**Testing Various Attacks Against Host-Based and Network-Based Firewalls**

****

**Group 11:** Lisa Cheung (CpE), Siva Prabakar (CpE), Roger Sengphanith (CpE), Krystal Wang (CpE)

**Abstract**

In this project, we will be analyzing various types of firewalls to determine their ease of use, capabilities, and their shortcomings. We will be using the default firewall found on Linux, configurable with the “iptables” command as well as the firewalls found on various routers. As the world becomes more and more internet and computer-oriented, it is only natural that more crackers will come out of the woodworks to exploit the vulnerabilities that present themselves. The risk of valuable data being taken increases with the amount of people who are willing to steal and attack from behind their computer monitor. We test the effectiveness of firewalls as well as host-based and network-based intrusion detection systems on three different kinds of common attacks: SYN Flooding, site filtering, and spyware. The SYN Flood, a popular attack that is easy to initiate, was mitigated to the point where it was ineffective, albeit not completely deterred. Though both got the job done, our network-based IDS performed better compared to our host-based IDS. For site filtering, it seems that a dedicated firewall is superior to a network-based one, as it can block HTTPS. Spyware was successfully blocked by the the firewalls, but in realistic cases it may not be so easy to do so. Overall, the strengths and weaknesses of the host-based versus the network-based firewalls can only be described as situational. In realistic situations, they are both implemented. It is important that they fulfill their role as crackers continue to increase their ability to intrude into places they do not belong.

**I. Table of Contents**

1. Table of Contents
2. Introduction
3. What is a Firewall?
4. Experimental Methodology and Results
   1. SYN Flooding
      1. Background
      2. Methodology
      3. Results and Analysis
      4. Conclusion
   2. Site Filter
      1. Background
      2. Methodology
      3. Results and Analysis
      4. Conclusion
   3. Spyware
      1. Background
      2. Methodology
      3. Results and Analysis
      4. Conclusion
5. Challenges and Solutions
6. References

**II. Introduction**

The advent of computers revolutionized the ways in which humans can store, retrieve, manipulate, and communicate information. The use of computers to store, retrieve, transmit, and manipulate data is known as information technology (IT). Nowadays, not only are computers ubiquitous, they are also connected to each other through the Internet. In fact, a broader term, the Internet of Things (IoT) was coined to express the extension of internet connectivity into physical devices and everyday objects. From individuals and small business, to large corporations and even governments, organizations require them to continue their internal operations and business solutions [9]. Despite the importance of the internet and computer networks, security is a big concern. The data stored on computers and the data in motion across networks can be accessed by outside parties if the networks or the computers are not secure enough. Sometimes, even secure networks and computers can be broken into.

Those who take advantage of network vulnerabilities to break into secure networks for malicious purposes are known as crackers. On the other hand, hackers are professionals who study network vulnerabilities and fix them, also known as ethical hacking. Though pop culture typically refers to crackers as hackers, they are different. The battle between crackers and security professionals has constantly evolved since computers were first made. Crackers and hackers constantly one up each other by finding new exploits and coming up with new security measures respectively. One byproduct of the struggles between crackers and hackers are intrusion detection systems. Intrusion detection systems are methods of security management for computers and networks that collect information from systems and network sources. They analyze the data to detect activity that shows signs of any intrusions that may lead to an attack on the system that is being defended. The two types of intrusion detection systems are host-based intrusion detection systems (HIDS) and network-based intrusion detection systems (NIDS).

Host-based intrusion detection systems aim to collect the information about activity on a specific system, hence its name. It only operates to protect the host that it is located on [10]. These systems are typically implemented on machines which are determined to be particularly susceptible to potential attacks. A host-based system can observe information which can clarify who did what, who accessed what, and when they did it. They trace malicious and suspicious activities to a specific user, which is assigned an ID. The downside to this is, because they are tied to a single host, they are unable to observe the overall network’s traffic.

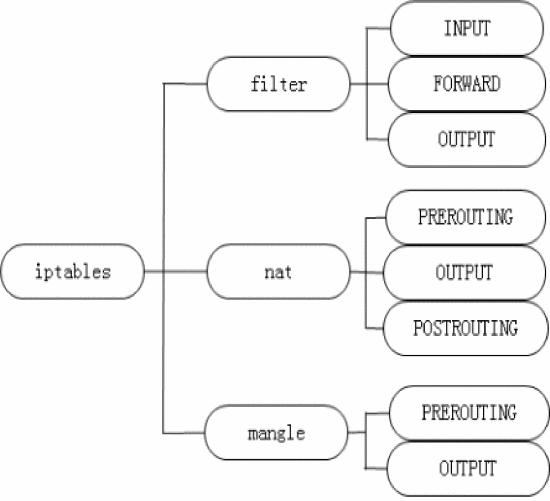
Network-based intrusion detection systems come from a different approach, where the systems aim to monitor and then collect information from the network itself. From before, the downside to a host-based intrusion detection system was said to be that it was not designed to observe the network traffic. This system operates much like it is wiretapping information that is collected from the traffic system, where the data travels. It checks for attacks by inspecting the headers and payloads of any and all packets navigating through the network.

The goal of this project is to test and compare the effectiveness of a network-based firewall versus a host-based firewall by incorporating both into several scenarios, namely Dos attacks, spyware, and site filtering.

**III. What is a FireWall?**

[1] By definition, a firewall is a means of protection of the internal network against malicious networks. It is part of a network that designed to block unauthorized access, while also permitting authorized communications. The firewall’s purpose is able to be configured, to perform more advanced operations such as permission, denial, encryption, decryption, and proxying all the computer traffic between varying security domains. This is a possibility because of the firewall’s ability to filter incoming and outgoing traffic. All messages which enter or leave the intranet will pass through the firewall, which then inspects each message and will block those who do not meet a specified security criteria. All in all, firewall is the mechanism which functions as access control situated in between the network or system.

In this project, we intend on using the Linux OS. Linux makes use of Iptables, which is its own host-based firewall. It has been the Linux firewall ever since version 2.4. [2] It functions as the filter management tool of the kernel packet.



*Figure 3.1 Iptables Structure*

In figure 3.1, it can be seen that the structure of Iptables consists of 3 main parts: The table, 3 rules, and 8 chains. The rules are the conditions for selecting packets and determining what should be done with the specified packets. The chains are the rules that have been grouped. It is a set of rules that will decide what will be done with the packets. Finally, the chains are grouped into tables under the names filter, NAT, and mangle.

It provides 3 functions:

* **Packet Filtering Table:**

The system’s configuration default, which is then comprised of 3 chains labeled INPUT, FORWARD, and OUTPUT. The three chains cover three different situations that result from when a data packet enters. When the packet’s destination is local, the system is is sent to the INPUT. When the packet’s destination is otherwise, the system will forward the packet to FORWARD. Finally, when the packet is generated by the local system, it will be sent to the OUTPUT.

* **Network Address Translation Table:**

Changes the addresses of the packets according to a set of rules. This particular function allows the internal network structure to achieve something called IP masquerading, which is when the external real IP address can have multiple addresses be mapped to. The PREROUTING chain comes into effect when the address translation occurs before routing, and is meant for source address. The POSTROUTING chain comes into effect when the address translation occurs after routing, and is used for the destination address. Finally, the OUTPUT chain comes into effect for the network address translation for packets which are generated by the firewall.

* **Packet Processing Table Mangle:**

The processing can modify field values in data packets save for the IP address and the port, and can also mark the packets for more processing in the future. It is used to alter fields of the IP headers, such as TTL (Time to Live), TOS (Type of Service). The PREROUTING chain operates before the packet has routing take effect. The OUTPUT chain is used on internally generated packets before routing begins. All in all, the the purpose of this table is to transform packets.

A packet is checked using each rule, starting from the top. Depending on the rules, the actions which are then taken are either ACCEPT or DROP (dropping the packet).

