www.pokernewsreport.com/bigger-online-super-series-cancelled-due-to-ddos-attacks-21870.

[42].Morbin, Tony. “National Lottery Hit by DDoS Attack - down 90 Mins at Peak Demand Time.” SC Media UK, 2 Oct. 2017, www.scmagazineuk.com/national-lottery-hit-by-ddos-attack--down-90-mins-at-peak-demand-time/article/697163/.

[43].S. Kumar, H. Kumar and G. Gunnam, “Evaluation of a Smart Electric Meter under Cyber Attacks,” IEEE International Conference on Data Intelligence and Security, ICDIS'19, June 2019.

[44].Barrera, C. Cheng and S. Kumar, “Hardware Implementation of Improved Mix Column Computation of Cryptographic AES,” IEEE International Conference on Data Intelligence and Security, ICDIS'19, June 2019.

[45].Roberto Talalolini and S. Kumar, “Automated Data Collection from production processes to Improve Industry Operation Efficiency,” IEEE International Conference on Data Intelligence and Security (ICDIS'19), June 2019.

[46].Edni Del Rosal and Sanjeev Kumar, "A Fast FPGA Implementation for Triple DES Encryption Scheme," Journal of Circuits and Systems, Vol.8, pp. 237-246, Sept 2017, (313 Text views) – citations tracked by Web of Science, Google Scholar.

[47].Munoz, S. Kumar "Buffer Management in the Sliding-Window Packet Switch," Int'l Journal of Communications, Network and System Sciences, vol.7, no. 7, pp. 448-255, July, 2014, (2595 Text views).

[48].Leonel Aguilera, S. Kumar, Jaime Ramos, "UTPA Solar System Efficiency," ASEE Conference, San Antonio, June 2012.

[49].Leonel Augilera, Jamie Ramos, S. Kumar, "UTPA Solar Array Efficiencies," HESTEC Science Symposium, Sept. 2011.

[50].Jaime Ramos, Leonel Aguilera, E. Garcia, S. Kumar, R. Garcia, Jose Sanchez "Commissioning a 5 KW PV Array for Electrical Engineering University Curriculum," ASEE Conference, Canada, June 2011.

[51].Sirisha Surisetty and Sanjeev Kumar, "McAfee SecurityCenter Evaluation under DDoS Attack Traffic," Journal of Information Security, vol 2, pp. 113-121, July 2011.

[52].S. Kumar and Orifiel Gomez, “Denial of Service due to direct and indirect ARP storm attacks in LAN environment,” – Journal of Information Security, vol. 2, no.3, pp. 88-94, Oct. 2010, DOI:10.4236/jis.2010.12010; ISSN Print: 2153-1234; 16,778 Text views - citations tracked by Web of Science, Google based Impact Factor = 2.23

[53].Leonel Aguilera Zambrano, Jamie Ramos, S. Kumar, "Validation of Power Data provided by Sunny WebBox for Renewable Solar Energy," HESTEC Science Symposium Sept. 2010, recipient of Best Poster Award.

[54].Sanjeev Kumar and Alvaro Munoz, "Comparison of Memory Assignment Schemes for Switch Architectures with Shareable Parallel Memory Modules," Journal of Electrical and Computer Engineering, June 2010.

[55].S. Surisetty and S. Kumar, "Evaluation of a Security Vulnerability in Apple's Leopard Operating System," International Conference on Internet Monitoring and Protection, May 2010. Available from IEEE online library Xplore

[56].S. Surisetty and S. Kumar, "Is McAfee SecurityCenter/Firewall Software Providing Complete Security for your Computer?" International Conference on Digital Society (ICDS'10), Feb. 2010. Available from IEEE online library Xplore

[57].H. Vellalacheruvu, S. Surisetty, R. Gade, and S. Kumar, "Evaluation of Syn-Cache protection against security Attacks," Poster presentation, Science Symposium, HESTEC'09, Sept. 2009.

[58].S. Surisetty, H. Vallalacheruvu, R. Gade, and S. Kumar, "Is McAFee Firewall really Protecting your System?," Poster presentation, Science Symposium, HESTEC'09, Sept. 2009 – Best Poster award

[59].S. Kumar and A. Munoz, "Performance Comparison of Switch Architectures with Parallel Memory Modules," IEEE Communications Letters, Vol.9, No.11, November 2005. (Journal's Acceptance Rate ~ 12%)

[60].S. Kumar and A. Munoz, "Memory-Bandwidth Performance of Shared Multi-Buffer Switch Versus Sliding-Window Switch," IEE Electronics Letters, Vol. 41, No.18, September 2005.

