Hands-on Lab Description

ThoTh Lab

CS-CNS-00003 –
Network Intrusion Detection (Snort)

CONTENTS

Category:

```
CS-CNS: Computer Network Security
```

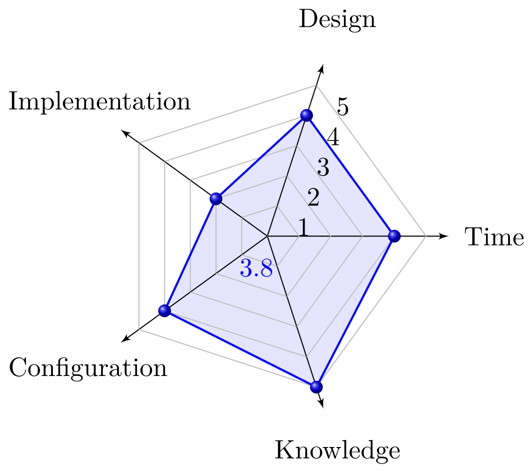
Objectives:

```
Setup network intrusion detection (snort) to detect malicious network traffic
Setup syslog to collect alerts generated by snort
Implement various network attacks for security testing
```

Estimated Lab Duration:

```
Expert: 180 minutes
Novice: 360 minutes
```

Difficulty Diagram:



Difficulty Table.

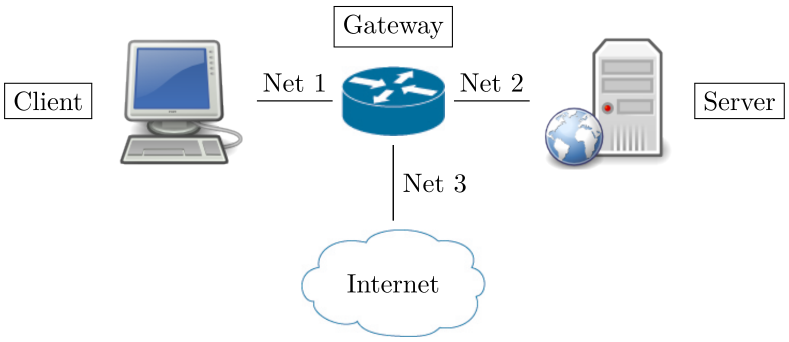
Measurements	Values (0-5)
Time	4
Design	4
Implementation	2
Configuration	4
Knowledge	5
Score (Average)	3.8

Required OS:

```
Linux: Ubuntu 18.04 LTS (Bionic Beaver)
```

Lab Running Environment:

```
ThoTh Lab: https://thothlab.org
```



```
Client: Linux (Ubuntu 18.04 LTS)
Server: Linux (Ubuntu 18.04 LTS)
Gateway: Linux (Ubuntu 18.04 LTS)
Network Setup:
Internet is connected through Net 3: 172.16.0.0/12
Client-side Net 1: 192.168.0.0/24
Server-side Net 2: 10.0.0.0/8
```

Lab Preparations:

```
Know how to use Linux OS (Reference Labs: CS-SYS-00001)
Basic knowledge about computer networking (Reference Labs: CS-NET-00001 and CS-NET-00002)
Know how to setup and use syslog for remote logging (Reference Labs: CS-SYS-00009)
Know how to deploy network testing and attacks using hping3 (Reference Labs: CS-CNS-10003)
```

Lab Overview

In this lab, students will explore the Snort Intrusion Detection Systems. Students will study Snort IDS, a signature-based intrusion detection system used to detect network attacks. Snort can also be used as a simple packet logger. For this lab, the students will use snort as a packet sniffer and write their own IDS rules.

Ethical Claim: This lab provides students access to the public domain. The attacks deployed in this lab potentially can damage the public domain. Thus, we require each student to agree the following requirements when performing this lab:

1. Agree not to purposely deploy network attacks targeting any public domain services or computers.

2. Remove the default gw on the gateway nodes to block all traffic to the public domain. To do so, you can issue the following command on your gateway node:
   $ *route -n* % identify the default gateway IP address, e.g., 172.16.0.1
   $ *sudo route del default gw 172.16.0.1* % delete your default gw to the public domain.
   By removing your default gateway, you can prevent accidentally attack external public services/nodes. You can resume the default gateway when you want to download necessary software packages from the public domain:
   $ *sudo route add default gw 172.16.0.1 -i ens3* % assume your default gw IP address is 172.16.0.1, and it is assigned to interface ens3.

3. The deployed attacks are resource-intensive. Thus, make sure to stop the attack once you finished the test. PLEASE do not leave or sign-off your system, letting the deployed DoS attack continuously running. This will significantly slow down the response time of the virtual lab.

1 Task 1 Snort Basics and Preparation

Snort is an open-source network intrusion detection and prevention system. It can analyze real-time traffic analysis and data flow in the network. It can check protocol analysis and can detect different types of attacks. In NIDS, snort basically checks the packet against rule written by the user. Snort rules can be written in any language; its structure is also good, and it can be easily read, and rules can also be modified. In a buffer overflow attack, snort can detect the attack by matching the previous attack pattern and then will take appropriate action to prevent the attack. In a signature-based IDS system, if the pattern matches, an attack can be easily found, but when a new attack comes, the system fails but snort overcome this limitation by analyzing real-time traffic. Whenever any packet comes into the network, then snort checks the network's behavior. If performance degrades the network, then snort stops the packet's processing, discards the packet, and stores its detail in the signature database.

