

## TP : Configuring Snort IDS

### Introduction :

In this lab we're going to learn how to install and configure SNORT intrusion detection system in simple network. For this, we will need a virtual machine Linux built on Ubuntu and Kali over Virtualbox.

### Tasks :

#### Step 1: Setting Up your Lab Environment

1. Download and install kali linux
2. Download and install ubuntu desktop

#### Step 2: Install and configure snort on your ubuntu machine:

1. Open your terminal
2. Make sure to update your system using:  
Command: `sudo apt-get update` and `sudo apt-get upgrade`
3. Install Snort using:  
Command: `sudo apt-get install snort`  
During the installation, you will be asked to configure the network. Use the IP address of your Ubuntu VM for the "HOME\_NET" configuration.
4. Edit the Snort configuration file (located usually at `/etc/snort/snort.conf`)  
Make sure that "HOME\_NET" variable is correctly set to your network which is the IP of your Ubuntu VM ( If you plan to use this for many VMs, set the network range to include your VMs)

In the example below do this if you want to monitor the Ubuntu machine on which Snort is installed.

```
# Note to Debian users: this value is overridden when starting
# up the Snort daemon through the init.d script by the
# value of DEBIAN_SNORT_HOME_NET s defined in the
# /etc/snort/snort.debian.conf configuration file
#
ipvar HOME_NET 192.168.0.0/24
```

If you want to monitor an entire subnet use this:  
`ipvar HOME_NET 192.168.0.0/24` for example.

5. Set EXTERNAL\_NET:  
If you want "EXTERNAL\_NET" to represent everything outside your "HOME\_NET", set it as `!$HOME_NET`

```
# Note to Debian users: this value is overridden when starting
# up the Snort daemon through the init.d script by the
# value of DEBIAN_SNORT_HOME_NET s defined in the
# /etc/snort/snort.debian.conf configuration file
#
ipvar HOME_NET [redacted]

# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET !$HOME_NET
```

This configuration will tell Snort to treat all IP addresses that are not part of “HOME\_NET” as external.

P.S: You can set your own rules and configuration depending on the different setups and environments you are using.

6. Make sure DNS servers, SMTP servers, web servers sql servers, telnet on your network are set as follows “\$HOME\_NET”.

```
# value of DEBIAN_SNORT_HOME_NET s defined in the
# /etc/snort/snort.debian.conf configuration file
#
ipvar HOME_NET [redacted]

# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET !$HOME_NET
# If HOME_NET is defined as something other than "any", alternative, you can
# use this definition if you do not want to detect attacks from your internal
# IP addresses:
#ipvar EXTERNAL_NET !$HOME_NET

# List of DNS servers on your network
ipvar DNS_SERVERS $HOME_NET
```

7. In step #1 in the configuration file, you will find this:

```
# List of ports you run web servers on
portvar HTTP_PORTS [80,81,311,383,591,593,901,1220,1414,1741,1830,2301,2381,280]

# List of ports you want to look for SHELLCODE on.
portvar SHELLCODE_PORTS !80

# List of ports you might see oracle attacks on
portvar ORACLE_PORTS 1024:

# List of ports you want to look for SSH connections on:
portvar SSH_PORTS 22

# List of ports you run ftp servers on
portvar FTP_PORTS [21,2100,3535]

# List of ports you run SIP servers on
portvar SIP_PORTS [5060,5061,5600]
```

You can leave this as it is for now, but you can also modify this according to your needs and setups (close ports, use different ports, etc ...)

Hit save and close this for now.

## 8. Now, Create Your Custom Rule File:

It's better practice to create a new rule file and keep those custom rules separated from other rules files.

To do this:

- Create a new file in “/etc/snort/rules” directory.
- Name it “custom.rules” ( You can name it whatever you want, i.e testing.rules)
- Add your Custom Rule by opening the ”custom.rules” that you just created:

```
ubuntu@ubuntu:/etc/snort/rules$ ls
attack-responses.rules      community-web-dos.rules      p2p.rules
backdoor.rules              community-web-iis.rules      policy.rules
bad-traffic.rules           community-web-misc.rules     pop2.rules
chat.rules                  community-web-php.rules      pop3.rules
community-bot.rules         custom.rules                  porn.rules
community-deleted.rules     ddos.rules                   rpc.rules
community-dos.rules         deleted.rules                 rservices.rules
community-exploit.rules     dns.rules                     scan.rules
community-ftp.rules         dos.rules                     shellcode.rules
community-game.rules        experimental.rules           smtp.rules
community-icmp.rules        exploit.rules                 snmp.rules
community-imap.rules        finger.rules                  sql.rules
community-inappropriate.rules ftp.rules                      telnet.rules
community-mail-client.rules icmp-info.rules              tftp.rules
community-misc.rules        icmp.rules                    virus.rules
community-nntp.rules        imap.rules                    web-attacks.rules
community-oracle.rules      info.rules                    web-cgi.rules
community-policy.rules      local.rules                    web-client.rules
community-sip.rules         misc.rules                     web-coldfusion.rules
community-smtp.rules        multimedia.rules              web-frontpage.rules
community-sql-injection.rules mysql.rules                    web-iis.rules
community-virus.rules       netbios.rules                 web-misc.rules
community-web-attacks.rules nntp.rules                     web-php.rules
community-web-cgi.rules     oracle.rules                   x11.rules
community-web-client.rules  other-ids.rules
ubuntu@ubuntu:/etc/snort/rules$ nano custom.rules
```

