

# IDS Evasion Techniques - SNORT

Ritwik Gupta  
School of Informatics and Cybersecurity  
Technological University Dublin,  
Dublin, Ireland  
b00168210@mytudublin.ie

**Abstract**—With the inevitable rise of cyberattacks, specifically those against critical infrastructure, there is a dire need for organizations to build finer suspicious traffic detection and prevention systems which allow cybersecurity analysts to monitor attack intrusion attempts and evade potential security incidents. This report delves into the deployment of open-source Intrusion Detection and Prevention systems (IDS/IPS) such as Snort to understand the internal operations of such cyber defense systems. In addition, research and implement various approaches to evade these IDS defenses to understand the threat actor's point-of-view. Further, train the IDS to detect such evasion techniques to build efficient defense mechanisms to prevent future attacks. This research gives a fair knowledge of open-source IDS i.e. Snort, about its purpose, modes of operations, its implementations and the applications.

**Keywords**—Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS), evasion techniques, Snort.

## I. INTRODUCTION

The struggle between attackers and defenders in the cyber world is an ongoing challenge. Attackers are constantly seeking to infiltrate systems for various motives, whether it's for political, financial, or in recent times, cyber warfare. As a result, it is crucial for cyber systems to be safeguarded against potential cyber-attacks and breaches. To address this issue, the development of intrusion prevention and intrusion detection systems has been pivotal. These systems utilize advanced technology to identify, prevent, and report cyber-attacks on network and telecommunication infrastructure. By analysing the entire network traffic and content, they provide a valuable advantage in protecting and defending against attackers from a centralized location.

The following sections discuss the functionalities and implementations of IDS, demonstration of IDS evasion techniques leveraging several tools and scripts used in this research to generate the attacks, evasions, and inspecting the evasions.

## II. BACKGROUND

### A. Intrusion Detection System (IDS)

An Intrusion Detection System is a collection of tools and systems designed to constantly survey and analyze the Network Traffic for any signs of unauthorized activity. If any anomaly is detected, the system promptly triggers alerts to mitigate any potential breaches. The network traffic can be mainly categorized into Benign traffic (regular, expected traffic that does not alert the IDS constantly) and Malicious traffic (abnormal traffic that would not typically occur under normal circumstances). The IDS needs to be able to detect and identify this type of traffic.

The different types of IDS can be classified as below:

- **Signature-based IDS:** While using Signature-based IDS, it is necessary to provide the system with specific information about malicious or unwanted

traffic. This means explicitly programming the detection engine with details of signatures of malware.

- **Anomaly-based IDS:** It majorly relies on the understanding of what regular traffic patterns look like. This involves "training" the system about what is considered normal so that it can identify abnormalities. This can be achieved through techniques such as machine learning or manual rules.

### B. Types of Open Source IDS

The top five leading open-source intrusion detection systems available in the market today are:

- **Snort:** Snort is one of the oldest IDS in the market, with command line interface, allows a high level of customization due to which many organizations choose it. Snort is available for Linux, Windows, Fedora, Centos, and FreeBSD.
- **Zeek:** An exceptional tool for monitoring networks and analysing traffic patterns, utilizes domain-specific language that does not depend on conventional signatures. It is compatible with Unix, Linux, Free BSD, and Mac OS X.
- **OSSEC:** A host-based IDS system, offers protection through log analysis, file integrity monitoring, Windows registry monitoring, centralized policy enforcement, rootkit detection, real-time alerting, and active response. It runs on a wide range of operating systems, from Linux and OpenBSD to FreeBSD, MacOS, Solaris, and Windows.
- **Suricata:** A powerful network threat detection engine that stands out for its ability to swiftly identify and respond to potential intrusions. Key strength is its multi-threaded design, Suricata boasts built-in hardware acceleration technology, utilizing the potential of graphic cards to thoroughly inspect network traffic.
- **Security Onion:** A dynamic open-source tool that offers advanced capabilities for threat hunting, intrusion detection, enterprise security monitoring, and log management. What sets it apart is its powerful combination of other security tools such as Snort, Kibana, Zeek, Wazuh, CyberChef, NetworkMiner, Suricata, and Logstash.

