



SNORT

A Detailed Guide



Table of Contents

Abstract.....	3
Introduction to Snort	4
Installation of snort	4
Snort Rule Format	10
Detect NMAP Scan Using Snort	17
Identify NMAP Ping Scan	17
Identify NMAP TCP Scan	19
Identify NMAP XMAS Scan	20
Identify NMAP FIN Scan	22
Identify NMAP NULL Scan	23
Identify NMAP UDP Scan	25
Detect SQL Injection Attack using Snort IDS	26
Identify Error Based SQL Injection	26
Testing Double Quotes Injection	28
Boolean Based SQL Injection	30
Testing OR Operator	31
Encoded AND/OR	33
Testing Encoded OR Operator	34
Identify Form Based SQL Injection	36
Identify Order by SQL Injection	37
Identify Union Based SQL Injection	39
Conclusion	41
References	41



Abstract

This report serves as a guide to understanding Intrusion Detection Systems (IDS) with a focus on utilizing Snort as an example. The discussion will cover the installation and configuration of Snort as an IDS in a Linux environment.

Furthermore, this report will delve into the detection of NMAP scans and SQL injection attacks using Snort. This information aims to assist network security analysts and administrators in setting up Snort rules and implementing effective actions against suspicious traffic.

Disclaimer: This report is provided for educational and informational purpose only (Penetration Testing). Penetration Testing refers to legal intrusion tests that aim to identify vulnerabilities and improve cybersecurity, rather than for malicious purposes.



Introduction to Snort

Snort is a Network Intrusion Detection System (NIDS). It's quite popular and is open-source software which helps in monitor network traffic in real-time, hence it can also be considered as a packet sniffer. Basically, it examines each and every data packet in depth to see if there are any malicious payloads. it can also be used for protocol analysis and content searching. It is capable of detecting various attacks like port scans, buffer overflow, etc. It's available for all platforms i.e., Windows, Linux, etc. It doesn't require any recompilation with the system or hardware to added to your distribution; root privileges are required though. It inspects all the network traffic against the provided set of rules and then alerts the administration about any suspicious activity. it's divided into multiple components and all the components work together to detect an intrusion. Following are the major components of snort:

- Packet Decoder
- Pre-processors
- Detection Engine
- Logging and Alerting System
- Output Modules

Installation of snort

First, use the ifconfig command in your Ubuntu to check the interface. As you can see the image below the interface is ens33.

```
pentest@ubuntu:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.21 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::1af7:6beb:970b:e74a prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:40:da:6e txqueuelen 1000 (Ethernet)
    RX packets 8882 bytes 12505286 (12.5 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2553 bytes 213839 (213.8 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 218 bytes 16051 (16.0 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 218 bytes 16051 (16.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

pentest@ubuntu:~$
```

Now, let's install snort by using the following command:



```
sudo apt-get install snort*
```

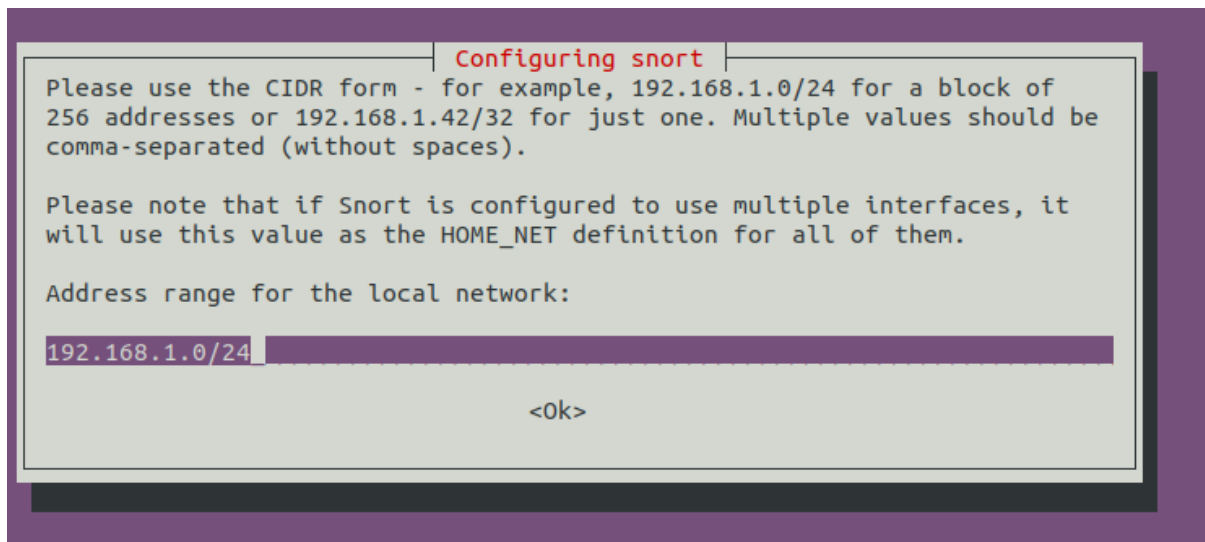
```
pentest@ubuntu:~$ sudo apt-get install snort*
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'snort-pgsql' for glob 'snort*'
Note, selecting 'snort-doc' for glob 'snort*'
Note, selecting 'snort-rules-default' for glob 'snort*'
Note, selecting 'snort-common' for glob 'snort*'
Note, selecting 'snort-mysql' for glob 'snort*'
Note, selecting 'snort' for glob 'snort*'
Note, selecting 'snort-common-libraries' for glob 'snort*'
Note, selecting 'snort-rules' for glob 'snort*'
The following additional packages will be installed:
  libdaq2 oinkmaster
The following NEW packages will be installed:
  libdaq2 oinkmaster snort snort-common snort-common-libraries snort-doc snort-rules-default
0 upgraded, 7 newly installed, 0 to remove and 431 not upgraded.
Need to get 3,255 kB of archives.
After this operation, 17.4 MB of additional disk space will be used.
```