1.1 Components of Snort

Snort is basically the combination of multiple components. All the components work together to find a particular attack and then take the corresponding action required for that particular attack. Basically it consists of following major components as shown in Figure CS-CNS-00003.1:

A packet comes from the internet and enters into the packet decoder, and it goes through several phases; required action is taken by snort at every phase like if the detection engine found any miscellaneous content in the packet, then it drops that packet and in the way towards output module packet is logged in, or alert is generated. The main functions of snort components are described as follows:

- *Packet Decoder*: is responsible for forming packets to be used by the other components. It has the job of determining which underlying protocols are used in the packet and determining the location and

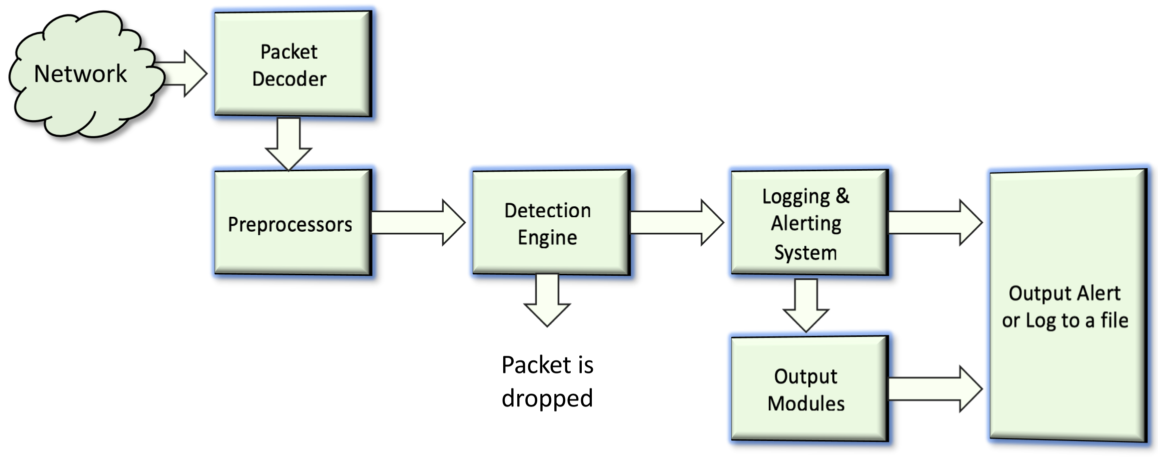


Figure CS-CNS-00003.1
Components of Snort.

size of the packet data, which is then used in later components. It should be noted that the Decoder also looks for anomalies in headers (such as invalid sizes), which may then cause it to generate alerts.

- *Preprocessors*: are used to verify anomalies specific to a service such as HTTP or FTP. Its job is ultimately to try and make it harder to fool the detection engine. Snort makes components work as plugins, and these plugin preprocessors can arrange or modify packet data. Thus, each service can have a corresponding preprocessor. Examples of using preprocessors are: decoding URI's to address defragmenting packets, in which fragmentation of packets can be used to fool the detection engine, detecting port scanning as well as to detect anomalies in ARP packets, such as ARP spoofing, etc.
- *Detection Engine*: has the responsibility to "detect if any intrusion activity exists in a packet". It does this by chaining together sets of rules, specified in configuration files which include these rules, and applying them to each packet. If the packet matches a rule, that rule's specified action is taken, or the packet is dropped. If snort performs this in real-time, depending on the network load, latency may be experienced, with worst-case scenarios resulting in packets being dropped altogether.
- *Logging and Alerting System*: If a packet is matched to a rule, the log and or alert will be generated by the Logging and Alerting System. The message and contents generated by this component can, of course, be configured through the configuration file. If a packet triggers multiple rules, the highest alert level is what will actually be generated by this component.
- *Output Modules*: are tasked with controlling the type of output generated, uses a plugin system giving the user flexibility, and is also highly configurable. This may include simply logging or logging to a database, sending SNMP traps, generating XML reports, or even sending alerts through UNIX sockets, allowing for (for example) dynamic modification of network configurations (Firewalls or Routers).

1.2 Preparation of setting up lab environment

Notice: This lab can run its own for only practice network intrusion detection. By using syslog (Lab-CS-SYS-00009 Syslog (rsyslog) for Linux), you can practice how to inspect network intrusions from multiple IDS sensors.

The first step is to check if snort is properly installed on the running VM.

```
$ snort --v
```

If snort is installed, it may show the following output:

```
root@ubuntu:~# snort --v

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.7.0 GRE (Build 149)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.8.1
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.2.11
```

To check the snort running status, you can run the following command:

```
service snort status
```

If it is running properly, it may show something like:

```
* snort.service - LSB: Lightweight network intrusion detection system
   Loaded: loaded (/etc/init.d/snort; generated)
   Active: active (running) since Fri 2020-10-02 16:25:11 UTC; 1min 15s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 25269 ExecStart=/etc/init.d/snort start (code=exited, status=0/SUCCESS)
    Tasks: 2 (limit: 2317)
   CGroup: /system.slice/snort.service
           --25319 /usr/sbin/snort -m 027 -D -d -l /var/log/snort -u snort -g snort -c
               /etc/snort/snort.conf -S HOME_NET=[192.168.0.0/24] -i ens4

Oct 02 16:25:05 ubuntu systemd[1]: Starting LSB: Lightweight network intrusion
    detection system...
Oct 02 16:25:05 ubuntu snort[25269]: * Starting Network Intrusion Detection System
    snort
Oct 02 16:25:11 ubuntu snort[25269]: ...done.
Oct 02 16:25:11 ubuntu systemd[1]: Started LSB: Lightweight network intrusion
    detection system.
```

The configuration files of snort are located in the */etc/snort* folder.

