Network Management Security Tool Report: Snort

Miles Brown, Zane Sampey, Marcus Simmons
CNT4403
Nov 28, 2023

Abstract Summary

Three students collaborated to gain proficiency in utilizing a Network Security Management tool (NSM). The primary goal was to investigate and assess the tool's capabilities through the creation of a virtual network topology within VMWare. This involved simulating data generation within the network and subsequently employing a selected tool for comprehensive analysis. The project employed Ubuntu Linux as the operating system, Docker for network topology configuration, and specifically leveraged Snort as the NSM tool of choice.

Detail Description of tool

Snort is a free Open Source Intrusion Prevention and Intrusion Detection System (IPS/IDS) that monitors network traffic and identifies potentially malicious activities on Internet Protocol (IP) networks. It was developed in 1998 by Martin Roesch. Now it is being developed by Cisco. Snort's IPS uses a series of rules that help define malicious network activity and uses those rules to find packets that match against them and generates alerts for users.
Snort is used on the command line and can be configured to run in three modes:

- **Sniffer mode**, which simply reads the packets off of the network and displays them for you in a continuous stream on the console (screen).
- **Packet Logger mode**, which logs the packets to disk.
- **Network Intrusion Detection System (NIDS) mode**, which performs detection and analysis on network traffic. This is the most complex and configurable mode.

Below are the features provided by Snort:

Real-time traffic monitoring - Snort provides real-time network traffic monitoring capabilities allowing users to detect malicious activity sooner. Real-time alerting capabilities are enabled with the help of third-party programs such as swatch (Simple Watcher) or syslog-ng (syslog-next generation).

Packet logging - Snort can capture packets and log the traffic to a log directory on a disk. You can configure Snort to use a variety of output plug-ins, allowing you to log data as ASCII text files (JSON, CSV, unified2), binary files, databases, and more. Add-on products such as snort_stat, SnortALog, Swatch, and Barnyard analyze the logs and alerts in easy-to-use formats. Logging is useful for traffic monitoring and auditing. With saved log files, network administrators can analyze the files to understand the network traffic. For instance, an attack may be analyzed in a log file to give a better understanding of how it worked for future prevention.

Analysis of protocol - Snort allows you to capture packets and match specific protocol related data. Protocol analysis is a network sniffing technique that collects data in protocol layers for further analysis. This allows the network administrator to inspect possibly harmful data packets in more detail.

Content matching - Snort categorizes rules by protocol (IP, TCP, UDP, ICMP), then by ports, and finally by those with content and those without content. For rules with

content, selecting rules using a multi-pattern matching scheme was found to have increased performance, especially when applied to large rule groups related to HTTP.

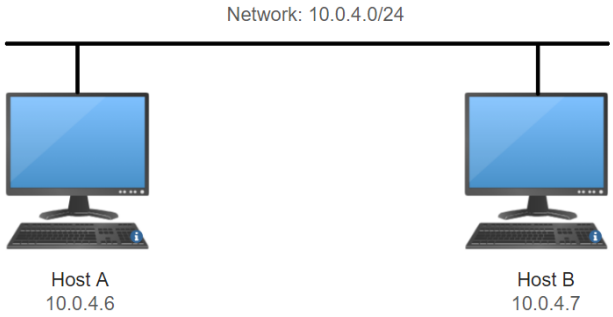
OS fingerprinting - Snort can determine the operating systems used by particular hosts based on the packet's header format, on the premise that all systems have a unique TCP/IP stack.

Compatibility - Snort can be easily installed into any IP network environment on OSs such as Windows and Linux.

Open Source - Snort is accessible for free and open source, so users can use the software as they please.

Rules are easy to implement - Snort provides an easy to implement signature-based rule system that allows you to define attack-signatures to defend a network.

Architecture and Methodology



The team set up a basic virtual network using two VMs, HostA and HostB. The network that both machines were on was 10.0.4.0/24 each assigned IP addresses 10.0.4.6 and 10.0.4.7 respectively.
The team also set up a similar virtual network using docker containers. This network was defined to be 10.0.0.0/24. The two machines that were a part of the network were assigned IP addresses 10.0.0.1 and 10.0.0.5.
With both architectures, one machine was used as a client to establish connections with the other machine, which acted as a server. The machine acting as a server ran Snort, logging the packets sent to it by the client machine.