Once the installation starts, it will ask you the interface that we previously checked. Give its name here and press enter.

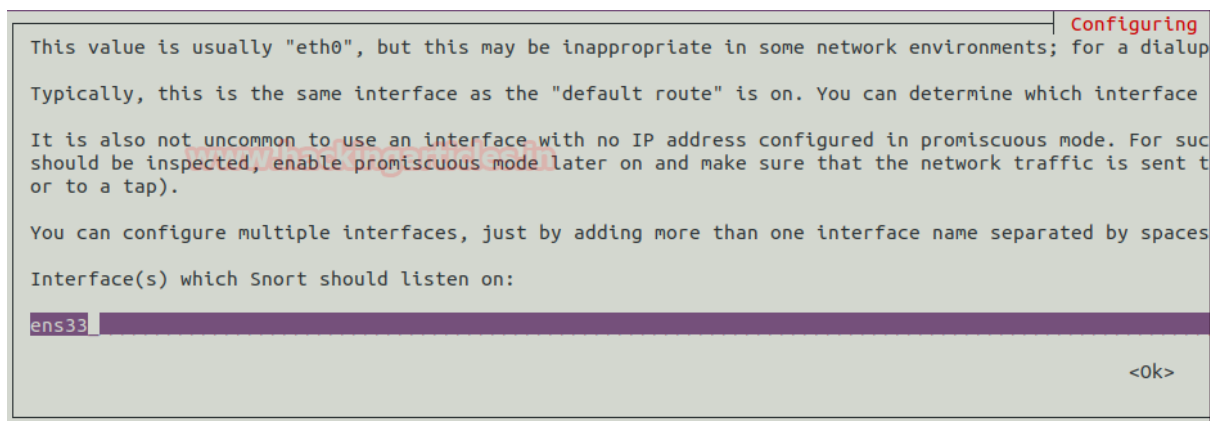
```
This value is usually "eth0", but this may be inappropriate in some network environments;
Typically, this is the same interface as the "default route" is on. You can determine which
It is also not uncommon to use an interface with no IP address configured in promiscuous mode
should be inspected, enable promiscuous mode later on and make sure that the network traffic
or to a tap).

You can configure multiple interfaces, just by adding more than one interface name separated by
Interface(s) which Snort should listen on:
ens33
```

Then it will ask you about your network IP. Here, you can either provide a single IP or the range of IPs as we have given below in the image:



Then possible, it will again ask you for the name of the interface, provide it again and press enter.



As the snort is installed, open the configuration file using nano or any text editor to make some changes inside. Use the following command to do so:

```
sudo nano /etc/snort/snort.conf
```

Scroll down the text file near line number 45 to specify your network for protection as shown in the given image.