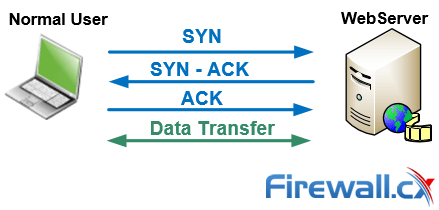
**IV. Experimental Methodology and Results**

1. **SYN Flooding:**
   1. **Background:**

[7] SYN Flooding is a type of Denial of Service (DOS) attack that takes advantage of the TCP three-way handshake, where TCP is a standard that defines how to establish as well as maintain a network conversation between application programs that are able to exchange data. [3]The handshake mentioned is the process of initiation, allowing for data transition to begin. Contrary to the ease of execution, guarding against it is much, much harder to do.In 2016, approximately 75% of all used attacks was the TCP SYN Flood.

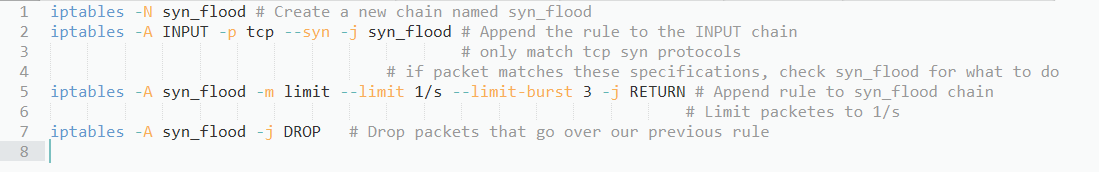
It is one of the most dangerous types, where the attackers will often use botnets to send overwhelmed spoof SYN packets. It overloads the network bandwidth with a massive amount of SYN request packets. [4] These packets contain information fields just as a normal packet would, so that the victim is unable to differentiate between which packet is normal and which one is the spoof . The attackers try to send flooding packets to the victim without allowing the complete process of the TCP handshake to go through, so the victim in return must spend resources to make amends for the incomplete connections, leading to exhaustion of resources and incapacitation from proper serving of requests from actual connections made. The legitimate users will be rendered unable to access the targeted server.

[1] The attack is very basic: an attacker sends multiple SYN messages, which in turn causes the victim to send SYN-ACK messages. However, the attacker never receives or responds to the SYN-ACK, which leaves the victim stuck in a waiting state, until the connection times out. This will cause the overflow tables to fill, causing denials to legitimate clients or crashes. Though there have been many studies on how to prevent this sort of attack, it is still heavily exploited and effectively abused by many.



* 1. **Methodology:**
     1. **Iptables:**

We will be attempting to mitigate the effects of SYN Flooding in this experiment. We have set up two Linux machines, one attacker and one victim. On the attacker computer, we have metasploit installed to cause the SYN flooding, and the victim with wireshark installed, to monitor packets. Then, we setup the firewall on the victim’s machine using the following commands:



*Figure 4.1 Firewall Setup*

We then restarted the SYN flood.

* + 1. **Router:**

To test the network-based intrusion detection system, we will attack the network using a linux machine while using different router firewall settings. We used the hping3 package for the TCP SYN Flood attack. Hping3 is a popular TCP penetration testing tool that can be installed on any linux distribution. Using the command shown below, we are able to conduct a SYN Flood attack:



*Figure 4.2 hping3 SYN Flood Command*

Breaking down this command, we are sending 15000 packets (-c 15000) that have a size of 120 bytes each (-d 120). ‘-S’ specifies that the SYN Flag should be enabled and ‘-w 64’ sets a TCP window size of 64. ‘-p 80’ directs our attack at port 80 in order to attack our victim’s HTTP web server. We use the ‘--flood’ flag to send packets as fast as possible and use ‘--rand-source’ to generate random spoofed IP addresses to not only disguise the real source and hide the attacker, but also to stop the victim’s SYN-ACK reply packets from reaching the attacker.

We used the default router provided by Spectrum, the ISP, and used two different firewall settings present in the router. We ran the SYN Flood attack with the firewall level at ‘low’ and then again with the firewall level set to ‘high’. The firewall setup was easy as we just had to select the level and apply. By default, the firewall level is set to medium.



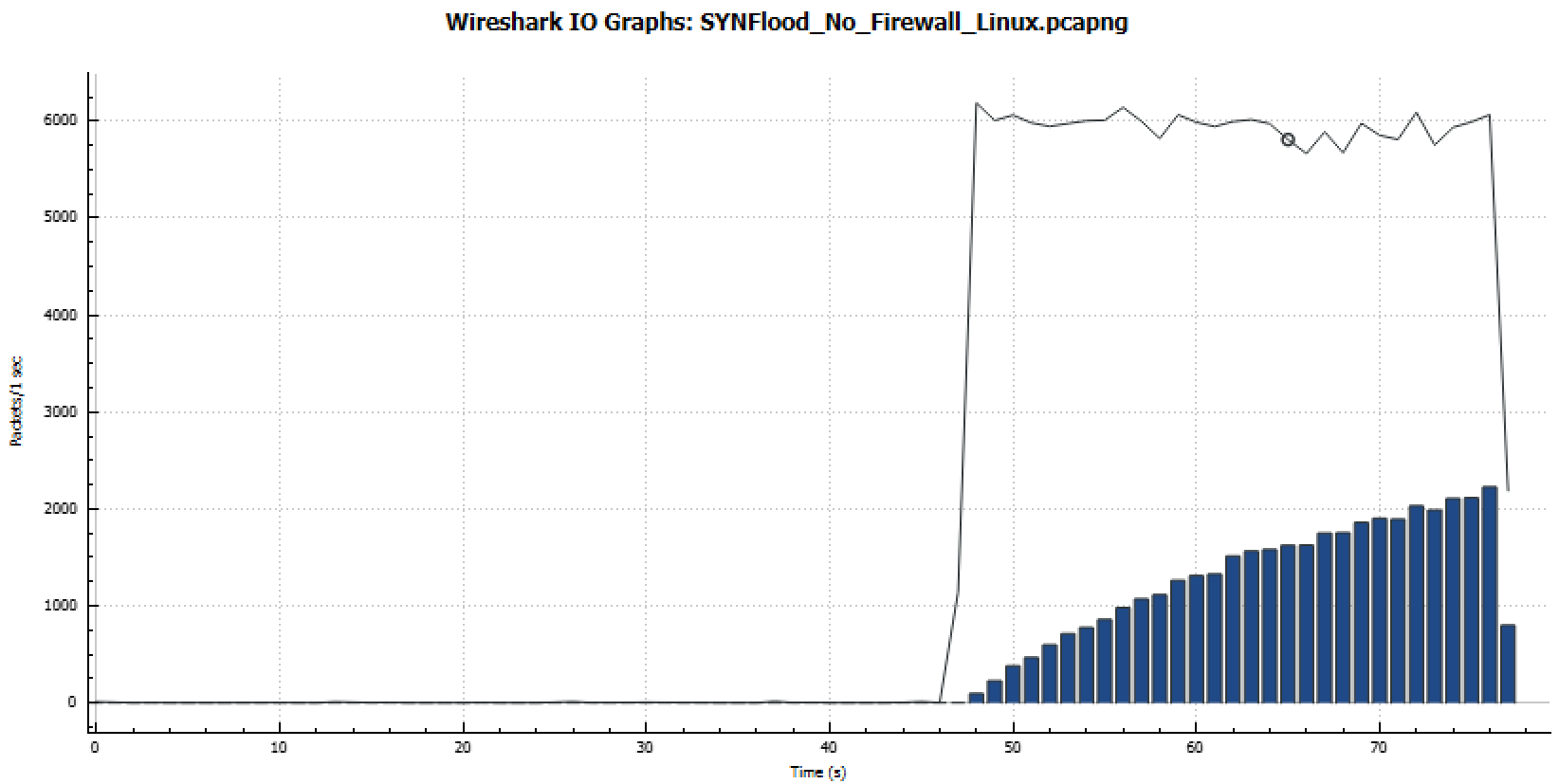
*Figure 4.3 Firewall Setup For Router*

* 1. **Results and Analysis:**

Typically, a SYN Flood attack will involve a suddenly high volume of SYN packets with varying source IP addresses that have a destination port of 80 (HTTP). When the filter tcp.flags.syn==1 and tcp.flags.ack==0 is applied, we see that when the filter tcp.flags.syn==1 and tcp.flags.ack==1 is applied there is a comparatively very small number of SYN/ACKS.