If the system does not contain snort, then you need to install it. Due to the variations of Linux distribution and snort versions, the installation procedure varies. A general approach is to issue:

```
$ sudo apt-get update -y % -y flag means to assume yes and silently install, without
    asking you questions in most cases.
$ sudo apt-get install -y snort*
```

Note that snort is built on several software packages of the Linux distribution. Usually, you may encounter the required dependencies. Snort is a libpcap-based packet sniffer/logger used as a lightweight network intrusion detection system. It features rules based logging and can perform content searching/matching in addition to detecting a variety of other attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, and much more. Snort has a real-time alerting capability, with alerts being sent to syslog, a separate "alert" file, or even to a Windows computer via Samba.

During snort installation, it will ask a few set-up questions. Some of configurations are set up in the */etc/snort/snort.debian.conf* files, such as interface and home network, etc. You can manually modify the configuration file, or you can run the following command to reconfigure the setup:

```
$ sudo dpkg-reconfigure snort
```

2 Task 2: Running Snort

In this task, we present the basics on how to setup and run snort.

2.1 Network Intrusion Detection System Setup

In general, to make snort a network IDS, you can issue a common command as the following format:

```
$ sudo snort -[options] -i [sniffing interface] -l [logfile] -h [home-net] -c
    [configuration file]
```

Each of the options given after the command snort is optional. An example is given as follows:

```
$ sudo  snort -dev -i ens5 -l /var/log/snort -h 10.0.0.0/8 -c /etc/snort/snort.conf
```

First, we need to understand the snort network infrastructure setup. Other options are explained in the following subsections. The network infrastructure of this lab is presented in Figure CS-CNS-00003.2.

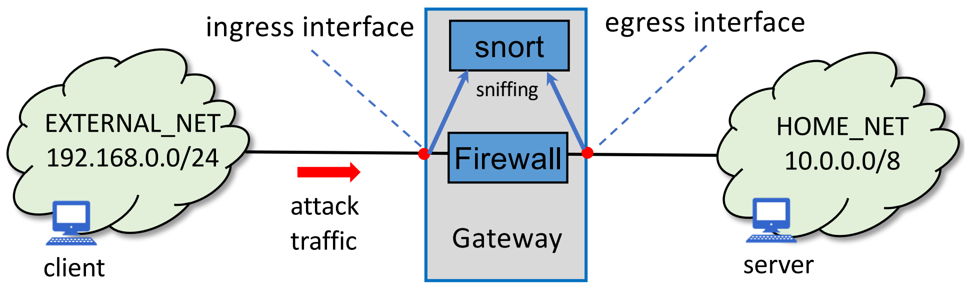


Figure CS-CNS-00003.2
Snort running environment.

Note that option "-h" determines the network that you want to protect. With this variable set, all decoded packet logging is done relative to the home network address space. In our experiment setup, we consider the home network is 10.0.0.0/8, where the server vm is attached, and the network 192.168.0.0/24 is an external network, where the attacker may locate on. In the following examples and illustrations, we assume the following IP address setup:

- client IP: 192.168.0.7
- ingress IP of the gateway: 192.168.0.3
- egress IP of the gateway: 10.0.0.4
- server IP: 10.0.0.6 (running ssh, web, ftp, dns services, etc.)

In a network intrusion detection system, snort is usually set into a sniffing mode. It only captures the traffic passing through the system instead of putting it in an "*in-line*" to intercept the traffic. Where putting the snort to sniff the traffic depends on the system security setup and requirements.

In Figure CS-CNS-00003.2, the snort can sniff one of two interfaces, i.e., the ingress interface or the egress interface. When sniffing the ingress interface, snort will capture all the traffic sent to the firewall. To detect attacks targeting at the firewall, this setup is appropriate. However, for detecting attacks targeting internal network systems, snort may consume significant computing resources to perform detection, which is not fully utilizing the benefit of the firewall's filtering function. Thus, for passing through traffic, snort is usually set up to sniff traffic that had already been inspected by the firewall.

You can set up the sniffing interface either on the ingress interface or egress interface in this lab. Ensure that your sniffing interface and HOME_NET need to be set up correctly to make sure that your snort can detect and identify the packet pattern.

2.2 Illustration of Snort Rules

Snort offers its user to write their own rule for generating logs of Incoming/Outgoing network packets. Only they need to follow the snort rule format where packets must meet the threshold conditions. Always bear in mind that the snort rule can be written by combining two main parts "the Header" and "the Options" segment. Basically, rules are created by a known intrusion signature system. It is divided into two parts:

```
rule header (rule option)
```

The snort rule format is presented in Figure CS-CNS-00003.3.