### C. IDS Evasion Techniques

Despite their overall reliability, IPS/IDS systems may still leave room for attackers to evade and infiltrate end systems. This means that if an attacker manages to bypass the IPS/IDS, they could easily reach their intended target. Though these systems continue to be regularly strengthened against such attacks, there are still techniques that can successfully bypass them.

Some of the best-known evasion techniques applicable to IPS/IDS system are:

### Tactical Denial of Service (DoS)

As the number of rules and network traffic volume expand, a strong processing power is essential for an IDS/IPS to effectively operate. This is even more critical for IDS in particular, as its primary response is to log any traffic that matches a specified signature. Therefore, it can be advantageous to experiment with the following strategies:

- Generate a significant volume of benign traffic that overwhelms the processing ability of the IDS/IPS.
- Produce a vast amount of non-malicious traffic that still triggers logging and potentially overwhelms the communication channel with the logging server or exceeds its disk writing capabilities.

### Payload Manipulation

Evasion via payload manipulation includes: Encrypting the communication channel, Encoding and Obfuscating the payload. One potential vulnerability of using an IDS/IPS is that it cannot analyse encrypted data, making it easier for attackers to avoid being detected by using encryption. In addition, payload encoding and obfuscation are another evasion techniques.

### IP Packet Fragmentation (Session Splicing)

The tactic of fragmenting attack packets with modified MTU values is a common evasion technique used by attackers. By adjusting the MTU value at the IP layer, the packets are divided and sent in smaller fragments. This complicates the task of properly reassembling the packet for the IPS/IDS correlation engine, potentially allowing an attack to slip through undetected. This is made even more effective when the MTU value is lowered, as it forces the IPS/IDS to handle a larger number of smaller packets. While the IPS/IDS may have an adjustable buffer, it is not always optimized to handle such a high volume of packets, leaving it vulnerable to DOS attacks.

### Tampering Source Port Number

The practice of modifying port numbers, which is a technique used at layer 4, is employed by attackers to bypass IPS/IDS detection. Typically, the UDP and TCP, source and destination ports are inspected by IDS rules, and without deep packet inspection, the port numbers are considered the primary indicator of the service being used. If a threat actor changes the port used for transmitting data, it can also prevent the IPS/IDS from generating any alerts.

## III. DEPLOYMENT

This section discusses the deployment, configuration, and implementation phases of Snort IDS for the assignment.

### A. Snort Installation in Ubuntu 22.04

For setting up the Intrusion detection system i.e. Snort, Ubuntu 22.04.3 LTS (Jammy Jellyfish) was selected due to its widespread use and popularity which means that Snort packages are readily available and well-maintained for this platform. In addition, Ubuntu's flexibility to customize and configure their systems is vital for Snort, as it allows necessary rule customization and tuning based on the network environment.

Debian-based Linux operating systems (OSes) like Ubuntu comes with a pre-installed packages manager "apt-get" utilized for installing, updating, and removing software packages. In our case, we leverage "apt-get" package manager to install Snort,

```
root@ritwik-host-vm:~# sudo apt-get install snort -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
snort is already the newest version (2.9.15.1-6build1).
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
```

Fig. 1. Snort Installation

Next, the directory structure of Snort includes the following files, with 'snort.conf' configuration file of utmost importance. 'snort.conf' is crucial for configuring Snort's working, specifying rules, and defining network settings, encapsulating the system's detection and prevention parameters in a concise, editable configuration file. In addition, all the Snort rules including local and community rules are stored in 'rules' directory. These rules allows Snort to function as IDS or IPS and detect or prevent any incoming attacks.

```
root@ritwik-host-vm:~# cd /etc/snort/
root@ritwik-host-vm:/etc/snort# ls -al
total 376
drwxr-xr-x  3 root root  4096 Dec 11 19:13 .
drwxr-xr-x 129 root root 12288 Dec 13 19:04 ..
-rw-r--r--  1 root root  1281 Dec  3 2019 attribute_table.dtd
-rw-r--r--  1 root root  3757 Dec  3 2019 classification.config
-rw-r--r--  1 root root 82469 Dec  3 2021 community-sid-msg.map
-rw-r--r--  1 root root 23657 Dec  3 2019 file_magic.conf
-rw-r--r--  1 root root 32789 Dec  3 2019 gen-msg.map
-rw-r--r--  1 root root   687 Dec  3 2019 reference.config
drwxr-xr-x  2 root root  4096 Dec 11 20:28 rules
-rw-r--r--  1 root root 29907 Dec 11 19:13 snort.conf
-rw-r--r--  1 root root   807 Dec  9 16:03 snort.debian.conf
-rw-r--r--  1 root root  2335 Dec  3 2019 threshold.conf
-rw-r--r--  1 root root 160606 Dec  3 2019 unicode.map
root@ritwik-host-vm:/etc/snort# sudo vim snort.conf
```