* + 1. **Vulnerable machine:**

From the packet capture, we can see that between the first packet and the last packet captured, approximately 30 seconds have passed. In that time, 87934 SYN packets have been sent, and the same number of RST,ACK packets have been sent. In Wireshark, it also becomes obvious when we pull up the IO graph:

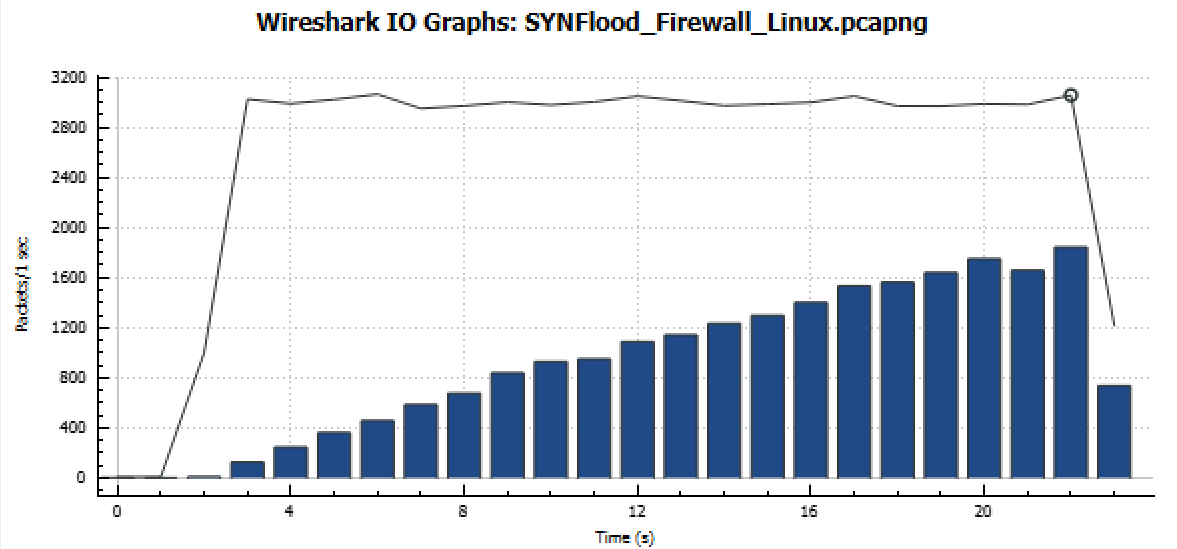


*Figure 4.4 SYNFlood\_No\_Firewall\_Linux.pcapng*

There is a sudden spike of around 6000 packets within a short span of time. This is an obvious sign of a SYN Flood attack occurring.

* + 1. **Iptables:**

After filtering, we see that in about 20 seconds, our attacker has sent 62173 SYN packets, but our victim has not responded to any of them, showing that our firewall rule is working properly. We can also see that the attacker’s ip address is different than before, telling us that it is easy to spoof the source ip, and that simply blocking a single ip will not work.

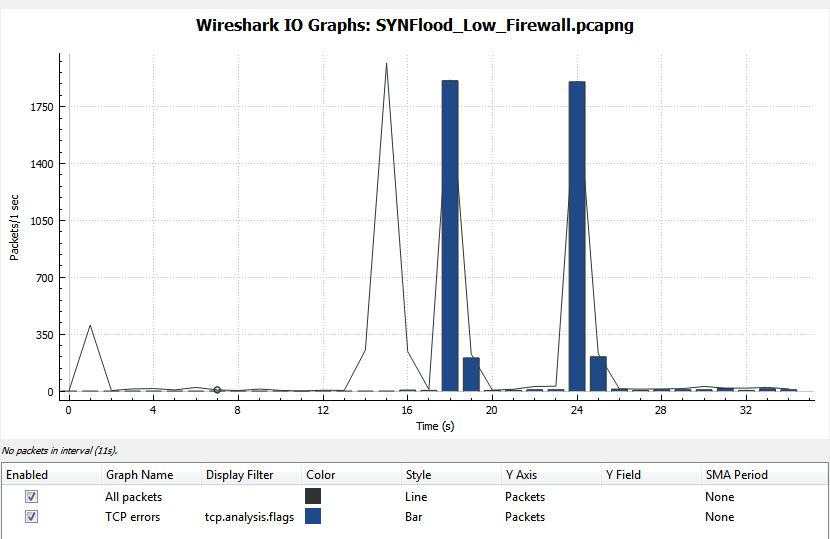
****

*Figure 4.5 SYNFlood\_Firewall\_Linux.pcapng*

Compared to the near instantaneous insertion of 6000 packets/1 sec, we see a mitigated version when firewall is active.

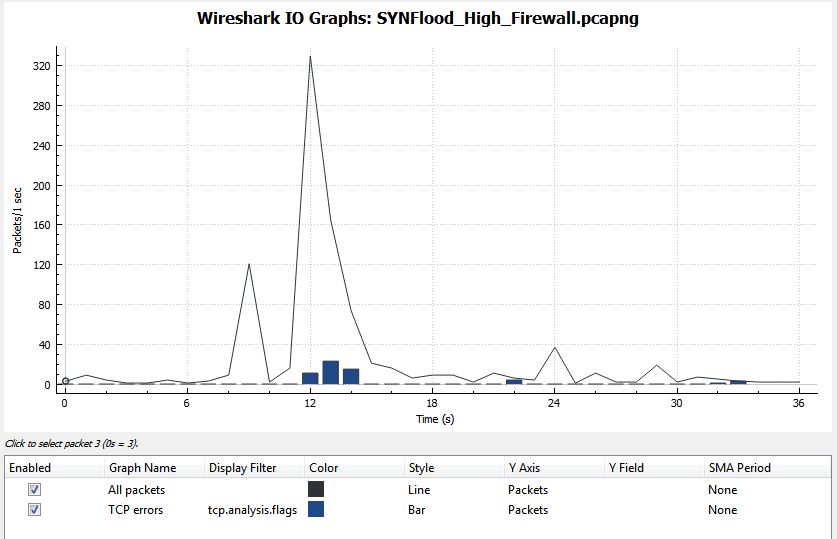
* + 1. **Router**

For both firewall settings, I started the attack at around 12 seconds and let it run until 35 seconds. The IO graphs for both runs are shown below.



*Figure 4.6 SYNFlood\_Low\_Firewall.pcapng*

The above IO graph resulted from our firewall level set to low. At around 15 seconds, you can see the huge jump in the number of packets being received to around 2000 packets being received a second, right after I started the attack. There was also a spike of TCP errors as indicative of a SYNFlood attack.



*Figure 4.7 SYNFlood\_High\_Firewall.pcapng*

The above IO graph is with our firewall level set to high. We can see that the SYNFlood attack is more or less mitigated. At its peak, we only receive about 300 packets/ second and there are very little TCP errors so the peak is not really related to the SYNFlood attack.

* 1. **Conclusion**

We can see that Linux’s default firewall does a pretty good job at mitigating the effects of the DOS attack, being able to ignore which IP address sends the attack as well as from which port the attack is being carried out on. However, the router’s firewall does an even better job at mitigating the attack. While Linux’s default firewall almost halved the number of packets being received, the router’s firewall reduced the number of packets being received by almost ⅙. The number of tcp errors occurring using our network based router firewall is far less than when we used the host-based linux firewall.

1. **Site Filtering**
   1. **Background:**

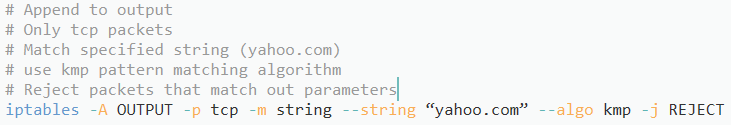
In a workplace, we may want to prevent employees from accessing certain websites that may be unproductive, or websites that are known to contain malware. [5] To even greater extents, more than 40 countries actively filter web access, which blocks out billions of internet users worldwide. The best known reason for these actions are related to politics and socially sensitive issues.

There are many ways to perform site filtering. For example, IP blacklists. IP blacklists block all content from a specified IP address. They can also be deployed at proxy-based filters. Proxy servers are often used to cache content locally, making it an attractively easy location to filter from.

In this experiment, we will attempt to block access to certain websites.