[61].S. Kumar and A. Munoz, "Switch Architecture for Efficient Transfer of High-Volume Data in Distributed Computing Environment," – Proceedings of the Workshop on High Performance Interconnects for Distributed Computing (HPI-DC 2005), in conjunction with IEEE HPDC'05, Research Triangle Park, North Carolina, July 2005.

[62].S. Kumar "On Impact of Distributed Denial of Service (DDoS) attack due to ARP storm," – Lecture Notes in Computer Science – Book Series, LNCS-3421, Networking - ICN 2005, Part-II, Publisher: Springer-Verlag, April 2005.

[63].Munoz and S. Kumar, "Effect of Unbalanced Bursty Traffic on Memory-Sharing Schemes for Internet Switching Architecture," Lecture Notes in Computer Science- Book Series, LNCS-3420, Networking - ICN 2005, Part-I, Publisher: Springer-Verlag, April 2005.

[64].S. Kumar and Charles Harlow, "Networking Research for BioMedical Applications," GM Wireless Networking Seminar Presentation, Department of Electrical and Computer Engineering, The University of Texas at Austin, Texas, Oct. 2004.

[65].Munoz and S. Kumar, "On Design of Internet Routers/Switches Deploying Parallel Memory Modules," Proceedings of the International Conference on Computing, Communications and Control, Austin, Texas, August, 2004.

[66].H. Robles and S. Kumar, "Throughput Measurements of a Cisco-2600 Router," Proceedings of the International Conference on Computing, Communications and Control, Austin, Texas, August, 2004.

[67].S. Kumar and T. Doganer, "Memory-Bandwidth Performance of the Sliding-Window based Internet Routers/Switches," Proceedings of the IEEE Workshop on Local and Metropolitan Area Networks, San Francisco, CA, April 2004.

[68].S. Kumar, T. Doganer, A. Munoz, "Effect of Traffic Burstiness on Memory-Bandwidth of the Sliding-Window Switch Architecture," Proceedings of the International Conference on Networking, March 2004.

[69].S. Kumar, A. Munoz, T. Doganer, "Performance Comparison of Memory-Sharing Schemes for Internet Switching Architecture," Proceedings of the International Conference on Networking, March 2004.

[70].S. Kumar, "A Novel Architecture for High Performance Internet Routing and Switching," IEEE Dallas Chapter meeting, Richardson, Texas, December 16, 2003. (Invited presentation).

[71].S. Kumar, "The Sliding-Window Packet Switch: A New Class of Packet Switch Architecture with Plural Memory Modules and Decentralized Control," IEEE Journal on

Selected Areas in Communications, Vol.21, No. 4, pp. 656-673, May 2003. (Impact factor: 10.2)

[72].S. Kumar, "Are Packet-Switches Delivering Bandwidth Capacity as Advertised?" 8th Annual The University of Texas System ITDE conference, May 2002.

[73].S. Kumar and J. Rios, "Trends in Optical Networking," 8th Annual The University of Texas System ITDE conference, May 2002.

[74].S. Kumar, "Multistage subtended DSLAM for cost effective deployment of ADSL networks," Comprehensive xDSL, International Engineering Consortium, 1999, ISBN: 0933-217-59-5. (Book Chapter)

[75].S. Kumar, "Lowering the Cost of Broadband Access in the local loop," AMTC, Aug. 1999, San Diego, CA.

[76].S. Kumar, "ADSL deployment issues & low-cost ADSL architecture," ISPcon, Oct. 1998, San Jose, CA.

[77].S. Kumar, "Commercial Deployment of DSL with DSLAM+," AiC, September 1998, Atlanta, GA.

[78].S. Kumar, "Impact of Backbone Switch Architecture on the Performance of ATM based ADSL network," AMTC, August 1998, Atlanta, GA

[79].S. Kumar, "DSLAM+ for Cost Effective Residential Broadband Deployment," XIWT (Cross Industry Working Team) meeting, August, 1998, Intel Corporation Facility, Hillsboro, Oregon. (Invited Presentation).

[80].S. Kumar, "DSLAM+: The multistage subtended DSLAM for cost effective deployment of ADSL networks," XDSL COMFORUM, International Engineering Consortium, July 21-22, 1998, Chicago.

[81].S. Kumar, "ADSL Deployment Realities – A Cost Effective approach," DSLcon, March 98, San Jose, CA.