Fig. 2. Snort directory structure with 'snort.conf' file

### B. Configuration and Implementation

We leverage Vim editor to modify/write the configuration file with appropriate parameters.

```
root@ritwik-host-vm:/etc/snort
1 #
2 # VRT Rule Packages Snort.conf
3 #
4 # For more information visit us at:
5 # http://www.snort.org Snort Website
6 # http://vrt-blog.snort.org SourceFire VRT Blog
7 #
8 # Mailing list Contact: snort-users@lists.snort.org
9 # False Positive reports: fpg@sourcefire.com
10 # Snort bugs: bugs@snort.org
11 #
12 # Compatible with Snort Versions:
13 # VERSIONS : 2.9.15.1
14 #
15 # Snort build options:
16 # OPTIONS : --enable-gre --enable-mpls --enable-targetbased --enable-ppm --enable-perfprofiling
17 # e-response --enable-normalizer --enable-reload --enable-react --enable-flexresp3
18 #
19 # Additional information:
20 # This configuration file enables active response, to run snort in
21 # test mode -T you are required to supply an interface -I <interface>
22 # or test mode will fail to fully validate the configuration and
23 # exit with a FATAL error
24 #
25 #####
26 # This file contains a sample snort configuration.
27 # You should take the following steps to create your own custom configuration:
28 #
29 # 1) Set the network variables.
30 # 2) Configure the decoder
31 # 3) Configure the base detection engine
32 # 4) Configure dynamic loaded libraries
33 # 5) Configure preprocessors
34 # 6) Configure output plugins
35 # 7) Customize your rule set
36 # 8) Customize preprocessor and decoder rule set
37 # 9) Customize shared object rule set
38 #####
```

Fig. 3. 'snort.conf' file via Vim editor

In order to configure the Snort to detect traffic within the network, we modify the 'snort.conf' file with internal network IP address range.

```

58 # Setup the network addresses you are protecting
59 #
60 # Note to Debian users: this value is overridden when starting
61 # up the Snort daemon through the init.d script by the
62 # value of DEBIAN_SNORT_HOME_NET s defined in the
63 # /etc/snort/snort.debian.conf configuration file
64 #
65 # ipvar HOME_NET 192.168.72.0/24
66 #
67 # Set up the external network addresses. Leave as "any" in most situations
68 # ipvar EXTERNAL_NET any
69 # If HOME_NET is defined as something other than "any", alternative, you can
70 # use this definition if you do not want to detect attacks from your internal

```

Fig. 4. Network parameters in ‘snort.conf’ file

The Snort rules can be specified in ‘\*.rules’ files categorised into local and community rules. Here, community rules are pre-defined ruleset maintained by Snort development community itself, whereas, local rules are created by security analysts or users for test purposes.

```

#####
# Step #7: Customize your rule set
# For more information, see Snort Manual, Writing Snort Rules
#
# NOTE: All categories are enabled in this conf file
#####
# Note to Debian users: The rules preinstalled in the system
# can be *very* out of date. For more information please read
# the /usr/share/doc/snort-rules-default/README.Debian file
#
# If you install the official VRT Sourcefire rules please review this
# configuration file and re-enable (remove the comment in the first line) those
# rules files that are available in your system (in the /etc/snort/rules
# directory)
#
# site specific rules
include $RULE_PATH/local.rules
#
# The include files commented below have been disabled
# because they are not available in the stock Debian
# rules. If you install the Sourcefire VRT please make
# sure you re-enable them again:
#
# include $RULE_PATH/app-detect.rules
# include $RULE_PATH/attack-responses.rules
# include $RULE_PATH/backdoor.rules
# include $RULE_PATH/bad-traffic.rules
# include $RULE_PATH/blacklist.rules