#Setup the network addresses you are protecting

```
ipvar HOME_NET 192.168.1.21
```



```
#####
# Step #1: Set the network variables.  For more information, see README.variables
#####

# Setup the network addresses you are protecting
#
# 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 192.168.1.21

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

# List of SMTP servers on your network
ipvar SMTP_SERVERS $HOME_NET
```

Now run given below command to enable IDS mode of snort:

```
sudo snort -A console -i ens33 -c /etc/snort/snort.conf
```

The above command will compile the complete file and test the configuration setting automatically as shown in given below image:



```
+-----+
[ Number of patterns truncated to 20 bytes: 1039 ]
pcap DAQ configured to passive.
Acquiring network traffic from "ens33".
Reload thread starting..
Reload thread started, thread 0x7fbe18885700 (4934)
Decoding Ethernet

--== Initialization Complete ==--

o"~
    -*> Snort! <*-
    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

    Rules Engine: SF_SNORT_DETECTION_ENGINE Version 2.4 <Build 1>
    Preprocessor Object: SF_DNS Version 1.1 <Build 4>
    Preprocessor Object: SF_REPUTATION Version 1.1 <Build 1>
    Preprocessor Object: SF_SDF Version 1.1 <Build 1>
    Preprocessor Object: SF_IMAP Version 1.0 <Build 1>
    Preprocessor Object: SF_MODBUS Version 1.1 <Build 1>
    Preprocessor Object: SF_SMTP Version 1.1 <Build 9>
    Preprocessor Object: SF_DNP3 Version 1.1 <Build 1>
    Preprocessor Object: SF_SSLPP Version 1.1 <Build 4>
    Preprocessor Object: SF_SIP Version 1.1 <Build 1>
    Preprocessor Object: SF_GTP Version 1.1 <Build 1>
    Preprocessor Object: SF_DCERPC2 Version 1.0 <Build 3>
    Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>
    Preprocessor Object: SF_SSH Version 1.1 <Build 3>
    Preprocessor Object: SF_POP Version 1.0 <Build 1>
    Commencing packet processing (pid: 4934)
```

Once the snort is installed and configured, we can start making changes to its rules as per our own requirement and desire. To the rules on which snort works use the following command:

```
cd /etc/snort/rules
```

```
ls -la
```

As shown in the image below, you can find all the documents related to rules.



```
pentest@ubuntu:~$ cd /etc/snort/rules/
pentest@ubuntu:/etc/snort/rules$ ls -la
total 1608
drwxr-xr-x 2 root root 4096 Feb 15 00:08 .
drwxr-xr-x 3 root root 4096 Feb 15 00:11 ..
-rw-r--r-- 1 root root 5520 Apr 3 2018 attack-responses.rules
-rw-r--r-- 1 root root 17898 Apr 3 2018 backdoor.rules
-rw-r--r-- 1 root root 3862 Apr 3 2018 bad-traffic.rules
-rw-r--r-- 1 root root 7994 Apr 3 2018 chat.rules
-rw-r--r-- 1 root root 12759 Apr 3 2018 community-bot.rules
-rw-r--r-- 1 root root 1223 Apr 3 2018 community-deleted.rules
-rw-r--r-- 1 root root 2042 Apr 3 2018 community-dos.rules
-rw-r--r-- 1 root root 2176 Apr 3 2018 community-exploit.rules
-rw-r--r-- 1 root root 249 Apr 3 2018 community-ftp.rules
-rw-r--r-- 1 root root 1376 Apr 3 2018 community-game.rules
-rw-r--r-- 1 root root 689 Apr 3 2018 community-icmp.rules
-rw-r--r-- 1 root root 2777 Apr 3 2018 community-imap.rules
-rw-r--r-- 1 root root 948 Apr 3 2018 community-inappropriate.rules
-rw-r--r-- 1 root root 257 Apr 3 2018 community-mail-client.rules
-rw-r--r-- 1 root root 7837 Apr 3 2018 community-misc.rules
-rw-r--r-- 1 root root 621 Apr 3 2018 community-nntp.rules
-rw-r--r-- 1 root root 775 Apr 3 2018 community-oracle.rules
-rw-r--r-- 1 root root 1621 Apr 3 2018 community-policy.rules
-rw-r--r-- 1 root root 3551 Apr 3 2018 community-sip.rules
-rw-r--r-- 1 root root 2722 Apr 3 2018 community-smtp.rules
-rw-r--r-- 1 root root 4063 Apr 3 2018 community-sql-injection.rules
-rw-r--r-- 1 root root 3742 Apr 3 2018 community-virus.rules
-rw-r--r-- 1 root root 2406 Apr 3 2018 community-web-attacks.rules
-rw-r--r-- 1 root root 5128 Apr 3 2018 community-web-cgi.rules
-rw-r--r-- 1 root root 4589 Apr 3 2018 community-web-client.rules
-rw-r--r-- 1 root root 254 Apr 3 2018 community-web-dos.rules
-rw-r--r-- 1 root root 1473 Apr 3 2018 community-web-iis.rules
-rw-r--r-- 1 root root 68917 Apr 3 2018 community-web-misc.rules
-rw-r--r-- 1 root root 163259 Apr 3 2018 community-web-php.rules
-rw-r--r-- 1 root root 7646 Apr 3 2018 ddos.rules
-rw-r--r-- 1 root root 64313 Apr 3 2018 deleted.rules
-rw-r--r-- 1 root root 6743 Apr 3 2018 dns.rules
-rw-r--r-- 1 root root 6296 Apr 3 2018 dos.rules
-rw-r--r-- 1 root root 1335 Apr 3 2018 experimental.rules
-rw-r--r-- 1 root root 30744 Apr 3 2018 exploit.rules
-rw-r--r-- 1 root root 4210 Apr 3 2018 finger.rules
-rw-r--r-- 1 root root 22000 Apr 3 2018 ftp.rules
-rw-r--r-- 1 root root 16482 Apr 3 2018 icmp-info.rules
-rw-r--r-- 1 root root 5352 Apr 3 2018 icmp.rules
-rw-r--r-- 1 root root 13741 Apr 3 2018 imap.rules
-rw-r--r-- 1 root root 3287 Apr 3 2018 info.rules
-rw-r--r-- 1 root root 199 Apr 3 2018 local.rules
-rw-r--r-- 1 root root 18486 Apr 3 2018 misc.rules
-rw-r--r-- 1 root root 3730 Apr 3 2018 multimedia.rules
-rw-r--r-- 1 root root 1935 Apr 3 2018 mysql.rules
-rw-r--r-- 1 root root 283854 Apr 3 2018 netbios.rules
```