* 1. **Methodology:**
     1. **Iptables:**

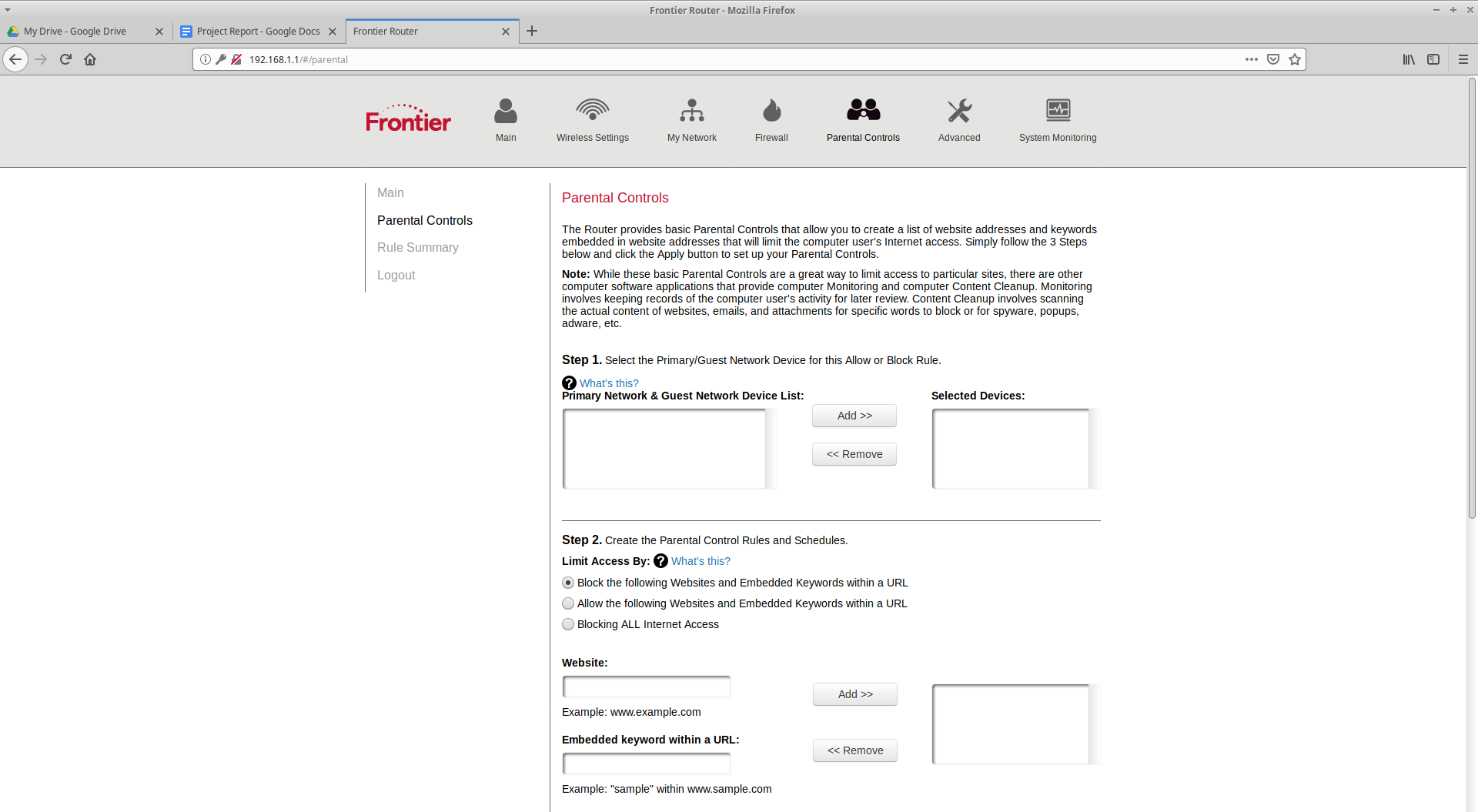
To block websites, we can run the following command: (Figure 4.8)



*Figure 4.8 Blocking specific websites (“*[*www.yahoo.com*](http://www.yahoo.com)*”)*

* + 1. **Router:**

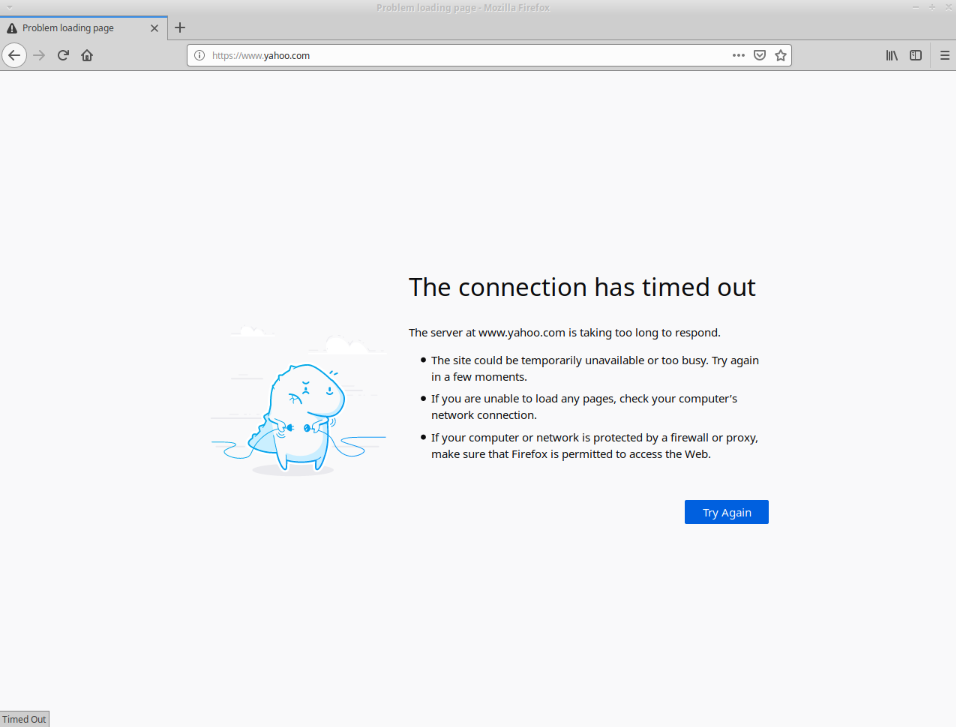
To block websites, we utilized the parental controls feature of our router:



*Figure 4.9 Router*

* 1. **Results and Analysis:**
     1. **Iptables**

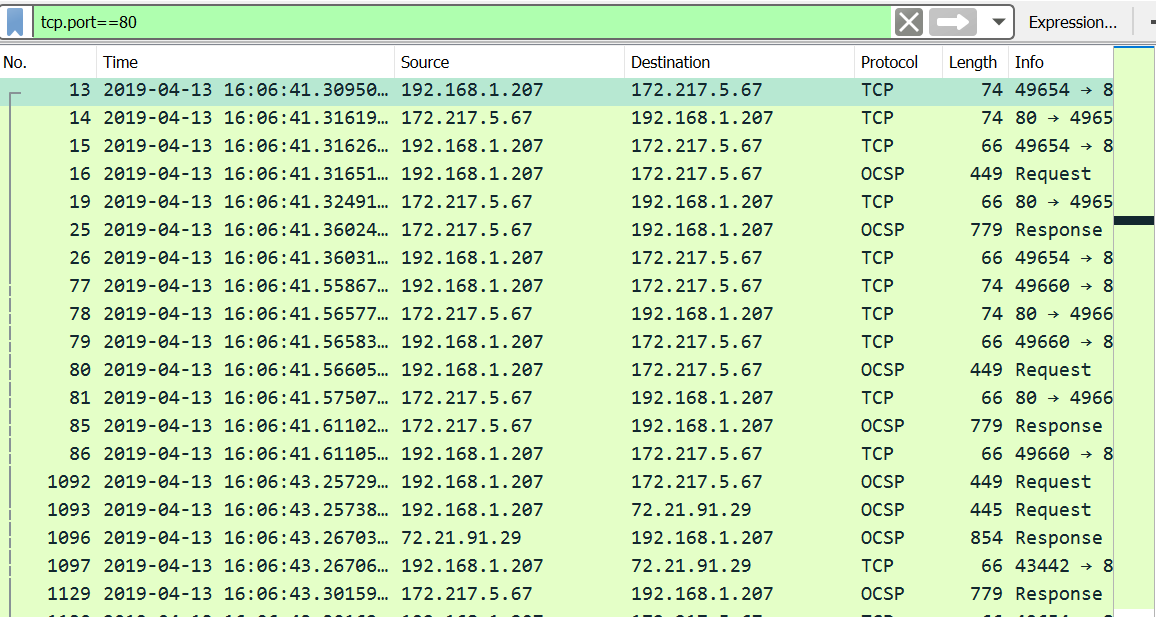
The results of entering the command from previously yields our site but with a “The connection has timed out” error instead. Because of the command, we have rejected the request to access “[www.yahoo.com](http://www.yahoo.com),” as seen in Figure 4.10.