Demonstration and Description

To test the NSM tool, we simulated TCP and spoofing attacks on the victim machine. We created custom rules within the local.rules file to test the different actions (alert, block, drop, log, pass) and detect or mitigate potentially malicious traffic. We customized the configuration file used by snort (snort.conf) to enable the rules in the local.rules file, as well as configured different output modules.
First, the team used Snort in sniffer mode to trace the TCP protocol's 3-way handshake. The 3-way handshake is used to establish a connection between two telnet machines. First, the handshake is initiated when the client sends a SYN packet to the TCP server. Then, the server responds with a SYN + ACK packet. Finally, the client sends the final ACK packet which establishes the connection.
Primarily, the team used Snort as a packet logger to log different types of network traffic by writing custom rules. Being able to create rules to target certain packets based on protocols and flags is useful for network administrators to monitor the right traffic. Using different rule options allowed us to specify which TCP flags a rule will alert on, and which direction of the flow of packets should Snort capture. Along with logging TCP packets, the team was able to create rules that logged ICMP and UDP packets. Then, the team wrote rules to detect a TCP SYN flood attack.This attack has to do with

exhausting a victim's TCP cache with half-open connections, thus preventing any other TCP connections from being established. Using different rule options we were able to alert on packets based on the SYN flag, and based on thresholds that alert on events like capturing 100 SYN packets in less than 60 seconds.
Some example rules:

alert tcp any any -> 10.9.0.5 23 (msg:"Caught 10 SYN packets in less than 10 seconds!"; threshold: type limit, track by_dst, count 10, seconds 10; priority: 1000000; flags:S; flow: to_server,not_established; sid: 1000001;)

alert tcp any any -> 10.9.0.5 23 (msg:"Caught 50 SYN packets in less than 50 seconds!"; threshold: type limit, track by_dst, count 50, seconds 50; priority: 100; flags:S; flow: to_server,not_established; sid: 1000003;)

alert tcp any any -> 10.9.0.5 23 (msg:"Caught 100 SYN packets in less than 100 seconds!"; threshold: type limit, track by_dst, count 100, seconds 60; priority: 1; flags:S; flow: to_server,not_established; sid: 1000004;)

log tcp any any -> 10.9.0.5 23 (msg:"Caught 100 SYN packets in less than 100 seconds! Blocking."; threshold: type limit, track by_dst, count 100, seconds 60; priority: 1; flags:S; flow: to_server,not_established; sid: 1000005;)

Demonstration Images

```
root@VM:/home/seed# snort -q -l /var/log/snort -i ens34 -A console -c
/etc/snort/snort.conf
11/20-21:21:21.582018  [**] [1:10001:0] Echo Request Detected [**] [Pr
iority: 0] {ICMP} 10.0.4.7 -> 10.0.4.6
11/20-21:21:21.582081  [**] [1:10002:0] Echo Reply Detected [**] [Prio
rity: 0] {ICMP} 10.0.4.6 -> 10.0.4.7
11/20-21:21:29.494779  [**] [1:20001:0] SYN packet sent to server for
telnet session [**] [Priority: 0] {TCP} 10.0.4.7:40822 -> 10.0.4.6:23
11/20-21:21:56.976352  [**] [1:30001:0] UDP connection to server [**]
[Priority: 0] {UDP} 10.0.4.7:53493 -> 10.0.4.6:4444
```

The image above displays the alerts sent to the console while running snort with the group's implemented rules.

```
[11/20/23]seed@VM:.../snort$ sudo tcpdump -r snort.log.1700534313
reading from file snort.log.1700534313, link-type EN10MB (Ethernet)
21:38:39.603928 IP 10.0.4.7 > VM: ICMP echo request, id 5, seq 1, length 64
21:38:39.603985 IP VM > 10.0.4.7: ICMP echo reply, id 5, seq 1, length 64
21:38:44.257179 IP 10.0.4.7.40824 > VM.telnet: Flags [S], seq 388396007, win 6
4240, options [mss 1460,sackOK,TS val 1725892369 ecr 0,nop,wscale 7], length 0
21:39:14.333406 IP 10.0.4.7.47525 > VM.4444: UDP, length 6
```