Snort Rule Format

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.

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.

Syntax: Action Protocol Source IP Source port -> Destination IP Destination port (options)

Header Fields:-

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, reject, and sdrops.

Protocol: After deciding the option for action in the rule, you need to describe specific Protocol (IP, TCP, UDP, ICMP, any) on which this rule will be applicable.

Source IP: This part of header describes the sender network interface from which traffic is coming.

Source Port: This part of 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 header describes the destination network interface in which traffic is coming for establishing the connection.

Destination Port: This part of header describes the destination Port on which traffic is coming for establishing the connection.

Option Fields:

The body for rule option is usually written between circular brackets “()” that contains keywords with their argument and separated by semicolon “;” from another keyword.



There are four major categories of rule options.

General: These options contain metadata that offers information with reference to 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.”

General Rule Options (Metadata)

In this article are going to explore more about general rule option for beginners so that they can easily write a basic rule in snort rule file and able to analyst packet of their network. Metadata is part of the optional rule which basically contains additional information of about snort rule that is written with the help of some keywords and with their argument details.

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.



Let's start writing snort rule:

To check whether the Snort is logging any alerts as proposed, add a detection rule alert on IP packets in the “local.rules file”



Before writing new rules let's empty the ICMP rule file by using the following command:

```
echo "" > icmp.rules
```

```
cat icmp.rules
```

The cat command will confirm whether the file is empty. Now, let's empty the icmp-info.rules:

```
echo "" > icmp-info.rules
```

```
cat icmp-info.rules
```

```
root@ubuntu:/etc/snort/rules# echo "" > icmp.rules
root@ubuntu:/etc/snort/rules# cat icmp.rules
root@ubuntu:/etc/snort/rules# echo "" > icmp-info.rules
root@ubuntu:/etc/snort/rules# cat icmp-info.rules
root@ubuntu:/etc/snort/rules#
```

Now let's write the rule:

```
alert icmp any any -> 192.168.1.21 any (msg: "ICMP Packet found"; sid:10000001; )
```

If you observe in the image below, we have used a one-way arrow which means that snort will alert us about incoming malicious traffic:

```
alert icmp any any -> 192.168.1.21 any (msg: "ICMP Packet found"; sid:10000001; )
```

The IP (192.168.1.10) we will attack from is shown in the image shown below:



```
Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix  . :
Link-local IPv6 Address . . . . . : fe80::c454:21cc:6485:fd35%7
IPv4 Address. . . . . : 192.168.1.10
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.1.1
```

Now, we will send two packets with the following command:

```
ping -n 2 192.168.1.21
```