Add the following rule inside your file:

```
alert icmp any any -> $HOME_NET any (msg:"ICMP Test"; sid:1000001;)
```

```
GNU nano 6.2 custom.rules
alert icmp any any -> $HOME_NET any (msg:"ICMP Test"; sid:1000001;)

[ File 'custom.rules' is unwritable ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^_ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

Save the file and exit the editor.

9. Open snort.conf again using “sudo nano /etc/snort/snort.conf”

Include the new created Rule File at the end of Step 7:

```
include $RULE_PATH/community-sql-injection.rules
include $RULE_PATH/community-web-client.rules
include $RULE_PATH/community-web-dos.rules
include $RULE_PATH/community-web-iis.rules
include $RULE_PATH/community-web-misc.rules
include $RULE_PATH/community-web-php.rules
include $RULE_PATH/community-sql-injection.rules
include $RULE_PATH/community-web-client.rules
include $RULE_PATH/community-web-dos.rules
include $RULE_PATH/community-web-iis.rules
include $RULE_PATH/community-web-misc.rules
include $RULE_PATH/community-web-php.rules
include /etc/snort/rules/custom.rules

#####
# Step #8: Customize your preprocessor and decoder alerts
# For more information, see README.decoder_preproc_rules
#####
```

Double check that Include “custom.rules” has the same name as the “custom.rules” you just created.

Save and exit the editor.

10. Test Snort Configuration for any errors using this command:

```
sudo snort -T -c /etc/snort/snort.conf
```

Ensure no errors are reported that related to the new rule that you created.

```
$ sudo snort -T -c /etc/snort/snort.conf
```

```
Snort successfully validated the configuration!
Snort exiting
```

11. Restart Snort:

To apply the changes you need to restart Snort using command:

```
sudo systemctl restart snort
```

```
ubuntu@ubuntu:~$ sudo systemctl restart snort
```

## 12. Monitor and Analyze:

You can run Snort in console mode to see alerts in real time using this command:

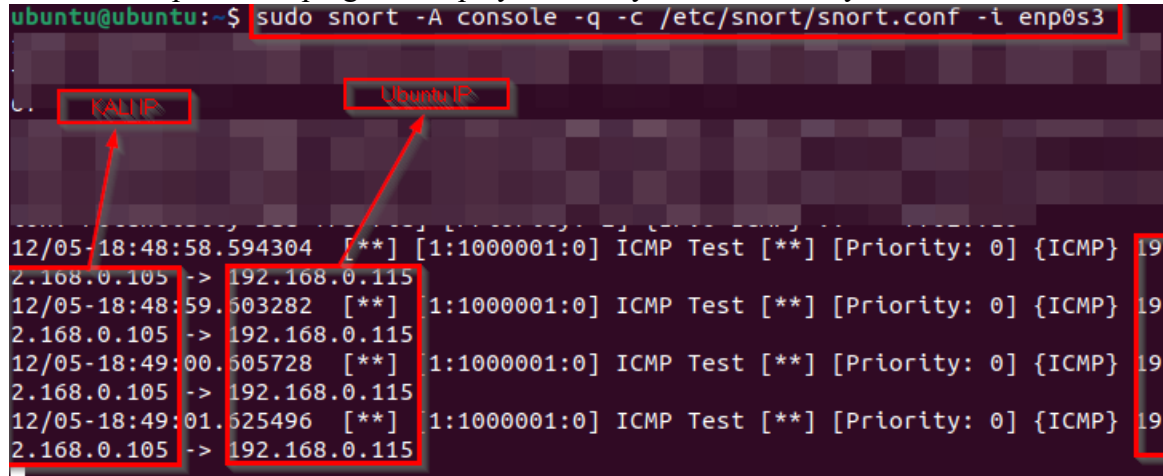
```
sudo snort -A console -q -c /etc/snort/snort.conf -I [interface, i.e, eth0]
```

```
sudo snort -A console -q -c /etc/snort/snort.conf -i enp0s3
```

Using your KALI machine, ping your ubuntu machine.

Snort will capture that ping and display it inside your console on your ubuntu.