The image above displays the contents of the log files generated by snort after completing the demonstration.

```
[**] [1:1000004:0] Caught 100 SYN packets in less than 100 seconds! [**]
[Priority: 1]
11/25-17:13:22.547572 20.15.204.29:3104 -> 10.9.0.5:23
TCP TTL:50 TOS:0x0 ID:39935 IpLen:20 DgmLen:40
******S* Seq: 0x3BDE3965  Ack: 0x0  Win: 0x4E20  TcpLen: 20

[**] [1:1000003:0] Caught 50 SYN packets in less than 50 seconds! [**]
[Priority: 100]
11/25-17:13:22.547572 20.15.204.29:3104 -> 10.9.0.5:23
TCP TTL:50 TOS:0x0 ID:39935 IpLen:20 DgmLen:40
******S* Seq: 0x3BDE3965  Ack: 0x0  Win: 0x4E20  TcpLen: 20

[**] [1:1000001:0] Caught 10 SYN packets in less than 10 seconds! [**]
[Priority: 1000000]
11/25-17:13:22.547572 20.15.204.29:3104 -> 10.9.0.5:23
TCP TTL:50 TOS:0x0 ID:39935 IpLen:20 DgmLen:40
******S* Seq: 0x3BDE3965  Ack: 0x0  Win: 0x4E20  TcpLen: 20

[**] [1:1000004:0] Caught 100 SYN packets in less than 100 seconds! [**]
[Priority: 1]
11/25-17:13:22.562076 23.241.133.11:388 -> 10.9.0.5:23
TCP TTL:50 TOS:0x0 ID:57470 IpLen:20 DgmLen:40
******S* Seq: 0xF3335047  Ack: 0x0  Win: 0x4E20  TcpLen: 20

[**] [1:1000003:0] Caught 50 SYN packets in less than 50 seconds! [**]
[Priority: 100]
11/25-17:13:22.562076 23.241.133.11:388 -> 10.9.0.5:23
TCP TTL:50 TOS:0x0 ID:57470 IpLen:20 DgmLen:40
******S* Seq: 0xF3335047  Ack: 0x0  Win: 0x4E20  TcpLen: 20

[**] [1:1000001:0] Caught 10 SYN packets in less than 10 seconds! [**]
[Priority: 1000000]
11/25-17:13:22.562076 23.241.133.11:388 -> 10.9.0.5:23
TCP TTL:50 TOS:0x0 ID:57470 IpLen:20 DgmLen:40
******S* Seq: 0xF3335047  Ack: 0x0  Win: 0x4E20  TcpLen: 20
```

The image above displays the syn flood alerts logged to the alert.full file, which shows the alert message, priority, timestamp, and tcp header.

References

Cisco Talos Detection Response Team. *Snort 3 rule writing guide*. [Online] Available at: https://docs.snort.org

Snort 3 User Manual. Amazon Web Services. [Online] Available at: <snort-org-site.s3.amazonaws.com>.

Snort IDS/IPS Explained. What - Why you need - How it works. zenarmor.com. [Online] Available at: <zenarmor.com>.

Real-Time Alerting with Snort. LinuxSecurity.com. [Online] Available at: <linuxsecurity.com>.

Working with Attack Signatures. f5.com. [Online] Available at: <f5.com>.

Snortalog - Free Software Directory. Free Software Foundation. [Online] Available at: <fsf.org>.

Snort | NXLog Docs. [Online] Available at: <nxlog.co>.

Installing and Configuring Swatch. Managing Snort Sensors. flylib.com. [Online] Available at: <flylib.com>.

Contributions

Marcus Simmons: Conducted research on NSM tool; Tested tool in a docker setup; Documented results and findings about the tool and reported it back to the team.

Zane Sampey: Assisted in setting up the virtual subnet needed for the demonstration; Developed rules and configured snort for the demonstration; Added screenshots related to the demonstration to the group presentation; Proofread the report

Miles: Designed network topology; Researched Snort to understand its installation and functionality; Tested the tool with custom rules in a VM environment; Designed and contributed to the slides;