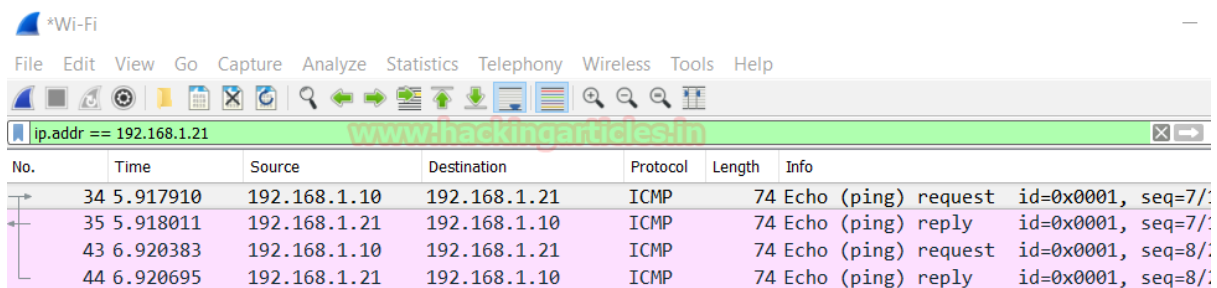
```
C:\Users\raj>ping -n 2 192.168.1.21

Pinging 192.168.1.21 with 32 bytes of data:
Reply from 192.168.1.21: bytes=32 time<1ms TTL=64
Reply from 192.168.1.21: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.1.21:
    Packets: Sent = 2, Received = 2, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\raj>
```

You can check the details of the packets that are being sent:



No.	Time	Source	Destination	Protocol	Length	Info
34	5.917910	192.168.1.10	192.168.1.21	ICMP	74	Echo (ping) request id=0x0001, seq=7/:
35	5.918011	192.168.1.21	192.168.1.10	ICMP	74	Echo (ping) reply id=0x0001, seq=7/:
43	6.920383	192.168.1.10	192.168.1.21	ICMP	74	Echo (ping) request id=0x0001, seq=8/:
44	6.920695	192.168.1.21	192.168.1.10	ICMP	74	Echo (ping) reply id=0x0001, seq=8/:



Use the following command to activate snort in order to catch the malicious packets:

```
sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i ens33
```

Here,

- A Set alert mode: fast, full, console, test or none
- q stands for Quiet, Don't show banner and status report.
- u Run snort uid as <uname> user
- g Run snort gid as <gname> group (or gid)
- c <rules> Use Rules File
- i listen on interface

And as you can see in the image below the alerts are being issued by snort:

```
root@ubuntu:~# sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i ens33
02/15-00:34:20.525109 00000001:0] ICMP Packet found 00000001:0] {Priority: 0} {ICMP} 192.168.1.10 -> 192.168.1.21
02/15-00:34:21.527735 00000001:0] ICMP Packet found 00000001:0] {Priority: 0} {ICMP} 192.168.1.10 -> 192.168.1.21
```

Now, add the following rule to see both incoming and outgoing traffic when an alert is issued:

```
alert icmp any any <> 192.168.1.21 any (msg: "ICMP Packet found"; sid:10000001; )
```

As the below image shows in this we have used '<>', it is used in order to monitor both sent and received packets when an alert is issued.

```
alert icmp any any <> 192.168.1.21 any (msg: "ICMP Packet found"; sid:10000001; )
```



Again, we will send two packets like before using the following command:

```
ping -n 192.168.1.21
```

```
C:\Users\raj>ping -n 2 192.168.1.21

Pinging 192.168.1.21 with 32 bytes of data:
Reply from 192.168.1.21: bytes=32 time<1ms TTL=64
Reply from 192.168.1.21: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.1.21:
    Packets: Sent = 2, Received = 2, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\raj>
```

And therefore, as a result, you can see both packets as shown in the image below:

```
[1:10000001:0] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.1.10 -> 192.168.1.21
[1:10000001:0] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.1.21 -> 192.168.1.10
[1:10000001:0] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.1.10 -> 192.168.1.21
[1:10000001:0] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.1.21 -> 192.168.1.10
```

Now we will apply rules on port 21, 22 and 80. This way, whenever a suspicious packet is sent to these ports, we will be notified. Following are the rules to apply to achieve the said:

```
alert tcp any any -> any 21 (msg: "FTP Packet found"; sid:10000002; )
```

```
alert tcp any any -> any 22 (msg: "SSH Packet found"; sid:10000003; )
```

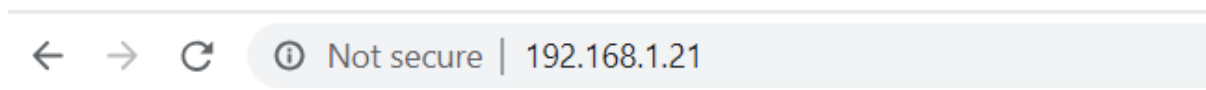
```
alert tcp any any -> any 80 (msg: "HTTP Packet found"; sid:10000004; )
```




```
alert tcp any any -> any 21 (msg: "FTP Packet found"; sid:10000002; )
alert tcp any any -> any 22 (msg: "SSH Packet found"; sid:10000003; )
alert tcp any any -> any 80 (msg: "HTTP Packet found"; sid:10000004; )
```

www.hackingarticles.in

When the packet is sent to port 80 as shown in the image:



Welcome to Hacking Articles

Snort will issue an alert of HTTP packet as its shown in the image below:

```
[1:10000004:0] HTTP Packet found [**] [Priority: 0] {TCP} 192.168.1.10:49977 -> 192.168.1.21:80
[1:10000004:0] HTTP Packet found [**] [Priority: 0] {TCP} 192.168.1.10:49977 -> 192.168.1.21:80
[1:10000004:0] HTTP Packet found [**] [Priority: 0] {TCP} 192.168.1.10:49977 -> 192.168.1.21:80
[1:10000004:0] HTTP Packet found [**] [Priority: 0] {TCP} 192.168.1.10:49977 -> 192.168.1.21:80
```

Similarly, when a data packet sent to ftp as given in the following image:

```
root@kali:~# ftp 192.168.1.21
Connected to 192.168.1.21.
220 (vsFTPd 3.0.3)
Name (192.168.1.21:root): raj
331 Please specify the password.
Password:
530 Login incorrect.
Login failed.
ftp>
```




The FTP packets will be detected and one will be notified.

```
[1:10000002:0] FTP Packet found [**] [Priority: 0] {TCP} 192.168.1.33:49304 -> 192.168.1.21:21
[1:10000002:0] FTP Packet found [**] [Priority: 0] {TCP} 192.168.1.33:49304 -> 192.168.1.21:21
[1:10000002:0] FTP Packet found [**] [Priority: 0] {TCP} 192.168.1.33:49304 -> 192.168.1.21:21
[1:10000002:0] FTP Packet found [**] [Priority: 0] {TCP} 192.168.1.33:49304 -> 192.168.1.21:21
[1:10000002:0] FTP Packet found [**] [Priority: 0] {TCP} 192.168.1.33:49304 -> 192.168.1.21:21
[1:10000002:0] FTP Packet found [**] [Priority: 0] {TCP} 192.168.1.33:49304 -> 192.168.1.21:21
[1:10000002:0] FTP Packet found [**] [Priority: 0] {TCP} 192.168.1.33:49304 -> 192.168.1.21:21
```

Again, in a similar manner, when one tries to send packets to SSH as shown in the image below:

```
root@kali:~# ssh raj@192.168.1.21
The authenticity of host '192.168.1.21 (192.168.1.21)' can't be established.
ECDSA key fingerprint is SHA256:JJEN3dWXWptI1FJcysl6z0Bc0zNNPMJdX03WQvLAmKY.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.21' (ECDSA) to the list of known hosts.
raj@192.168.1.21's password:
```

Snort will notify the administration as shown below:

```
[**] [1:10000003:0] SSH Packet found [**] [Priority: 0] {TCP} 192.168.1.33:38318 -> 192.168.1.21:22
[**] [1:10000003:0] SSH Packet found [**] [Priority: 0] {TCP} 192.168.1.33:38318 -> 192.168.1.21:22
```

This way, using snort or any other IDS one can be protected from network attacks by being notified of them in time.

Detect NMAP Scan Using Snort

Requirement

Attacker: Kali Linux (NMAP Scan)

Target: Ubuntu (Snort as IDS)

Optional: Wireshark (we have added it in our tutorial so that we can clearly confirm all incoming and outgoing packets of a network)

Identify NMAP Ping Scan



As we know any attacker will start the attack by identifying host status by sending ICMP packet using ping scan. Therefore, be smart and add a rule in snort which will analyse NMAP Ping scan when someone tries to scan your network for identifying a live host of a network.

Execute given below command in ubuntu's terminal to open snort local rule file in text editor.

```
sudo gedit /etc/snort/rules/local.rules
```

Now add given below line which will capture the incoming traffic coming on 192.168.1.105(ubuntu IP) network for ICMP protocol.

```
alert icmp any any -> 192.168.1.105 any (msg: "NMAP ping sweep Scan";  
dsize:0;sid:10000004; rev: 1;)
```

```
sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
```

Turn on IDS mode of snort by executing given below command in terminal:

Now using attacking machine execute given below command to identify the status of the target machine i.e., host is UP or Down.

```
nmap -sP 192.168.1.105 --disable-arp-ping
```