[82].S. Kumar and D.P. Agrawal, "Design and Performance of Shared-Buffer Direct-Access (SBDA) ATM Switch Architecture for Broadband Networks," International Journal of Computer Systems and Engineering, vol. 12, no. 2, pp. 69-79, March 1997.

[83].S. Kumar and R. Dantu, "The Era of the Full Service Networks," America's Network, March 15, 1997.

[84].S. Kumar and D.P. Agrawal, "The Sliding-Window Approach to High Performance ATM Switching for Broadband Networks, " IEEE GlobeCom, November 1996.

[85].S. Kumar and D.P. Agrawal, "On multicast support for Shared-Memory based ATM Switching Architectures," IEEE Network, pp. 34-39, January 1996. (Impact factor: 7.02)

[86].S. Kumar and D.P. Agrawal, "Performance of the class of Sliding-Window ATM Switch Architectures for Broadband Communications Networks," IEEE SOUTHEASTCON , March 1995.

[87].S. Kumar and D.P. Agrawal, "A Shared-Buffer Direct-Access Switch Architecture for ATM-based Networks," IEEE International Conference on Communications, pp 101-105, May 1994.

[88].S. Kumar and D.P. Agrawal, "On Design of a Shared-Buffer based ATM-Switch for B-ISDN," IEEE Int'l Phoenix Conference on Computers & Communications, pp 377-383, April 1994.

[89].S. Kumar and D.P. Agrawal, "Efficient Reconfiguration of 2-D Rectangular Systolic Arrays," IEEE Int'l Phoenix Conference on Computers &Communications, pp 16-22, March 1993.

[90].S. Kumar and D.P. Agrawal, "On Design of Easily Reconfigurable Systolic Computation Networks," International Conference on Signals,Data and Systems, pp 86-93, December 1992.

[91].S. Kumar and D.P. Agrawal, "Design And Analysis of a Highly Reconfigurable Hexagonal Systolic Arrays," Government Microcircuit Applications Conference, Vol XVII, pp 53-56, June 1991.

[92].“Working with the Apache Configuration File.” Working with the Apache Configuration File - Passenger Library, www.phusionpassenger.com/library/install/apache/working_with_the_apache_config_file.html.

[93].“How to Install Apache on CentOS 7.” Linode Guides & Tutorials, 1 June 2018, www.linode.com/docs/web-servers/apache/install-and-configure-apache-on-centos-7/.

[94].“Introduction to FirewallD on CentOS.” Linode Guides & Tutorials, 23 Aug. 2019, www.linode.com/docs/security/firewalls/introduction-to-firewalld-on-centos/.

[95].Godard, S. SYSSTAT, sebastien.godard.pagesperso-orange.fr/tutorial.html.

[96].Costantini, Davide. “How to Install and Configure IIS on Windows Server 2012 R2.” The Solving, 6 Mar. 2018, thesolving.com/server-room/how-to-install-and-configure-iis-on-windows-server-2012-r2/.

[97].Kalin, Marty. "CFS: Completely Fair Process Scheduling In Linux". *Opensource.Com*, 2019, https://opensource.com/article/19/2/fair-scheduling-linux.

[98].Freeman, Allen. "How To Spy On Your "Buddy's" Network Traffic: An Intro To Wireshark And The OSI Model". *Wonderhowto*, 2012, https://null-byte.wonderhowto.com/how-to/spy-your-buddys-network-traffic-intro-wireshark-and-osi-model-0133807/.

[99]."Changes Introduced In Macos Server 5.7.1". Apple Support, 2020, https://support.apple.com/en-us/HT208312.

[100]."Linux Vs. Windows: A Comparison Of The Best Web Server Solutions". *IONOS Digitalguide*, 2020, https://www.ionos.com/digitalguide/server/know-how/linux-vs-windows-the-big-server-check/.

BIOGRAPHICAL SKETCH

William Robert Rivas was born December 23, 1991. He has earned his Bachelors of Science in Electrical Engineering from the University of Texas Rio Grande Valley in Edinburg, Texas in December 2015.He has completed his Master of Science in Electrical Engineering from the University of Texas Rio Grande Valley in December 2020. He worked as an Assistant Technician for the Engineering Department at UTRGV from January 2013 to December 2015. He worked as a Teaching Assistant in the Engineering Department at UTRGV from January 2016 to May 2020. He worked as a Research Assistant in the Network Research Lab at UTRGV.

Permanent Address:

2006 Village Dr. Mission, Texas 78572

Email:wr4842@gmail.com

Publication:

1. H. A. Herrera, W. R. Rivas and S. 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), South Padre Island, TX, 2018, pp. 123-126.