```

Fig. 5. RULE\_PATH for Local and community rules

In this research, we will be creating our custom rules to detect IDS evasion techniques under ‘local.rules’ file, in order to learn and understand the internal workings of intrusion detections systems.

```

root@ritwik-host-vm: /etc/snort/rules

1 # $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
2 # -----
3 # LOCAL RULES
4 # -----
5 # This file intentionally does not come with signatures. Put your local
6 # additions here.
7

```

Fig. 6. ‘local.rules’ file for custom rules

In order to test all the configurations, we run snort in self-test mode which checks all the supplied command line switches and rules files.

```

root@ritwik-host-vm: /home/ritwik-host# sudo snort -T -i ens33 -c /etc/snort/snort.conf
Running in Test mode

--== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/snort.conf"
PortVar 'HTTP_PORTS' defined : [ 80:81 311 383 591 593 901 1220 1414 1741 1830 2301 2381 2809 30
88 7000:7001 7144:7145 7510 7777 7779 8000 8008 8014 8028 8080 8085 8088 8090 8118 8123 8180:818
9000 9060 9080 9090:9091 9443 9999 11371 34443:34444 41080 50002 55555 ]
PortVar 'SHELLCODE_PORTS' defined : [ 0:79 81:65535 ]
PortVar 'ORACLE_PORTS' defined : [ 1024:65535 ]
PortVar 'SSH_PORTS' defined : [ 22 ]
PortVar 'FTP_PORTS' defined : [ 21 2100 3535 ]
PortVar 'SIP_PORTS' defined : [ 5060:5061 5600 ]
PortVar 'FILE_DATA_PORTS' defined : [ 80:81 110 143 311 383 591 593 901 1220 1414 1741 1830 2301
4848 5250 6988 7000:7001 7144:7145 7510 7777 7779 8000 8008 8014 8028 8080 8085 8088 8090 8118 8
800 8888 8899 9000 9060 9080 9090:9091 9443 9999 11371 34443:34444 41080 50002 55555 ]
PortVar 'GTP_PORTS' defined : [ 2123 2152 3186 ]

```

Fig. 7. Snort in self-Test mode

#### IV. IDS EVASION TECHNIQUES DEMONSTRATION

In this section, we emulate a few attack scenarios, where threat actors can find a flaw in victim’s machine and exploit it to carry out an IDS evasion technique to go undetected by existing defence mechanisms. Further, to strengthen our

defences, we develop rulesets for our IDS i.e. Snort to trigger alerts in case of any suspicious anomaly observed in our network being monitored.

All experiments are done in virtual environment with VMware Fusion. The test VM environment consists of an attacker (Kali Linux) and a victim (Ubuntu) machine with IDS (Snort) setup in Ubuntu machine itself.

```

root@ritwik-host-vm: /etc/snort/rules# ifconfig
ens160: flags=419<UP,BROADCAST,RUNNING,PROMISC,MULTICAST> mtu 1500
    inet 192.168.72.129 netmask 255.255.255.0 broadcast 192.168.72.255
    inet6 fe80::214d:d519:cb1:5450 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:5e:5c:37 txqueuelen 1000 (Ethernet)
    RX packets 1181856 bytes 696623677 (696.6 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 204615 bytes 61067721 (61.0 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 46 memory 0x3fe00000-3fe20000

```

In order to generate the attacks, evade detection, and inspect the evasions, numerous tools and scripts were utilized in this study. For reconnaissance phase, port scanning via Nmap. Next, in the exploitation phase, tools like Hydra and Hping3 are used for brute force and DOS attacks.

##### A. SSH Brute Force Attack for Payload Encryption Evasion

In order to gain access to target machine, we need to carry out recon using Nmap tool. Based on the response, we can identify any attack surface to carry out the exploitation phase.

```

root@ritwik-host-vm: /etc/snort/rules# nmap -A 192.168.72.129 -Pn
Starting Nmap 7.94 ( https://nmap.org ) at 2023-12-13 23:28 GMT
Nmap scan report for 192.168.72.129
Host is up (0.0055s latency).
Not shown: 997 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.4 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_ 256 86:52:87:9c:e3:7c:75:ak:a9:49:d5:00:58:82:fc:f2 (ECDSA)
|_ 256 db:e7:16:8b:e4:57:29:c1:4d:32:6c:e2:02:29:65:9b (ED25519)
53/tcp    closed domain
3306/tcp  closed mysql
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.15 seconds

```

From this port scan, we observe that on the target machine Port 22 (SSH) is left open by default. This can allow a threat actor to gain access to victim machine and execute arbitrary commands.