****

*Figure 4.10 Error Page*

We test the following scenarios:

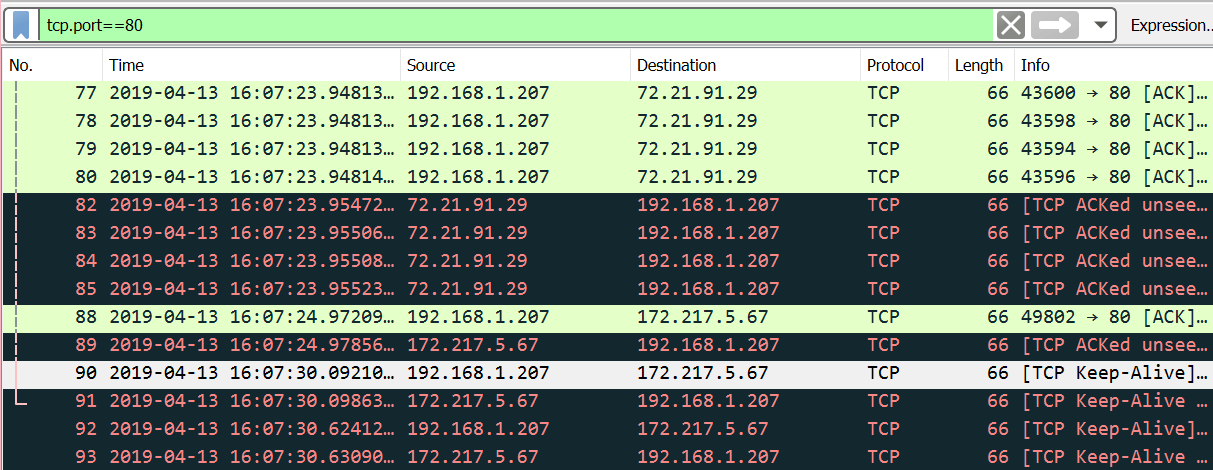
1. **Without “**[**www.yahoo.com**](http://www.yahoo.com)**” being filtered:**



*Figure 4.11 “*[*www.yahoo.com*](http://www.yahoo.com)*” without the filter in Wireshark*

In figure 4.11, we see the usual routine of sending requests to the desired accessed site, and receiving of responses from said site by setting the filter to tcp.port==80, which is the port for Hypertext Transfer Protocol. With minimal warning from Wireshark, most of the “handshaking” done by the TCP is performed safely and normally.

1. **With “**[**www.yahoo.com**](http://www.yahoo.com)**” being filtered:**

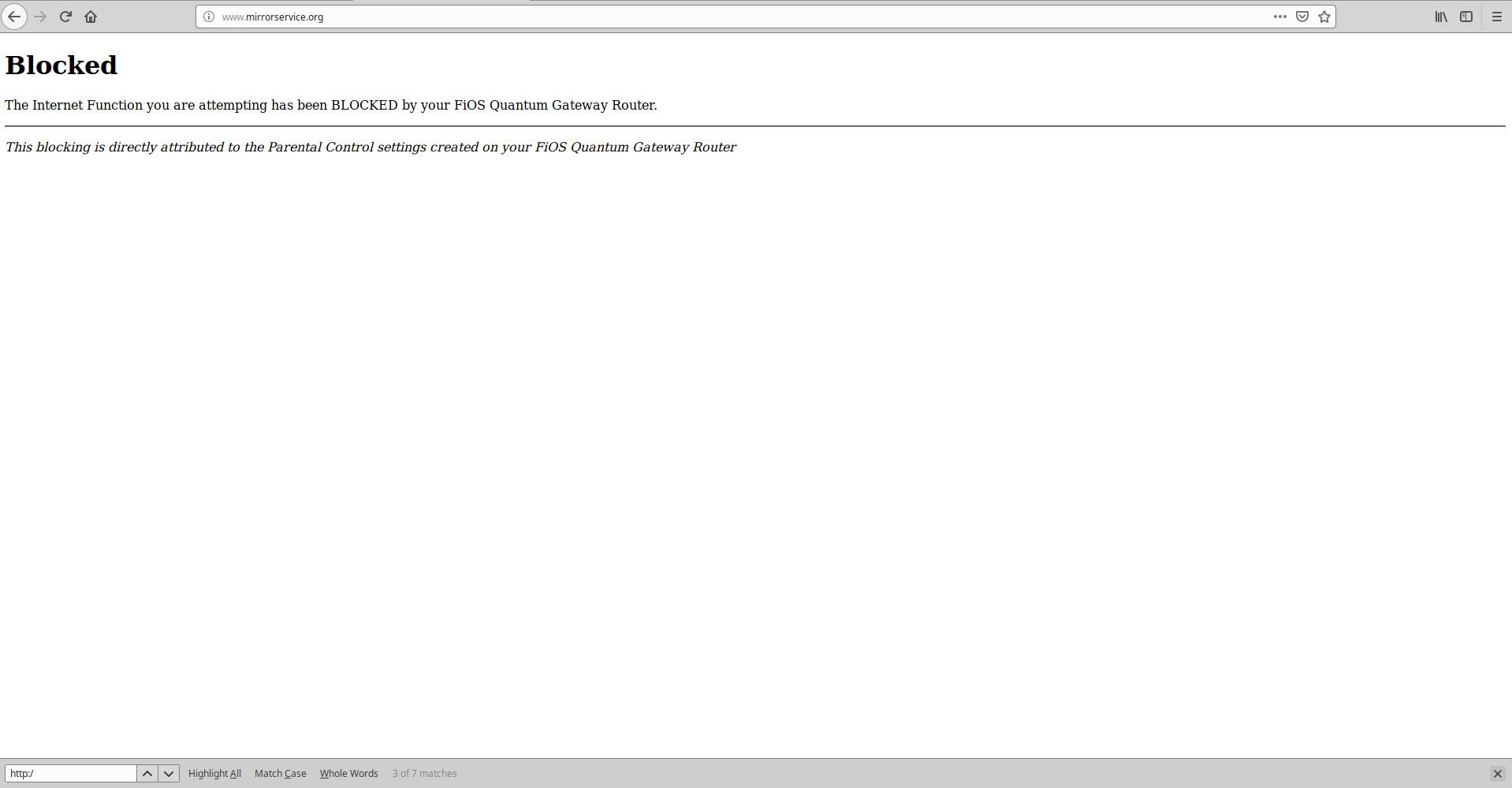


*Figure 4.12 “*[*www.yahoo.com*](http://www.yahoo.com)*” with the filter in Wireshark*

In figure 4.12, when put aside figure 4.11, there is an obvious note to make. Because of the filter coming into play, Wireshark starts sending out many more warnings that include frames with information such as “TCP ACKed unseen segment”, which means that the packet has acknowledged that data was not captured. The packet was transferred regularly, and the receiver has acknowledged it, but Wireshark itself was unable to find the packet in the capture. Sometimes, there are long TCP sessions running when the capture begins so that there are missing parts to the TCP session. Other reasons being for any of the ACKS to be duplicated or missing include switches, routers, firewalls, or client-side filtering.