If you will execute above command without parameter "disable arp-ping" then will work as default ping sweep scan which will send arp packets in spite of sending ICMP on targets network and maybe snort not able to capture NMAP Ping scan in that scenario, therefore we had use parameter "disable arp-ping" in the above command.

```
root@kali:~# nmap -sP 192.168.1.105 --disable-arp-ping ↵  
  
Starting Nmap 7.60 ( https://nmap.org ) at 2017-12-20 11:38 EST  
Nmap scan report for 192.168.1.105  
Host is up (0.0061s latency).  
MAC Address: 00:0C:29:6B:71:A7 (VMware)  
Nmap done: 1 IP address (1 host up) scanned in 0.19 seconds
```

As I had declaimed above why we are involving Wireshark in this tutorial so that you can clearly see the packet sends from attacker network to targets network. Hence in given below image, you can notice ICMP request packet along with ICMP reply packets. These both are parts of network traffic.



ip.addr == 192.168.1.105							Expression...	+
No.	Time	Source	Destination	Protoc	Length	Info		
→ 516	3.7366...	192.168.1.104	192.168.1.105	ICMP	42	Echo (ping) request id=0xa		
517	3.7367...	192.168.1.104	192.168.1.105	TCP	58	37516 → 443 [SYN] Seq=0 Win		
518	3.7367...	192.168.1.104	192.168.1.105	TCP	54	37516 → 80 [ACK] Seq=1 Ack=		
519	3.7367...	192.168.1.104	192.168.1.105	ICMP	54	Timestamp request id=0xa		
— 520	3.7369...	192.168.1.105	192.168.1.104	ICMP	60	Echo (ping) reply id=0xa		
521	3.7369...	192.168.1.105	192.168.1.104	TCP	60	443 → 37516 [SYN, ACK] Seq=		
522	3.7369...	192.168.1.104	192.168.1.105	TCP	54	37516 → 443 [RST] Seq=1 Win		
523	3.7370...	192.168.1.105	192.168.1.104	TCP	60	80 → 37516 [RST] Seq=1 Win=		
524	3.7370...	192.168.1.105	192.168.1.104	ICMP	60	Timestamp reply id=0xa		

Come back to over your target machine where snort is capturing all in-coming traffic. Here, you will observe that it is generating an alert for NMAP Ping Sweep scan. Hence, you can block the attacker's IP to protect your network from further scanning.

```
sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
```

```
pentest@ubuntu:~$ sudo snort -A console -q -u snort -g snort -c /etc/snort/snort
.conf -i eth0
12/22-01:52:18.051911  [**] [1:10000004:1] NMAP ping sweep Scan [**] [Priority:
0] {ICMP} 192.168.1.104 -> 192.168.1.105
```

Identify NMAP TCP Scan

Now in order to connect with the target network, an attacker may go for networking enumeration either using TCP Protocol or UDP protocol. Let's assume attacker may choose TCP scanning for network enumeration then in that situation we can apply the following rule in snort local rule file.

```
alert tcp any any -> 192.168.1.105 22 (msg: "NMAP TCP Scan";sid:10000005; rev:2; )
```

Above rule is only applicable for port 22 so if you want to scan any other port then replace 22 from the port you want to scan or else you can also use "any" to analysis all ports. Enable the NIDS mode of snort as done above.



Now again using the attacker machine execute the given below command for TCP scan on port 22.

```
nmap -sT -p22 192.168.1.105
```

```
root@kali:~# nmap -sT -p22 192.168.1.105 ↩️
Starting Nmap 7.60 ( https://nmap.org ) at 2017-12-20 11:46 EST
Nmap scan report for 192.168.1.105
Host is up (0.00065s latency).
www.hackingarticles.in
PORT      STATE SERVICE
22/tcp    open  ssh
MAC Address: 00:0C:29:6B:71:A7 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.68 seconds
```

From the image given below, you can observe Wireshark has captured TCP packets from 192.168.1.104 to 192.168.1.105

ip.addr == 192.168.1.105							
No.	Time	Source	Destination	Protoc	Length	Info	
453	4.0527...	192.168.1.104	192.168.1.105	TCP	74	51338 → 22 [SYN] Seq=0 Win=	
454	4.0533...	192.168.1.105	192.168.1.104	TCP	74	22 → 51338 [SYN, ACK] Seq=0	
455	4.0534...	192.168.1.104	192.168.1.105	TCP	66	51338 → 22 [ACK] Seq=1 Ack=	
456	4.0549...	192.168.1.104	192.168.1.105	TCP	66	51338 → 22 [RST, ACK] Seq=1	