Next, in order to SSH into the target machine, we require the username and password which a threat actor can brute force using tools like HYDRA.

```

root@ritwik-host-vm: /etc/snort/rules# hydra -l root -P ubuntu 192.168.72.129 ssh
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-12-13 16:05:51
[DATA] max 4 tasks per 1 server, overall 4 tasks, 1211 login tries (119/p/1211), ~228 tries per task
[121] attacking 192.168.72.129:22
[121][ssh] host: 192.168.72.129 login: root password: ubuntu
1 of 4 targets successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-12-13 16:05:54

```

We choose the default user account i.e. ‘root’ and used a wordlist to carry out dictionary attack to guess the SSH password i.e. ‘ubuntu’. As now we have both username and password, we are able to successfully SSH into the target machine and execute any arbitrary commands. As SSH creates an encrypted tunnel, all the commands executed further will pass through encrypted channel and thus IDS won’t be able to detect and create any alerts.



```
(b00168210@kali)-[~]
└─$ ssh root@192.168.72.129
root@192.168.72.129's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-39-generic aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

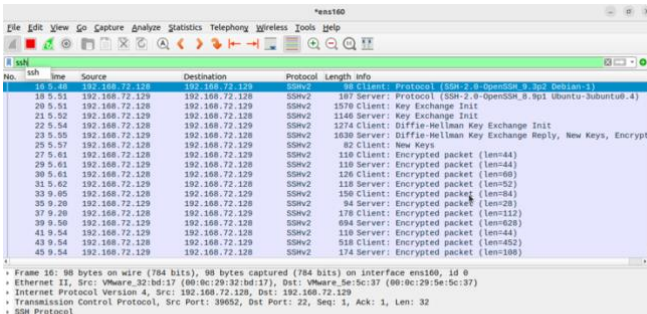
Expanded Security Maintenance for Applications is not enabled.

1 update can be applied immediately.
To see these additional updates run: apt list --upgradable

7 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

Last login: Mon Dec 11 16:48:36 2023 from 192.168.72.128
root@ritwik-host-vm:~# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
```

This can also be examined in the victim machine's Wireshark where SSH communication packets are observed.



With the help of below mentioned rule, we'll be able to detect any SSH connection attempts going forward.

```
root@ritwik-host-vm:/etc/snort/rules
1 # $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
2 #
3 # LOCAL RULES
4 #
5 # This file intentionally does not come with signatures. Put your local
6 # additions here.
7
8 # ICMP ping
9 alert icmp any any -> $HOME_NET any (msg:"ICMP Ping Detected"; sid:10101; rev:1;)
10
11 # SSH connection attempt
12 alert tcp any any -> $HOME_NET 22 (msg:"SSH connection attempt"; sid:10102; rev:1;)
13
14 # SYN attack
15 alert tcp any any -> $HOME_NET any (msg:"SYN attack"; flags:S,12; sid:10103; rev:1;)
16
17 # XMAS scan
18 alert tcp any any -> $HOME_NET 22 (msg:"XMAS Scan"; flags:PF; sid:10104; rev:1;)
19
20 # Detect TCP SYN Flood attack - a bogus
21 alert tcp any any -> $HOME_NET 80 (msg:"Potential TCP SYN Flood attack detected";
22 flags:S;
23 threshold: type threshold, track by_dst, count 500, seconds 1;
24 sid:101; rev:1;)
25
26
27 # Alert rule to detect SSH brute force attack
28 alert tcp any any -> $HOME_NET 22 (msg:"Potential SSH Brute-Force attempt"; \
29 flow:to_server; \
30 content:"SSH"; nocase; \
31 flags:S; \
32 detection_filter:track by_src, count 5, seconds 10; \
33 sid:1; \
34 rev:1;)
```