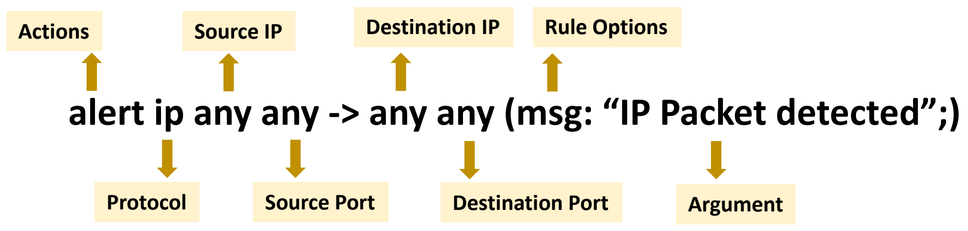


Figure CS-CNS-00003.3
Snort Rule Format.

The header part contains information such as the action, protocol, the source IP and port, the network packet Direction operator towards the destination IP and port. The remaining will be considered in the options part.

```
action protocol source port -> destination port (options)
```

In the header filed:

- Action: It informs Snort what kind of action to be performed when it discovers a packet that matches the rule description. There are five existing default job actions in Snort: alert, log, pass, activate, and dynamic are keyword use to define the action of rules. You can also go with additional options, which include drop and reject.
- Protocol: After deciding the option for action in the rule, you need to describe specific Protocols (IP, TCP, UDP, ICMP, any) on which this rule will be applicable.
- Source IP: This part of the header describes the sender network interface from which traffic is coming.
- Source Port: This part of the header describes the source Port from which traffic is coming.
- Direction operator ("->", "<>"): It denotes the direction of traffic flow between sender and receiver networks.
- Destination IP: This part of the header describes the destination network interface in which traffic is coming for establishing the connection.
- Destination Port: This part of the header describes the destination Port on which traffic is coming for establishing the connection.

Thus, there are two levels of logical considerations of detection/matching operations:

- *Within a rule*: all of the elements that make up a rule must be true for the indicated rule action to be taken. When taken together, the elements can be considered to form a logical **AND** statement.
- *Among multiple rules*: Simultaneously, the various rules in a Snort rules library file can be considered to form a large logical **OR** statement.

2.3 Rule Options (Metadata)

The body for rule option is usually written between circular brackets "()" that contains keywords with their argument and separated by semicolon ";" from another keyword. Note that any rule does not specifically require the rule options section; they are just used for the sake of making tighter definitions of packets to collect or alert on (or drop, for that matter). Metadata is part of the optional rule that basically contains additional information about the snort rule that is written with the help of some keywords and their argument details. There are four major categories of rule options:

- General: These options contain metadata that offers information about them.
- Payload: These options all come across for data contained by the packet payload and can be interconnected.
- Non-payload: These options come across for non-payload data.
- Post-detection: These options are rule-specific triggers that happen after a rule has fired.

The snort general rule options (metadata) is presented in Table CS-CNS-00003.1.

Table CS-CNS-00003.1
General Rule Options (metadata)

Keyword	Description
msg	The msg keyword stands for "Message" that informs to snort that written argument should be print in logs while analyst of any packet.
reference	The reference keyword allows rules to a reference to information present on other systems available on the Internet such as CVE.
gid	The gid keyword stands for "Generator ID "which is used to identify which part of Snort create the event when a specific rule will be launched.
sid	The sid keyword stands for "Snort ID" is used to uniquely identify Snort rules.
rev	The rev keyword stands for "Revision" is used to uniquely identify revisions of Snort rules.
classtype	The classtype keyword is used to assigned classifications and priority numbers to the group and distinguish them a rule as detecting an attack that is part of a more general type of attack class. Syntax: config classification: name, description, priority number.
priority	The priority keyword to assigns a severity rank to your rules.

It should be noted that while most options are optional, the sid (Snort ID) is required, and should not conflict with the SID of another rule. It is the unique identifier given to each rule. Snort reserves SIDs from 0 - 1,000,000.

2.4 Packet sniffer

In its simplest form, snort is a packet sniffer. That said, its the easiest way to start.

```
$ sudo snort -options
```

where the options can be:

```
-v Put Snort in packet-sniffing mode (TCP headers only)
-d Include all network layer headers (TCP, UDP, and ICMP)
-e Include the data link layer headers
-p capturing without putting the interface into promiscuous mode
```

Note that both option "-d" and "-e" can consume system resource and their usage should be avoided for production system.

2.5 Packet logger

Snort has built-in packet-logging mechanisms that you can use to collect the data as a file, sort it into directories, or store the data as a binary file.

```
$ sudo snort -l {logging-directory} -h {home-subnet-slash-notation}
```

If you wanted to log the data into the directory */var/snort/logs*, you would use the following:

```
$ sudo snort -l /var/log/snort
```

If you want to instruct snot to alert in the terminal, then you can use the option "-A console":

```
$ sudo snort -A console -c /etc/snort/snort.conf
```

For logging in binary format, you do not need all options. The binary format makes packet collection much faster for Snort, because Snort does not have to translate the data into human readable format immediately.

```
$ sudo snort -l {log-file} -b
```

Option "-b" means binary mode. Binary mode logs the packets in tcpdump format to a single binary file in the logging directory.

For reading the log file, you can use the following snort command:

```
$ sudo snort [-d|e] -r {log-file} [tcp|udp|icmp]
```

Here last item in line is optional, because if you want to filter the packets based on packet type like tcp , udp or icmp.

