|  |
| --- |
|  |
| Capstone Experience  IST 894 |
| Lab 4 – Firewall Configuration and Intrusion Detection |

Table of Contents

[1. General Context 3](#_Toc84110058)

[2. Technical Context 4](#_Toc84110059)

[3. Solution 5](#_Toc84110060)

[References 11](#_Toc84110061)

# General Context

Network security is a broad term that covers a multitude of technologies, devices, and processes. In it’s simplest term, it is a set of rules and configurations designed to protect the confidentiality, integrity, and availability of computer networks and data using both hardware and software technologies (What Is Network Security?, 2018). Two of the most important components of network security are firewalls, and intrusion detection systems. Both of these components come in various different shapes, sizes, configurations, and types.

Firewalls can be hardware appliances or software that is built into an operation system. There are several different types of firewalls, but the two most common are stateless, also called a packet filtering firewall, and stateful inspection. The main difference between stateful and stateless firewalls is that stateful firewalls maintain a state table that tracks all active connections between two end points, because it knows the source and destination IP’s and ports you only have to put a rule in once and the firewall is smart enough to allow a connection one way, and then allow the return traffic back through the other way. Stateless firewalls don’t care if a connection is active or in progress or not, it will compare each packet that passes through until the first rule matches and that is the action it will take, but because of this you have to put mirrored rules on opposite interfaces to allow traffic in and traffic out.

Intrusion detection systems (IDS) can also be hardware appliances or software packages that run on a server. There are Network IDS (NIDS) that monitor all traffic on a network, and there are Host IDS (HIDS) that run locally on a computer or server and monitor traffic locally on that system. IDSs can be signature based, anomaly based, or a hybrid of the two. Signature based IDS allow you to put in various known traffic patterns that items of interest are known to do, so you can only detect items that you already know exist, where anomaly based systems create a baseline of your network traffic and look for things that don’t match regular patterns.

In this lab we take a look at Snort, which is a signature-based NIDS. We look at some of the signature rules and patterns to see how they are written, and then inspect a packet capture to see how it actually works. Then we work with a stateful Linux host-based firewall called iptables.

# Technical Context

Snort is a signature-based network intrusion detection system (NIDS) that was created in 1998 by Martin Roesch. It was developed and managed by a company called Sourcefire until 2013 when Cisco acquired them. It’s actively developed and still widely used and highly relevant 23 years later. There are several different security company that actively develop signatures for Snort, some are open source and some are paid commercial products. The firewall that we use in this lab was also initially released in 1998, it’s called iptables and it’s the default host firewall on many Linux distributions.

We take a closer look at a signature with Snort ID number (SID) 648 (Snort - Rule Docs - Sid 1-648, n.d.) which watches network conversations for a pattern of NOP which is a machine language instruction to instruct the computer to do nothing, a lot of attackers use NOPs in buffer overflow attempts so this signature could be useful, but a lot of binary files use NOPs for timing purposes, to force memory alignment, to prevent hazards, or as a place-holder to be replaced by an active instruction at a later time to name a few so this signature might be useful, but it can also be quite noisy and fire on a lot of false positives (“NOP (Code),” 2021). Snort is able to replay packet captures (PCAP) against various different signatures to look for different things in different ways. We run an old PCAP against the base Snort ruleset and it generated over 27,000 different signature matches, some of these were relevant and important but many of them were false positives or just noise. This demonstrates the need to properly ‘tune’ an IDS for your network, meaning you know certain traffic types or patterns that are common and innocuous for your network so you don’t need to generate an alert when you see it.

When using a stateful inspection firewall, you have the ability to tell inbound or outbound interfaces to handle traffic differently based on if it’s a new connection, or if it’s pre-established. This is helpful for if you want to allow traffic one way, like HTTP(S) out, but you don’t want to allow it back in unless it’s an established connection that was created internally. This is especially important to stop attacks like reverse shells that otherwise would try to send specially crafted packets on specific, commonly used ports in an effort to bypass the firewall and gain access to a network that they otherwise would be prevented from doing.

# Solution

**We start this lab by taking a look at all of the various Snort rule files, as shown in Figure 1, that are available on our virtual machine. Specifically, we read the ‘shellcode.rules’ file to find the Snort ID number 648 which, as shown in Figure 2, has both a TCP and a UDP signature but are otherwise identical. To break down what the rule means, lets look at what it actually says. First, ‘alert’ is the action that Snort will take, which is to generate an alert using the selected alert method, and then log the packet. Next is the protocol, TCP or UDP. Then, it’s the source network/port -> destination network/port which are configured variables or ‘any’. After that, ‘msg’ is the text of the alert that is generated. Then, ‘content’** **the content keyword is one of the more important features of Snort. It allows the user to set rules that search for specific content in the packet payload and trigger response based on that data. Whenever a content option pattern match is performed, the Boyer-Moore pattern match function is called and the (rather computationally expensive) test is performed against the packet contents (3.5 Payload Detection Rule Options, n.d.). This content section utilizes the pipe ‘|’ character to surround the pattern which indicates that the data enclosed is binary data and represented as bytecode so this specific rule is looking for 14 NOPs. The last thing that we will discuss is the ‘depth’ keyword which allows the rule writer to specify how far into a packet Snort should search for the specified pattern. Depth modifies the previous `content' keyword in the rule. Depth of 128 tells Snort to only look for this specific pattern within the first 128 bytes of the payload. This rule, SID 648, can trigger often as false positives on a network where large binary files are transferred often as NOPs are commonly used for timing purposes, to force memory alignment, to prevent hazards, to occupy a branch delay slot, to render void an existing instruction such as a jump, as a target of an execute instruction, or as a place-holder to be replaced by active instructions later on in program development.**