These are the alerts observed with this rule.

```
root@ritwik-host-vm:/etc/snort/rules# sudo snort -q -i /home/ritwik-host-vm/snort/logs -t ens160 -A console -e /etc/snort/snort.conf
12/13-23:49:50.062103 ** [1:10102:1] SSH connection attempt ** (Priority: 0) (TCP 192.168.72.128:52318 -> 192.168.72.129:22)
12/13-23:49:50.065598 ** [1:10102:1] SSH connection attempt ** (Priority: 0) (TCP 192.168.72.128:52318 -> 192.168.72.129:22)
12/13-23:49:50.067675 ** [1:10102:1] SSH connection attempt ** (Priority: 0) (TCP 192.168.72.128:52318 -> 192.168.72.129:22)
12/13-23:49:50.116747 ** [1:10102:1] SSH connection attempt ** (Priority: 0) (TCP 192.168.72.128:52318 -> 192.168.72.129:22)
12/13-23:49:50.156017 ** [1:10102:1] SSH connection attempt ** (Priority: 0) (TCP 192.168.72.128:52318 -> 192.168.72.129:22)
12/13-23:49:50.165148 ** [1:10102:1] SSH connection attempt ** (Priority: 0) (TCP 192.168.72.128:52318 -> 192.168.72.129:22)
```

### B. TCP/SYN Flooding Attack – Tactical DDoS

In SYN flooding attack, a threat actor repeatedly sends SYN packets to target/victim's IP address. In response, the victim repeatedly keeps sending a SYN,ACK acknowledgment packet to the attacker, conveying that he/she is ready to establish a connection. Ideally, if an attacker really wanted to establish a connection with the victim, they would send back an ACK packet to the victim in response to their

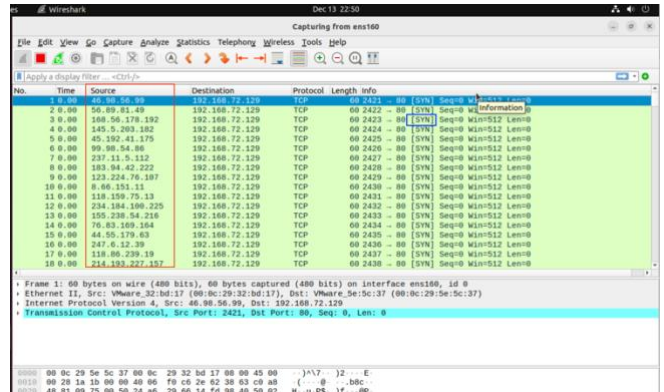
SYN, ACK packet. However, during a SYN flooding attack, the attacker never acknowledges the victim's SYN,ACK packets, but keeps sending their SYN packets to the victim, which overwhelms the network SYN packets. Finally, when the victim's network is heavily flooded with SYN packets and the victim acknowledges one of these SYN packets with their SYN,ACK packet, the attacker cleverly continually sends reset RST packets to the victim so that he can further continue flooding the victim's network with SYN packets. This attack technique can be nefariously used by threat actor to overwhelm the IDS system with SYN packets depleting the IDS host system's resources.

To carry out this attack, we use Hping3 tool to flood TCP SYN packets to IDS machine with IP (192.168.72.129).

```
(b00168210@kali)-[~]
└─$ sudo hping3 -S -p 80 --flood --rand-source 192.168.72.129 -V
Using eth0, addr: 192.168.72.128, MIU: 1500
HPING 192.168.72.129 (eth0 192.168.72.129): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown

- 192.168.72.129 hping statistic -
1327597 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

This sends a huge amount of SYN packets which can be examined with the help of Wireshark on the victim's machine.



Here, we can observe that the Source IP addresses are unique for each packet. This is a case of Distributed Denial of Service (DDoS) attack carried using the '--rand-source' option of Hping3 tool. With this attack, threat actor can make the IDS system unavailable by overwhelming it with bogus SYN packets.

Now, for a security analyst to detect this IDS evasion technique, we attempt to build a Rule in Snort's local rules file that is capable of detecting such SYN flooding attacks and triggering alerts for the same to be actioned upon by an analyst.