```
ubuntu@ubuntu:~$ sudo snort -A console -q -c /etc/snort/snort.conf -i enp0s3
```



```
12/05-18:48:58.594304 192.168.0.105 [**] [1:1000001:0] ICMP Test [**] [Priority: 0] {ICMP} 192.168.0.115
2.168.0.105 -> 192.168.0.115
12/05-18:48:59.503282 192.168.0.105 [**] [1:1000001:0] ICMP Test [**] [Priority: 0] {ICMP} 192.168.0.115
2.168.0.105 -> 192.168.0.115
12/05-18:49:00.505728 192.168.0.105 [**] [1:1000001:0] ICMP Test [**] [Priority: 0] {ICMP} 192.168.0.115
2.168.0.105 -> 192.168.0.115
12/05-18:49:01.525496 192.168.0.105 [**] [1:1000001:0] ICMP Test [**] [Priority: 0] {ICMP} 192.168.0.115
2.168.0.105 -> 192.168.0.115
```

As you can see, Snort has captured the ping request from your kali machine (the attacker machine) AND provided it's IP addressing showing who exactly is trying to ping your ubuntu machine.

This concludes the demonstration of live analysis inside the Snort Console with a simple ping test from an attacker machine (kali) to your victim machine (ubuntu).

13. You can also check Snort logs that are located inside “/var/log/snort/alert”, to see the ICMP traffic that was detected.

## Step 3: Detect SQL Injection Attempts

### 1. Install Apache and PHP:

Open your terminal on your ubuntu machine.

Install Apache and PHP using this command: `sudo apt-get install apache2 php libapache2-mod-php`

### 2. Create a basic PHP Page for testing:

- a) Navigate to the web root using : `cd /var/www/html`
- b) Create a simple PHP file : `sudo nano test.php`
- c) Add the following content and save:

```
<?php
echo "Test page. Your input: ";
if(isset($_GET['input'])) {
    echo $_GET['input'];
}
?>
```

```
ubuntu@ubuntu:/var/www/html$ ls
index.html  test.php
```

This page will echo back any input passed via the “input” GET parameter.

3. Start the Apache service using: `sudo systemctl start apache2`
4. Edit your custom Snort rule file using: `sudo nano /etc/snort/rules/custom.rules`

Add a rule to detect common SQL injection payloads using:

```
alert tcp any any -> $HOME_NET 80 (msg:"SQL Injection attempt";
flow:to_server,established; content:"SELECT"; nocase; sid:1000002;)
```

```
GNU nano 6.2 /etc/snort/rules/custom.rules *
alert icmp any any -> $HOME_NET any (msg:"ICMP Test"; sid:1000001;)
alert tcp any any -> $HOME_NET 80 (msg:"SQL Injection attempt"; flow:to_server,established; content:"SELECT"; nocase; sid:1000002;)
```

This rule will look for the keyword “SELECT” in HTTP traffic to port 80 which is common SQL command used in injections.

5. Restart snort using: `sudo systemctl restart snort`
6. Inside your kali machine (the attacker), try to curl this basic test.php that you created using:  
`curl 'http://192.168.0.115/test.php?input=SELECT%20*%20FROM%20users'`

```
(root@kali)~# curl 'http://192.168.0.115/test.php?input=SELECT%20*%20FROM%20users'
Test page. Your input: SELECT * FROM users
```

7. Verify Alerts by using: `sudo cat /var/log/snort/alert` (alert might have a different name on your ubuntu).

#### **Step 4:**

Now, configure your own rules to simulate a Denial of Service (DoS) attack using hping3 from your kali machine. Configure Snort on your ubuntu to detect this attack.

1. Open snort.
2. Add a DoS Detection rule  

```
alert tcp any any -> $HOME_NET any (flags: S; threshold: type both, track by_src, count 100, seconds 1; msg:"Possible SYN Flood Attack"; sid:1000003;)
```

This rule triggers an alert if there are more than 100 SYN packets from the same source in one second indicating a SYN flood.
3. Restart Snort to apply the new rule
4. Use hping3 on your kali machine to simulate an attack.  
`sudo hping3 -S --flood -p 80 [Ubuntu_IP]`
5. Monitor the Snort Alerts on Ubuntu.