Note that snort could have output you two kinds of output file format depending on snort output plugin option for that files:

- tcpdump pcap, or
- snort's unified2.

In order to know what kind format generated, e.g., the log file /var/log/snort/snort.log. You can use *file* command to check the log file:

```
$ sudo file /var/log/snort/snort.log
```

If it tells you *tcpdump capture file*, you can use wireshark, tshark -r, tcpdump -r, or snort -r to read the file. If it tells you *data*, It is in the *unified2* format, which is a "Native" snort format. You can read it with *u2spewfoo <file>* that is included in snort, or convert it to a pcap by using *u2boat*. In this lab, you can simply add "-b" option to create the tcpdump capture file formation, and you can use snort -r to read the log file.

You can also add various other options to tweak snort's behavior. For example, "*-A Fast*" turns on a shorter alert log format, which can reduce disk accesses and log file size. Whether or not you use this option, Snort creates a log directory tree, much like the one it creates when you use snort as a packet sniffer. One difference is that the "Fast" log files contain less data; only packets that match a rule are logged. There is also a file called alert, which contains a summary of all the suspicious activity snort has detected.

To send alerts to syslog, use the "-s" switch. The default facilities for the syslog alerting mechanism are LOG_AUTHPRIV and LOG_ALERT. If you want to configure other facilities for syslog output, use the output plugin directives in snort.conf.

For example, use the following command line to log to default (decoded ASCII) facility and send alerts to syslog:

```
$ sudo snort -c /etc/snort/snort.conf -l /var/log/snort -h 10.0.0.0/8 -s
```

3 Task 3: Create and Test Snort Rules

To check whether the Snort is logging any alerts as proposed, add a detection rule alert on IP packets in the "local.rules file". To test your created new snort rules, you can disable all snort rules files presented in the *snort.conf* except the *local.rules*. First back up your snort.conf:

```
$ cp snort.conf snort.backup.conf
```

Then, you can edit snort.conf file:

```
$ vim /etc/snort/snort.conf
```

You can comment out all snort rules files by adding a "#" symbol before each rule file, such as:

```
# site specific rules
include $RULE_PATH/local.rules  % leave the local.rules uncommented
...
#include $RULE_PATH/app-detect.rules % comment all other rules files
#...
```

To this end, you can run the following command to comment out all the include statement, and then later uncomment the line that includes *local.rules*:

```
$ sudo sed -i 's/include \$RULE_PATH/#include \$RULE_PATH/' /etc/snort/snort.conf
```

3.1 Task 3.1 Create and test icmp rules

Now edit */etc/snort/rules/local.rules* and add the following icmp rule:

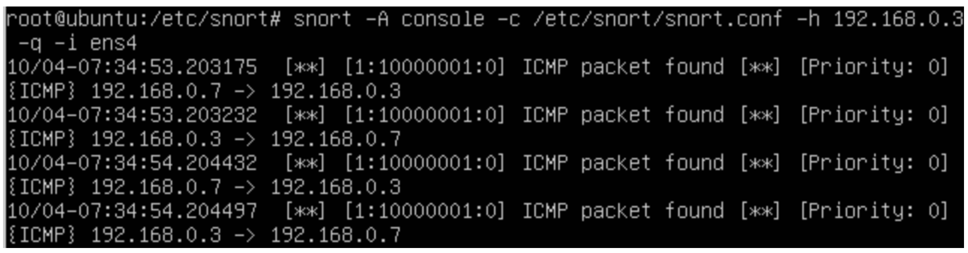
```
alert icmp any any <> 192.168.0.3 any (msg: "ICMP Packet found"; sid:10000001; )
```

In this example, you can set snort to sniff the ingress interface, e.g., *ens4*, with the IP address 192.168.0.3. The option "<>" means that you want to capture ping messages in both directions.

To test the icmp rule, you can run snort command:

```
$ sudo snort -A console -c /etc/snort/snort.conf -q -i ens4
```

For testing, you can issue snort without a logging option. In this command, the option "-q" is to suppress the initial snort information message. The option "-i" specify the sniffing network interface is *ens4* that is configured as the IP address 192.168.0.3 in this example. Then, you can test by either ping to another IP address (e.g., 192.168.0.7 is the IP address of the client in our example); or from 192.168.0.7, you can ping the snort sniffing interface IP address 192.168.0.3. The snort detection result is printed on the console as presented in Figure CS-CNS-00003.4.



```
root@ubuntu:/etc/snort# snort -A console -c /etc/snort/snort.conf -h 192.168.0.3
 -q -i ens4
10/04-07:34:53.203175  [**] [1:10000001:0] ICMP packet found [**] [Priority: 0]
{ICMP} 192.168.0.7 -> 192.168.0.3
10/04-07:34:53.203232  [**] [1:10000001:0] ICMP packet found [**] [Priority: 0]
{ICMP} 192.168.0.3 -> 192.168.0.7
10/04-07:34:54.204432  [**] [1:10000001:0] ICMP packet found [**] [Priority: 0]
{ICMP} 192.168.0.7 -> 192.168.0.3
10/04-07:34:54.204497  [**] [1:10000001:0] ICMP packet found [**] [Priority: 0]
{ICMP} 192.168.0.3 -> 192.168.0.7
```

Figure CS-CNS-00003.4
Snort ICMP Alert.