```
root@ritwik-host-vm:/etc/snort/rules
1 # $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
2 #
3 # LOCAL RULES
4 #
5 # This file intentionally does not come with signatures. Put your local
6 # additions here.
7
8 # ICMP ping
9 alert icmp any any -> $HOME_NET any (msg:"ICMP Ping Detected"; sid:10101; rev:1;)
10
11 # SSH connection attempt
12 alert tcp any any -> $HOME_NET 22 (msg:"SSH connection attempt"; sid:10102; rev:1;)
13
14 # SYN attack
15 alert tcp any any -> $HOME_NET any (msg:"SYN attack"; flags:S,12; sid:10103; rev:1;)
16
17 # XMAS scan
18 alert tcp any any -> $HOME_NET 22 (msg:"XMAS Scan"; flags:PF; sid:10104; rev:1;)
19
20 # Detect TCP SYN Flood attack - a bogus
21 alert tcp any any -> $HOME_NET 80 (msg:"Potential TCP SYN Flood attack detected";
22 threshold: type threshold, track by_dst, count 500, seconds 1;
23 sid:101; rev:1;)
24
25
26 # Alert rule to detect SSH brute force attack
27 alert tcp any any -> $HOME_NET 22 (msg:"Potential SSH Brute-Force attempt"; \
28 flow:to_server; \
29 content:"SSH"; nocase; \
30 flags:S; \
31 detection_filter:track by_src, count 5, seconds 10; \
32 sid:1; \
33 rev:1;)
```

In this highlighted (green) rule, we trigger an alert when more than 500 packets in less than 5 seconds are received with SYN flag on destination port 80. This rule triggers an alert with message “Potential TCP SYN Flood attack detected”.

```
root@rtwik-host-vm: /etc/snort/rules
root@rtwik-host-vm: /etc/snort/rules
root@rtwik-host-vm: /etc/snort/rules# sudo snort -q -l /home/rtwik-host/Snortlogs/ -i ens108 -A console -c /etc/snort/snort.conf
12/13-22:49:06.538016 *** [1:10:1] Potential TCP SYN Flood attack detected *** [Priority: 0] (TCP) 121.96.175.57:2920 -> 192.168.72.129:80
12/13-22:49:06.566642 *** [1:10:1] Potential TCP SYN Flood attack detected *** [Priority: 0] (TCP) 177.66.55.123:3420 -> 192.168.72.129:80
12/13-22:49:06.638275 *** [1:10:1] Potential TCP SYN Flood attack detected *** [Priority: 0] (TCP) 118.19.164.145:3920 -> 192.168.72.129:80
12/13-22:49:07.467292 *** [1:10:1] Potential TCP SYN Flood attack detected *** [Priority: 0] (TCP) 170.59.56.234:4420 -> 192.168.72.129:80
12/13-22:49:08.426212 *** [1:10:1] Potential TCP SYN Flood attack detected *** [Priority: 0] (TCP) 195.123.142.11:4920 -> 192.168.72.129:80
12/13-22:49:09.215816 *** [1:10:1] Potential TCP SYN Flood attack detected *** [Priority: 0] (TCP) 185.127.36.39:5420 -> 192.168.72.129:80
12/13-22:49:10.013702 *** [1:10:1] Potential TCP SYN Flood attack detected *** [Priority: 0] (TCP) 11.56.69.241:1920 -> 192.168.72.129:80
12/13-22:49:10.825192 *** [1:10:1] Potential TCP SYN Flood attack detected *** [Priority: 0] (TCP) 136.127.247.185:6420 -> 192.168.72.129:80
```

In this way we can train our IDS to detect and alert any such TCP SYN flooding attack scenarios.

## V. CONCLUSION

In this paper, we identified different ways to evade IDS like tactical DDoS and Payload encryption techniques. We

were successfully able to implement these techniques along with creating specific snort rules to detect them as well. Cyber defenders can leverage this intelligence to better understand the threat landscape and recommend streamlined recommendations to enterprises as preventive controls to tackle similar cyberattacks and security incidents in the future.

## REFERENCES

- [1] N. Khamphakdee, N. Benjamas and S. Saiyod, "Improving Intrusion Detection System based on Snort rules for network probe attack detection", 2nd International Conference on Information and Communication Technology (ICoICT), 2014
- [2] M. Ashraf, "TryHackMe | Network Security Solutions Evasions - Mohamed Ashraf - medium," Medium, May 26, 2023. [Online]. Available: <https://medium.com/@Mx0o14/tryhackme-network-security-solutions-evasions-8a14de2dcc1#:~:text=Evasion%20via%20protocol%20manipulation%20includes,Sending%20invalid%20packets>