* + 1. **Router:**

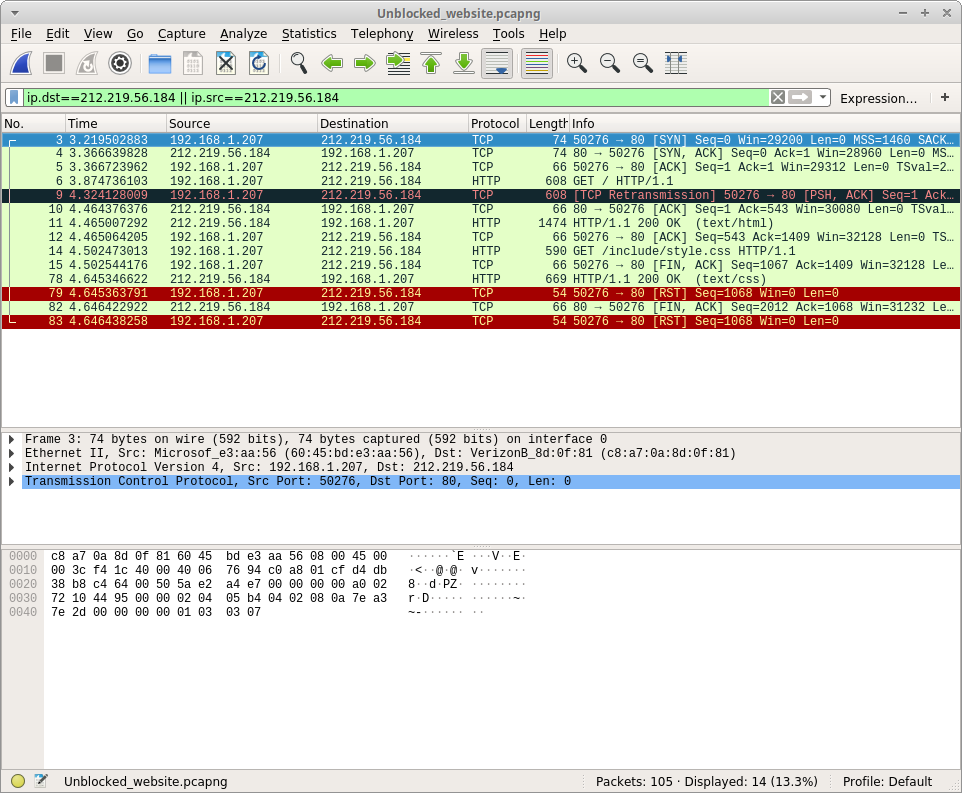
Using the router firewall, we get a slightly different prompt for the website, but a similar result, as seen in figure 4.13.



*Figure 4.13 Blocked Site*

We test the following scenario:

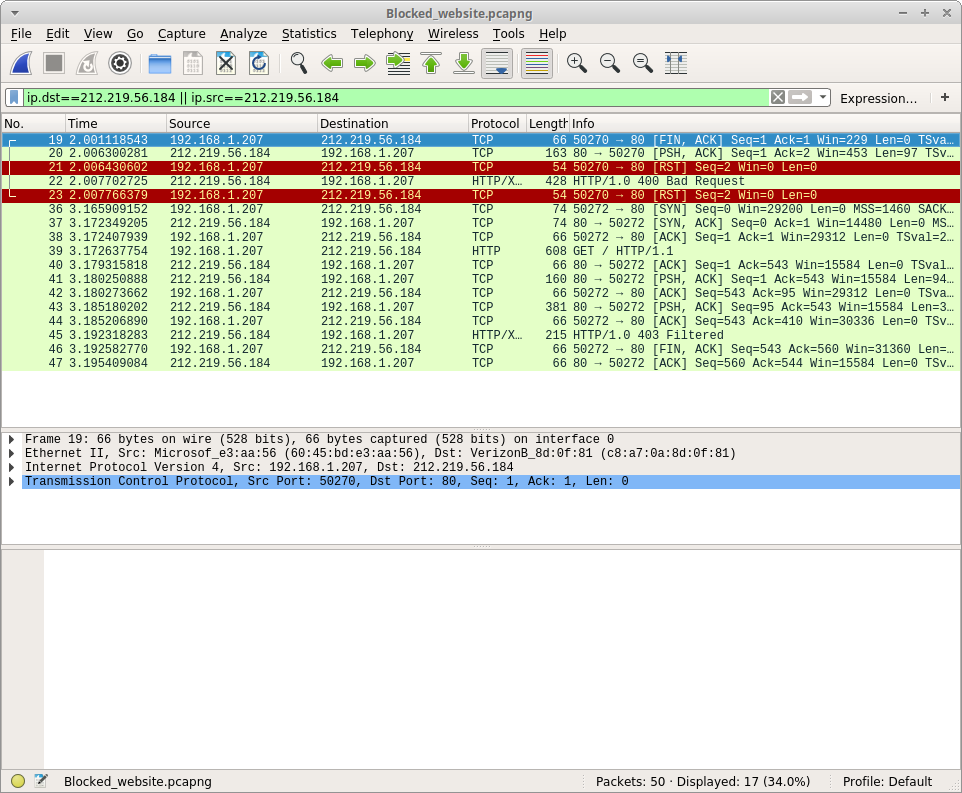
* + - 1. **Without filtering**



*Figure 4.14 Wireshark (Without the filter)*

In figure 4.14, we sorted our packet capture by displaying only packets with an ip address of 212.219.56.184 (The ip address of the website we are accessing). We can see that we are able to access the website with no errors.

* + - 1. **With filtering**



*Figure 4.15 Wireshark (With the filter)*

In figure 4.15, we can see now that there are more packets. Notably, packet number 45 explicitly says that the site has been filtered.

* 1. **Conclusion**

We can see that it is very simple to perform site filtering with both our router and the linux firewall. The difference between the two would be that if we wanted to make sure that nobody on our entire network could access the site, we would use our hardware firewall. However, if we just wanted to prevent access on a single, or a few computers, we would use our software firewall. However, on the router we used, we found that we were unable to filter any website that used HTTPS as opposed to the unsecured HTTP protocol. This is possibly due to the fact that our routers did not have very good firewall capabilities, and that a dedicated firewall would handle this better.

1. **Spyware**
   1. **Background:**

Spyware is usually an unwanted software whose function is to monitor a computer user’s activities, and then proceed to capture data about the user. [6]That data is then stored so that a third access it. It became the reason for more than half of Windows OS failures back in 2004.

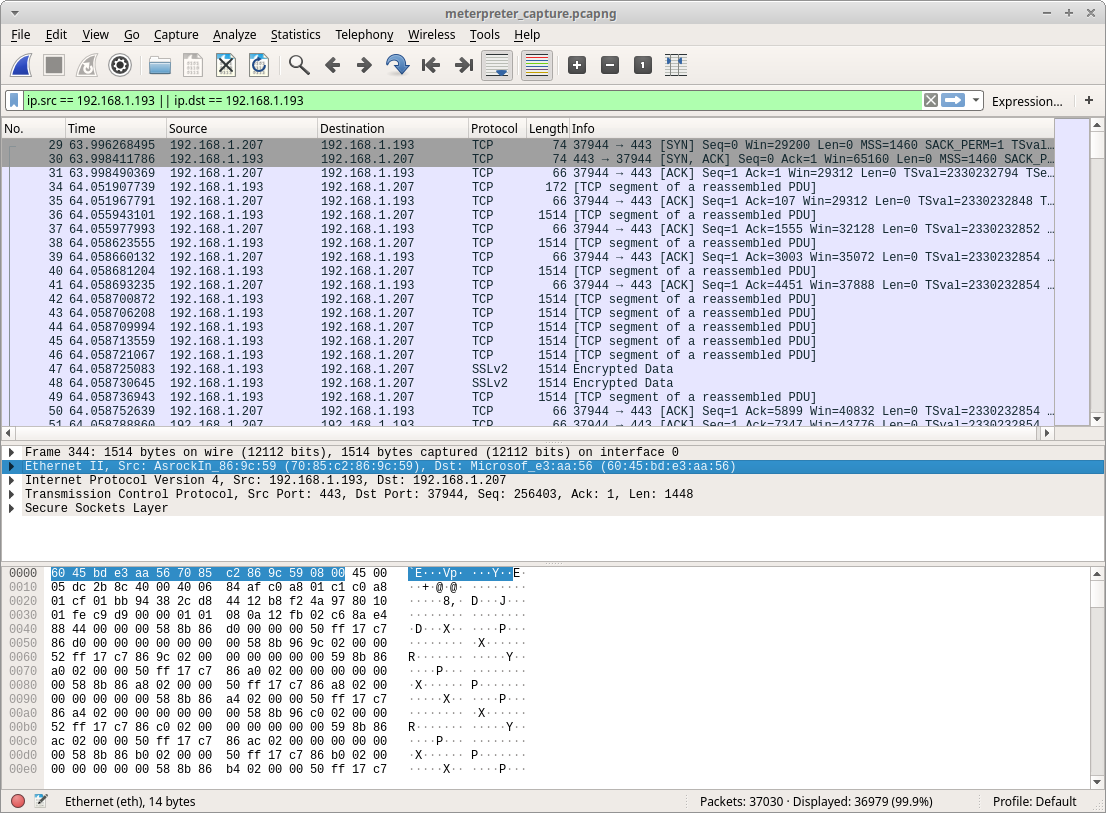
The severity of spyware can vary from mild to very dangerous. In cases of mild spyware, for example, a cookie. A cookie is a small file of information. It can allow a user to access a frequented site without having to reenter the username and password everytime, there poses little user risk. It’s generally useful to the user, who most likely consented to the sharing of his data with the site.