3.2 Task 3.2 Create and test snort rules using system variables

To manage your snort more effectively, you should use system parameters instead of using hard-coded numerical values. For example, you can use system variables EXTERNAL_NET and HOME_NET in your rule setup. These two system-level variables can be set up in the /etc/snort/snort.conf file. For example, you can edit the following:

```
ipvar HOME_NET 10.0.0.0/8 % set the server-side network as the home net

ipvar EXTERNAL_NET !$HOME_NET
```

In this example, the protected HOME_NET is set to the network 10.0.0.0/24. We can use "!" (i.e., NOT) to specify networks other than HOME_NET are considered as EXTERNAL_NET.

When you start snort from the command-line, you can use -h option to override the HOME_NET given in the snort.conf file. For example,

```
$ sudo snort -A console -h 10.0.0.0/24 -c /etc/snort/snort.conf -q -i ens4
```

HOME_NET can be a single IP address, and it also can be multiple IP addresses, can be a network, or can be mixed. Here, we list a few examples:

```
var HOME_NET [10.0.0.6,10.0.0.6,192.168.0.3] % list individual IP addresses
var HOME_NET 10.10.10.0/8 % set the server-side network as home net
var HOME_NET [192.168.0.3,10.0.0.0/8] % set the server-side network and ingress IP as
    the home net
```

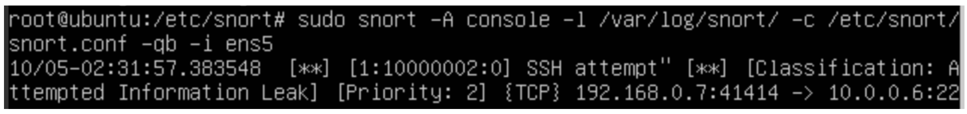
Then, you can append a TCP rule in the /etc/snort/rules/local.rules as following:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"SSH attempt";flags:S;
    classtype:attempted-recon; sid:10000002; rev:0;)
```

This rule tells the system to issue an alert message when it detects traffic from a computer on $EXTERNAL_NET directed at port 22 on a $HOME_NET (local) computer. The information within parentheses includes a logging message and classification codes. The flags keyword indicates TCP flags "*S*" indicates that the rule matches only packets on which the SYN flag is set. In this example, attack classifications defined by Snort reside in the *classification.config* file. The keyword and value "classtype: attepmpted-recon" represents attempted information leak attacks. You will also find information on packet contents (indicated by a content keyword) or other packet features in many cases.

Once you are satisfied with your snort ruleset, you can launch snort. Typically, you will have snort log its output, which you can peruse later. You can launch snort to function as a NIDS using a command such as the following:

```
$ sudo snort -c /etc/snort/snort.conf -l /var/log/snort -b -q -i ens5 % in this
    example, ens5 is the egress interface behind the firewall
```



```
root@ubuntu:/etc/snort# sudo snort -A console -l /var/log/snort/ -c /etc/snort/
snort.conf -qb -i ens5
10/05-02:31:57.383548  [**] [1:10000002:0] SSH attempt" [**] [Classification: A
ttempted Information Leak] [Priority: 2] {TCP} 192.168.0.7:41414 -> 10.0.0.6:22
```

Figure CS-CNS-00003.5
Snort captured ssh connection.

In Figture CS-CNS-00003.5, it shows the ssh connection captured at the egress interface (internal network).

3.3 Task 3.3 Create and test snort rules using payload rule options

As described in the previous *Rule Options* subsection, payload options shows what data to be captured in the packet content. If data matching the argument data string is contained anywhere within the packet's payload, the test is successful, and the remainder of the rule option tests are performed. The match is case sensitive by default. For example, the following rule matches HTTP to get the message.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (msg:"Http Get
    request";content:"GET";classtype:web-application-activity;sid:10000003;rev:0;)
```

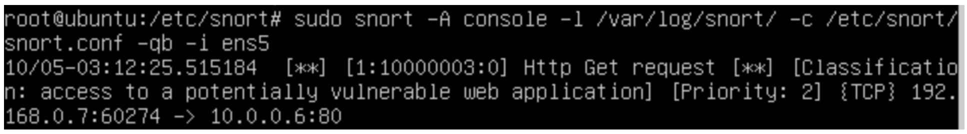
We can mix and match the text and binary data. We enclose hexadecimal representation of binary data within pipe ('|') characters to specify binary data. If we want to specify more than one binary character, we use space (' ') to separate the binary characters from each other. An example is presented as follows:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (content:"|5c 00|P|00|I|00|P|00|E|00
    5c|";)
```

Now you can test the above specified http get detection rule by issuing an http access from the client to the server:

```
$ curl 10.0.0.6    % you can access the IP address directly and run: curl
    http://10.0.0.6
```

The captured packet is presented in Figure CS-CNS-00003.6