**Graphical user interface, text

Description automatically generated**

Figure 1

Text

Description automatically generated

Figure 2

**Next, we run Snort on a specific packet capture (PCAP) to inspect the traffic, the command is seen in Figure 3. We’re told that an attacker was trying to steal a specific file, and in Figure 4 you can see that the attacker was attempting to exploit a directory traversal vulnerability to access ‘cheddar.pdf’. However, while the attempted directory traversal was a legitimate alert, there were thousands of other alerts that were nothing but noise. Figure 5 shows some of these noisy alerts. The alerts here are false positives as it shows typical DHCP traffic. A client doesn’t know the address of a DHCP server so it floods a DHCP discover message to a broadcast IP address of 255.255.255.255 with a source port of 68 and a destination port of 67. Since the purpose of sending a DHCP discovery message is to obtain an IP address, the sending endpoint doesn’t have an IP address yet so it’s shown as 0.0.0.0. Since the time that this virtual machine was configured, this particular rule was deprecated and moved to the ‘deleted’ category (Snort - Rule Docs - Sid 1-527, n.d.). Rules are never totally removed from the ruleset, they are moved to the deleted category.**

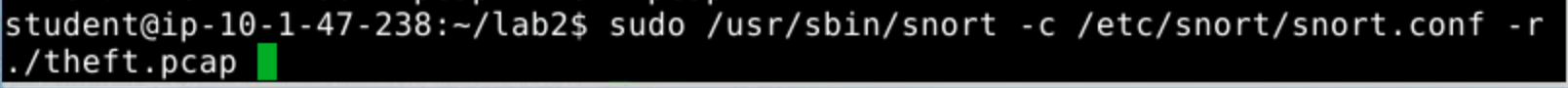
****

Figure 3

Table

Description automatically generated

Figure 4

A picture containing text

Description automatically generated

Figure 5

**For the next part of the lab, we focus on iptables. We run `sudo iptables -L -n`, shown in Figure 6, which prints out the list of active firewall rules on the machine. The output is split into three chains: INPUT, FORWARD, and OUTPUT. Each chain has a default policy, which means what will happen if traffic doesn’t match any of the configured rules. In the Figure 6 example, INPUT has a default policy of DROP, while FORWARD, and OUTPUT both have a default policy of ACCEPT. On the INPUT chain, the first rule allows all ICMP traffic, the second rule allows all UDP traffic, the third rule allows TCP traffic on port 22 (SSH) for new and established connections, the forth rule allows TCP traffic on 3389 (RDP) for new and established connections, and finally the last rule allows all traffic from anywhere to anywhere on any protocol and any port. The OUTPUT chain accepts TCP traffic to ports 22 and 3389 in the first two rules, but only for established connections, however the third rule allows all traffic from anywhere to anywhere on any protocol and any port. The take-aways here are that Secure Shell (SSH) and Remote Desktop Protocol (RDP) are both remote access tools these rules are meant to allow outbound connections to be created or maintained, but only pre-existing connections are allowed back in. However, both chains have an allow any rule at the bottom which negates the need for any of the above accept rules. There is an error in these rules that doesn’t present itself because of the catch-all rule at the bottom, this error is on the OUTPUT chain and that 22 and 3389 should be set as the source ports not the destination ports. If the default policy on OUTPUT was DROP and the ACCEPT all wasn’t under it, inbound SSH connections wouldn’t be able to connect.**

A screenshot of a computer

Description automatically generated with medium confidence

Figure 6

**In our last step for this lab, we create some basic rules in the INPUT and OUTPUT chains to allow outbound connections on ports 80 and 443 since most websites now require you to connect via HTTPS. Figure 7 shows the rules for each 80 and 443 that allow inbound TCP traffic on a source port of 80/443 for established connections, but will allow new and established connections outbound on destinate port of 80/443. Finally, since I don’t want to allow connections inbound to an internal HTTP(S) server, I put in rules to drop all traffic coming in with a destination port of 80/443 and outbound traffic with a source port of 80/443. Figure 8 shows these new rules in the active ruleset, Figure 9 shows that we are still able to access cnn.com, and Figure 10 shows the output of `iptables -vL -n` which is a verbose output which shows the traffic stats for each rule and shows that my new rules are being triggered.**

A screen shot of a computer

Description automatically generated with medium confidence

Figure 7

A picture containing diagram

Description automatically generated

Figure 8

Graphical user interface, text

Description automatically generated

Figure 9

Calendar

Description automatically generated

Figure 10

# ****References****

*3.5 Payload Detection Rule Options*. (n.d.). Retrieved October 2, 2021, from <http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node32.html>

NOP (code). (2021). In *Wikipedia*. <https://en.wikipedia.org/w/index.php?title=NOP_(code)&oldid=1045613275>

*Snort—Network Intrusion Detection & Prevention System*. (n.d.). Retrieved October 2, 2021, from <https://www.snort.org/rules_explanation>

*Snort—Rule Docs—Sid 1-527*. (n.d.). Retrieved October 2, 2021, from <https://www.snort.org/rule_docs/1-527>

*Snort—Rule Docs—Sid 1-648*. (n.d.). Retrieved October 2, 2021, from <https://www.snort.org/rule_docs/1-648>

*Understanding the Basic Operations of DHCP*. (n.d.). Network Manias. Retrieved October 2, 2021, from <https://www.netmanias.com/en/?m=view&id=techdocs&no=5998>

*What is Network Security?* (2018, August 9). Forcepoint. <https://www.forcepoint.com/cyber-edu/network-security>