Actual spywares however, stem from associated cookies. They identify a user each time they interact with a member site, and then tracks activity and stores data gathered from interactions made by the user with the site. They are best known to be used to advertising companies.

Spyware can also be referred to as backdoor malware. Hackers will often use this to steal important information and uses a variety of techniques to hide their presence [11]. Researchers are continuously coming up with ways to detect their presence, so in realistic situations we can assume that even a firewall may be possibly bypassed.

* 1. **Methodology**
     1. **Setup**

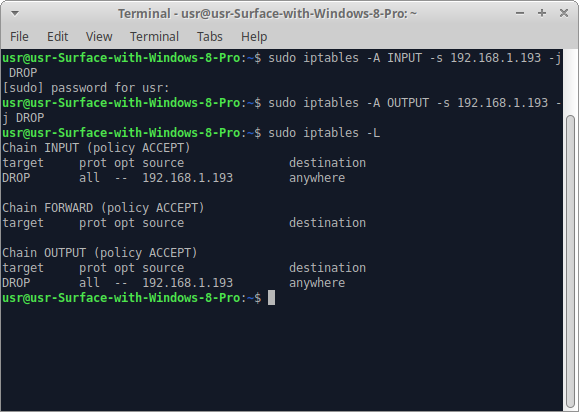
On the attacker’s computer, we have created an infected application using Metasploit’s “msfvenom” command. We then transferred the infected application to our victim via flash drive and ran the application. From here, we can access the meterpreter console on our attacker computer. In this experiment, we used the webcam recorder to access the victim’s webcam to spy on them. After running the webcam recorder, we opened wireshark to view the packet captures. From here, we see that there are a large number of packets being sent back and forth between 192.168.1.193 (The attacker’s computer) and 192.168.1.207 (The victim’s computer).



*Figure 4.16 Wireshark Setup*

* + 1. **Iptables**

Using iptables, we will try to stop this attack. Because we already know an attack is occuring and that this ip address is that of our attacker, we know to block this ip. We do this by running the commands:



*Figure 4.17 Iptables commands*

We then run the exploit again.

* + 1. **Router**

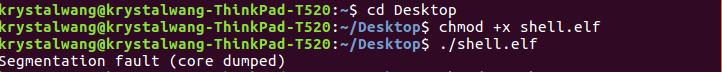
To set it up, we had the victim download a file called shell.elf, which was the spyware that we had set up. To block it, we required a router that would be able to block specific ips.

After downloading shell.elf, the victim would open terminal and execute it using the command:

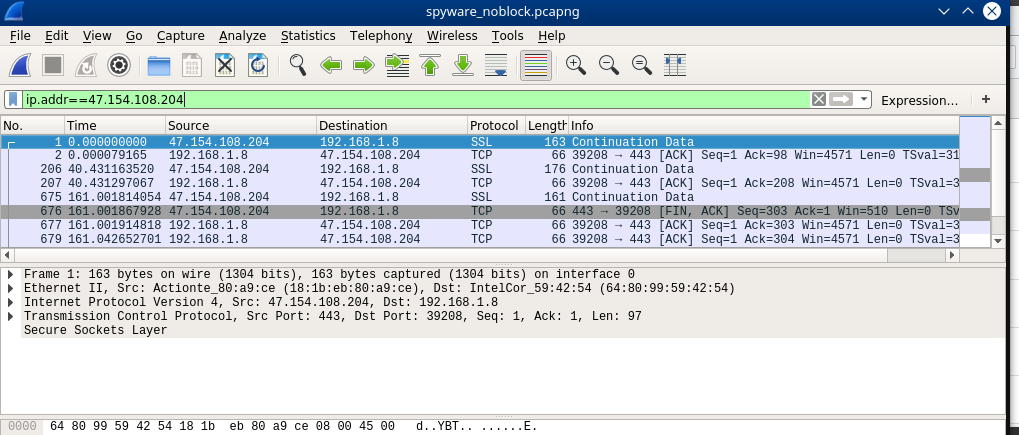
chmod +x shell.elf

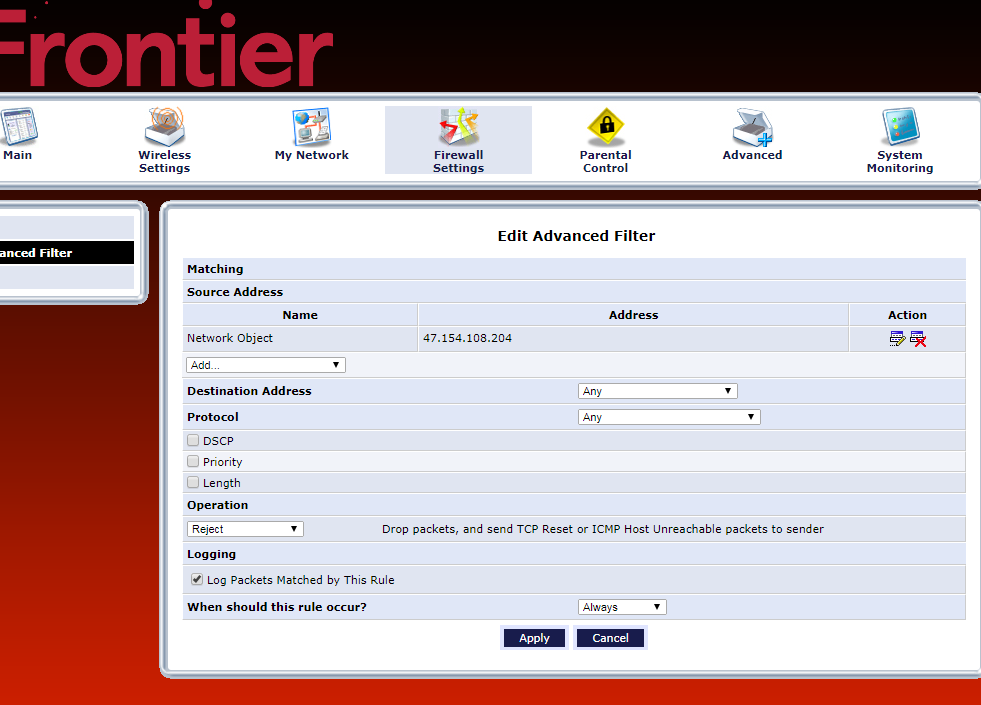
./shell elf

When that was complete, we ran wireshark to capture what was being done.