```
root@ubuntu:/etc/snort# sudo snort -A console -l /var/log/snort/ -c /etc/snort/
snort.conf -qb -i ens5
10/05-03:12:25.515184  [**] [1:10000003:0] Http Get request [**] [Classificatio
n: access to a potentially vulnerable web application] [Priority: 2] {TCP} 192.
168.0.7:60274 -> 10.0.0.6:80
```

Figure CS-CNS-00003.6
Snort captured TCP Get message.

3.4 Task 3.4 Create and test snort rules using Non-payload Rule Options

As described in previous *Rule Options* subsection, non-payload options come across for non-payload data such as packet head field or protocol related options. For example, you can use *ttl* keyword to check the IP time-to-live value. This option keyword was intended for use in the detection of traceroute attempts. This keyword takes numbers from 0 to 255.

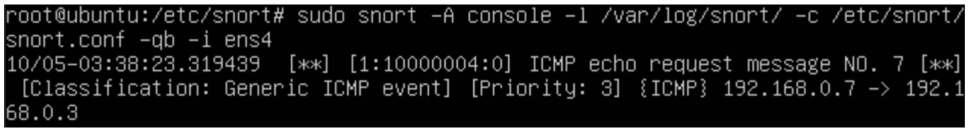
You can also use icmp_seq to identify the number of received icmp message. For example

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP echo request message NO.
    7";classtype:icmp-event;icmp_seq:7;sid:10000004;rev:0;)
```

Now, you can test ping by issuing the ping message:

```
$ ping -c 8 192.168.0.7
```

Then the snort packet detection results is shown in Figure CS-CNS-00003.7.



```
root@ubuntu:/etc/snort# sudo snort -A console -l /var/log/snort/ -c /etc/snort/
snort.conf -qb -i ens4
10/05-03:38:23.319439  [**] [1:10000004:0] ICMP echo request message NO. 7 [**]
 [Classification: Generic ICMP event] [Priority: 3] {ICMP} 192.168.0.7 -> 192.1
68.0.3
```

Figure CS-CNS-00003.7
Snort captured icmp message with sequence number equals to 7.

4 Task 4: Lab requirements

In the previous tasks, we had presented several walk-through on how to set up and run snort. To set up your lab running environment, you need to work in a simple client-gateway-server networking system, as shown in Figure CS-CNS-00003.2. In this environment, you should:

1. Enable packet forwarding on the Gateway. Note that you have two interfaces that snort can sniff the traffic. For testing purposes, you can sniff the client-side of network IP (ingress interface IP). However, some attack scenarios (e.g., smurf attack) may need you to to set up your firewall (e.g., setup the system given in Lab CS-CNS-00001 Packet filter firewall) to emulate a real firewall system and implement the sniffing the server-side of the network (i.e., the egress interface IP). Also, you can set up the Apache, ssh, FTP, DNS services on the server.

2. If you have a firewall established on the gateway, e.g., iptables, you can disable all the network filter rules and enable blacklist (iptables -P chain-name ACCEPT) for testing purpose to allow all the traffic pass through. In this lab, we consider this setup is OK for the minimum requirement.

3. In all the following described requirements, you need to comment on all the rules specified in the /etc/snort/snort.conf, except the rule file /etc/snort/rules/local.rules you need to implement and demonstrate your snort rules in this file.

4.1 Task 4.1 Land Attack Deployment and Detection

Land attack occurs when an attack host sends spoofed TCP SYN packets (connection initiation) with the target host's IP address and the TCP port as both source and destination. The reason a Land attack works is because it causes the machine to reply to itself continuously. That is, the target host responds by sending the SYN-ACK packet to itself, creating an empty connection that lasts until the idle timeout value is reached. Flooding a system with such empty connections can overwhelm the system, causing a DoS situation.

In this task, you need to

1. deploy Land attack using hping3 (refer to LAB-CNS-10003 hping3 Tutorial) from the client to the server. Attack requirements:
   - TCP header: source port=80, Destination port=80, TCP SYN Flag=1.
   - IP Header: source IP=Target host's IP address, Destination IP=Target host's IP address

2. implement snort rule to detect Land attack.

Note that latest Linux OS usually come with address spoofing attack protection. If this message pass a gateway node (e.g., a Linux), it may stop forwarding the spoofed packet. To disable the address spoofing protection on a gateway, you can do the follows:

1. Set the following configuration in the */etc/sysctl.conf*:

```
net.ipv4.conf.default.rp_filter = 0
net.ipv4.conf.all.rp_filter = 0
net.ipv4.ip_forward = 1
net.ipv4.conf.all.accept_redirects = 1
net.ipv4.conf.all.send_redirects = 1
```

2. Restart the gateway.

Then the gateway will be able to forward the spoofed packet.

4.2 Task 4.2 SYN Flood Attack Deployment and Detection

A SYN flood occurs when a host becomes so overwhelmed by TCP SYN packets initiating incomplete connection requests that it can no longer process legitimate connection requests. In fact, when a client system attempts to establish a TCP connection to a system providing a service (the server), the client and server exchange a sequence of messages known as a three-way handshake. The client system begins by sending a SYN (synchronization) message to the server. The server then acknowledges the SYN message by sending a SYN-ACK (acknowledgment) message to the client. The client then finishes establishing the connection by responding with an ACK message.

TCP incomplete connections are created when SYN messages are sent to the victim server using spoofed source IP addresses. Hence, the victim server will never receive ACK messages from the spoofed clients to complete establishing the connections. Flooding the victim server with such spoofed TCP SYN traffic can overwhelm the server, causing a DoS situation.