*Figure 4.18 Router-blocked spyware methodology*

**

**

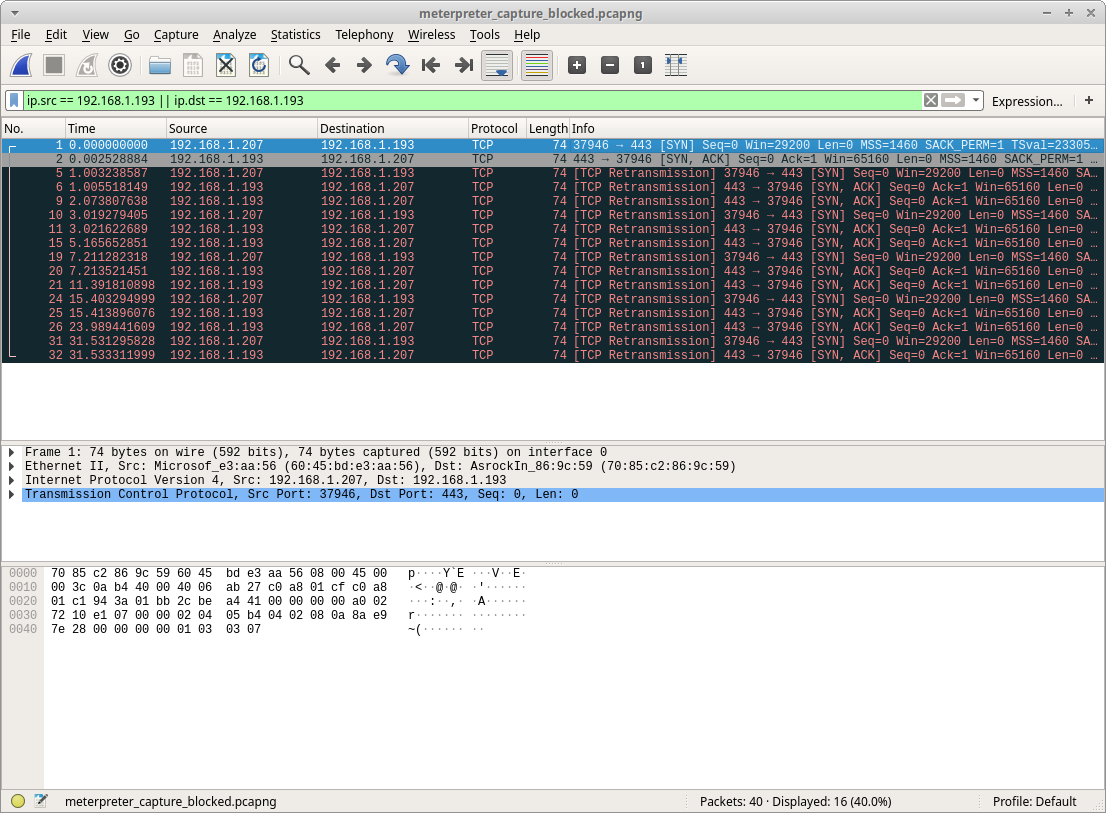
*Figure 4.19 Router Settings To Block IP Address*

* 1. **Results and Analysis**
     1. **Vulnerable machine:**

On a vulnerable computer, we were able to easily access the webcams and record the person using the computer.

* + 1. **Iptables**

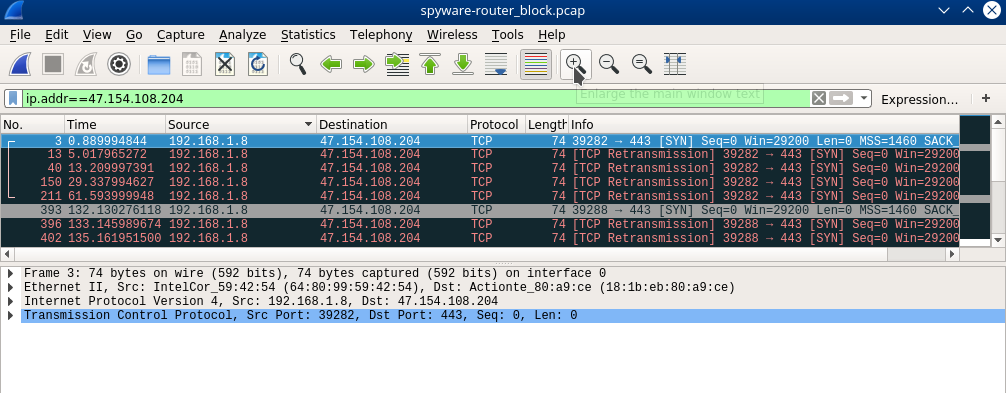
Using iptables, we can see that we can properly block the attack, seeing that the attacker can no longer connect to our victim’s computer. Looking at the packet capture, we can see that the only packet received/sent have a black background. This means that there was some sort of error with the packets. We can also see that there are far less packets with the firewall setup, as opposed to a vulnerable computer. Finally, the only packets are SYN and SYN,ACK packets, meaning no data is being transferred, unlike our firewall-less computer capture, which had a few packets with encrypted data.



*Figure 4.20 Iptables blocked*

* + 1. **Router**

Similar to our Iptables config, we can see that our router has blocked the attacker’s IP (47.154.108.204).



*Figure 4.21 Router-blocked spyware tables/packets*

* 1. **Conclusion**

After our computer has been infected with spyware, we can see that it is still possible to block the attacker’s ip address to prevent any data from transferring back to them. However, there are many problems with this approach. First, we would need to know that we are being spied on. Second, we would need to determine which ip address to block. In our case, our vulnerable computer had no software installed, and was not accessing the network frequently, meaning we receive a small amount of packets. In the real world, there may be a lot more packets being captured, making it harder to pinpoint which ip address is the attacker’s. Furthermore, we have almost no way of knowing whether or not we succeeded in blocking the attack. The attacker may also try to spoof their ip address, bypassing our filter. From this, we concluded that using a firewall to prevent, or mitigate the effects of spyware is unpractical, and that safe internet practices and anti-viruses may be a better solution.

**V. Challenges and Solutions**

* When we were setting up the spyware experiment, we tried to use the more popular SAMBA shares exploits. This exploit allowed us to infect a computer with just an ip address. (For our experiment, we needed our target to download and open an infected file) However, the exploit has since been patched. We then figured that some computers may still be running older, exploitable versions of samba, so we attempted to install an exploitable version of samba. Unfortunately, there were too many outdated dependencies that it was not possible for us to use this exploit. Instead, we created a trojan virus to use.
* We tried a variety of hardware firewalls, but found that some had very little security capabilities. For example, some of our router firewalls had no filtering capabilities. To fix this, we utilized various routers throughout our experiments.
* Many of our firewalls did not give us the option to block specific ip addresses. the Netgear router had the ability to, but a router firmware problem caused it to repeatedly send out a “400 Bad Request” error code and we were unable to use that.

**References**

|  |  |
| --- | --- |
| [1] | S. Mirzaie, A. K. Elyato and M. A. Sarram, "Preventing of SYN Flood Attack with Iptables Firewall," *2010 Second International Conference on Communication Software and Networks*, Singapore, 2010, pp. 532-535. |
| [2] | L. Xuan and P. Wu, "The Optimization and Implementation of Iptables Rules Set on Linux," *2015 2nd International Conference on Information Science and Control Engineering*, Shanghai, 2015, pp. 988-991. |
| [3] | T. M. Thang, C. Q. Nguyen and K. Van Nguyen, "Synflood Spoofed Source DDoS Attack Defense Based on Packet ID Anomaly Detection with Bloom Filter," *2018 5th Asian Conference on Defense Technology (ACDT)*, Hanoi, 2018, pp. 75-80. |
| [4] | 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. |
| [5] | T. M. Chen and V. Wang, "Web Filtering and Censoring," in *Computer*, vol. 43, no. 3, pp. 94-97, March 2010. |
| [6] | W. Ames, "Understanding spyware: risk and response," in *IT Professional*, vol. 6, no. 5, pp. 25-29, Sept.-Oct. 2004. |
| [7] | “What is a TCP SYN Flood | DDoS Attack Glossary | Imperva,” *Learning Center*. [Online]. Available: https://www.imperva.com/learn/application-security/syn-flood/. [Accessed: 04-Apr-2019]. |
| [8] | S. M. Bellovin and W. R. Cheswick, "Network firewalls," in *IEEE Communications Magazine*, vol. 32, no. 9, pp. 50-57, Sept. 1994. |
| [9] | Brackney, R: Cyber-Intrusion Response. Reliable Distributed Systems, 1998. Proceedings. Seventeenth IEEE Symposium on, 1998, Page(s): 413 –415. |
| [10] | Bace, Rebecca: An Introduction to Intrusion Detection & Assessment. Infidel Inc., prepared for ICSA Inc. Copyright 1998. |
| [11] | H. Loi and A. Olmsted, "Low-cost detection of backdoor malware," *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, Cambridge, 2017, pp. 197-198. |