In this task, you need to

1. deploy SYN flood attack using hping3 (refer to LAB-CNS-10003 hping3 Tutorial) from the client to the server. Attack requirements:
    - TCP header: source port=any, destination port=open TCP port number, TCP SYN flag=1.
    - source IP=spoofed or random IP address (e.g., 192.168.0.100), destination IP=target host's IP address
2. implement snort rule to detect SYN flood attack. Special threshold detection requirement: you need to logs every 20th event on this attack during a 60-second interval.

4.3 Task 4.3 Smurf Attack Deployment and Detection

The Smurf DoS attack consists into sending a large amount of ICMP echo request (ping) traffic to a broadcast IP address, all of it having a spoofed source IP address of a victim host. Each host on the network represented by that broadcast IP address will take the ICMP echo request and send an ICMP echo reply to the victim host, multiplying the traffic by the number of hosts responding.

In this task, you need to

1. deploy Smurf attack using hping3 (refer to LAB-CNS-10003 hping3 Tutorial) from the client to the server.
    - ICMP header: type=8 (echo request), code=0.
    - IP header: source IP=Victim host's IP address, destination IP=Broadcast IP address
2. implement snort rule to detect smurf attack. Specific requirements are given as follows:
    (a) The victim is the server (10.0.0.x), the targeting broadcasting network is 192.168.0.0/24, and the broadcasting IP address is 192.168.0.255.
    (b) By default Ubuntu does not respond to a broadcasting ping. To enable the broadcasting ping targeting at the gateway, you need to change the gateway's setup in any of the following approaches:

```
$ sudo echo 0 >/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts % enable
    temporary responds to broadcasting request
$ sudo sysctl net.ipv4.icmp_echo_ignore_broadcasts 0
```

or you can modify the /etc/sysctl.conf by adding the following line:

```
net.ipv4.icmp_echo_ignore_broadcasts=0 % you may need to restart the node
```

After the change, you can issue a broadcasting message to the gateway, e.g., 192.168.0.255, and you will expect a reply message originated from the gateway to the server's IP address (i.e., the spoofed IP address).

(c) You need to create a snort rule to detect smurf attack (icmp reply message) originated from the gateway.

4.4 Task 4.4 UDP Flood Attack Deployment and Detection

A UDP flood attack consists into flooding target UDP ports on a victim system with UDP packets. If enough UDP packets are delivered to the destination UDP port, the victim host or UDP application may slow down or go down.

In this task, you need to

1. deploy UDP flood attack using hping3 (refer to LAB-CNS-10003 hping3 Tutorial) from the client to the server. Hint: in this attack, you need to target the gateway's IP address on the client side, i.e., 192.168.0.3.
   - UDP header: source port=any, destination port=open UDP port number.
   - IP header: source IP=spoofed or random IP address, destination IP=target host's IP address.
2. implement snort rule to detect UDP flood attack. Special detection requirement: you need to logs every 10th event on this attack during a 60-second interval.

4.5 Task 4.5 Port Scanning Deployment and Detection

A port scanning is a method for determining which ports on a network are open. As ports on a computer are the place where information is sent and received, port scanning is analogous to knocking on doors to see if someone is home.

In this task, you need to

1. deploy port scanning using hping3 (refer to LAB-CNS-10003 hping3 Tutorial) from the client to the server. Hint: in this attack, you need to target the gateway's IP address on the client side, i.e., 192.168.0.3.
   - TCP ACK Scan.
   - TCP FIN Scan.
   - TCP Xmas Scan.
   - TCP Null Scan (you may want to use nmap).
   - UDP Scan.
2. implement snort rules to detect above described scans. Special detection requirement: you need to logs every 20th event on this attack during a 60-second interval.

4.6 Task 4.6 Set up syslog for snort

In this task, you wan to use a centralized log server to store your snort issued alerts. Please refer to LAB-SYS-00009 (Syslog (rsyslogd) Remote Logging on Linux) to set up your syslog service on the server node. The requirements are given as follows:

1. Set up rsyslog service on the server node to receive the rsyslog client from the gateway. (Note that you need to configure rsyslog on both the gateway and server to allow gateway send its snort events to the server)
2. You need to log all your captured events from snort rules specified for Tasks 4.1-4.5 in the binary format, which is tcpdump compatible to your log server.
3. Demonstrate the logged events on the log server given from task 4.1 to task 4.5.

5 Lab Assessments (100 points)

Lab assessment for accomplishing Task 4 depends on the following facts:

1. (16 points) Successfully demonstrate the attack and detection in task 4.1.
2. (16 points) Successfully demonstrate the attack and detection in task 4.2.
3. (16 points) Successfully demonstrate the attack and detection in task 4.3.
4. (16 points) Successfully demonstrate the attack and detection in task 4.4.
5. (16 points) Successfully demonstrate the attack and detection in task 4.5.
6. (20 points) Successfully demonstrate remote logs specified in task 4.6.

6 Related Information and Resource

```
Snort Users Manual 2.9.16:
http://manual-snort-org.s3-website-us-east-1.amazonaws.com/snort_manual.html
Snort IDS Tools:
http://books.gigatux.nl/mirror/snortids/0596006616/main.html
Snort Command-Line Options:
http://books.gigatux.nl/mirror/snortids/0596006616/snortids-CHP-3-SECT-3.html
Snort
Snort Configuration:
https://www.sbarjatiya.com/notes_wiki/index.php/Snort_configuration
```