Here you can confirm that our snort is absolutely working when the attacker is scanning port 22 using nmap TCP scan and it is showing attacker's IP from where traffic is coming on port 22. Hence you can block this IP to protect your network from further scanning.

```
pentest@ubuntu:~$ sudo snort -A console -q -u snort -g snort -c /etc/snort/snort
.conf -i eth0 ↩️
12/20-08:46:53.932278  [**] [1:1000005:2] NMAP TCP Scan [**] [Priority: 0] {TCP}
192.168.1.104:51338 -> 192.168.1.105:22
12/20-08:46:53.932991  [**] [1:1000005:2] NMAP TCP Scan [**] [Priority: 0] {TCP}
192.168.1.104:51338 -> 192.168.1.105:22
12/20-08:46:53.934417  [**] [1:1000005:2] NMAP TCP Scan [**] [Priority: 0] {TCP}
192.168.1.104:51338 -> 192.168.1.105:22
```

Identify NMAP XMAS Scan



As we know that TCP communication follows three-way handshake to established TCP connection with target machine but sometimes instead of using SYN, SYN/ACK, ACK flag attacker chooses XMAS scan to connect with the target by sending data packets through Fin, PSH & URG flags.

Let assume attacker may choose XMAS scanning for network enumeration then in that situation we can apply the following rule in snort local rule file.

```
alert tcp any any -> 192.168.1.105 22 (msg:"Nmap XMAS Tree Scan"; flags:FPU; sid:1000006; rev:1; )
```

Again, above rule is only applicable for port 22 which will listen for incoming traffic when packets come from Fin, PSH & URG flags. So, if you want to scan any other port then replace 22 from the port you want to scan else you can also use “any” to analysis all ports. Enable the NIDS mode of snort as done above.

Now again using the attacker machine execute the given below command for XMAS scan on port 22.

```
nmap -sX -p22 192.168.1.105
```

```
root@kali:~# nmap -sX -p22 192.168.1.105
Starting Nmap 7.60 ( https://nmap.org ) at 2017-12-20 11:54 EST
Nmap scan report for 192.168.1.105
Host is up (0.00047s latency).

PORT      STATE      SERVICE
22/tcp    open|filtered ssh
MAC Address: 00:0C:29:6B:71:A7 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.43 seconds
```

From given below image you can observe that Wireshark is showing 2 packets from attacker machine to target machine has been sent using FIN, PSH, URG flags.

ip.addr == 192.168.1.105							Expression...	+
No.	Time	Source	Destination	Protoc	Length	Info		
590	8.3603...	192.168.1.104	192.168.1.105	TCP	54	38157 → 22 [FIN, PSH, URG]		
592	8.4619...	192.168.1.104	192.168.1.105	TCP	54	38158 → 22 [FIN, PSH, URG]		



Come back to over your target machine where snort is capturing all incoming traffic here you will observe that it is generating an alert for NMAP XMAP scan. Hence you can block the attacker's IP to protect your network from further scanning.

```
sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
```

```
pentest@ubuntu:~$ sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
12/20-08:54:09.407169  [**] [1:1000006:1] Nmap XMAS Tree Scan [**] [Priority: 0]
{TCP} 192.168.1.104:38157 -> 192.168.1.105:22
12/20-08:54:09.508960  [**] [1:1000006:1] Nmap XMAS Tree Scan [**] [Priority: 0]
{TCP} 192.168.1.104:38158 -> 192.168.1.105:22
```

Identify NMAP FIN Scan

Instead of using SYN, SYN/ACK and ACK flag to established TCP connection with the target machine may attacker choose FIN scan to connect with the target by sending data packets through Fin flags only.

Let assume attacker may choose FIN scanning for network enumeration then in that situation we can apply the following rule in snort local rule file.

```
alert tcp any any -> 192.168.1.105 22 (msg:"Nmap FIN Scan"; flags:F;
sid:1000008; rev:1;)
```

Again, above rule is only applicable for port 22 which will listen for incoming traffic when packets come from Fin Flags. So, if you want to scan any other port then replace 22 from the port you want to scan else you can also use "any" to analysis all ports. Enable NIDS mode of snort as done above.

Now again using the attacker machine execute the given below command for FIN scan on port 22.

```
nmap -sF -p22 192.168.1.105
```



```
root@kali:~# nmap -sF -p22 192.168.1.105

Starting Nmap 7.60 ( https://nmap.org ) at 2017-12-20 12:00 EST
Nmap scan report for 192.168.1.105
Host is up (0.00018s latency).
PORT      STATE      SERVICE
22/tcp    open|filtered ssh
MAC Address: 00:0C:29:6B:71:A7 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.48 seconds
```

From given below image you can observe that Wireshark is showing 2 packets from attacker machine to target machine has been sending using FIN flags.

No.	Time	Source	Destination	Protoc	Length	Info
723	8.6023...	192.168.1.104	192.168.1.105	TCP	54	39392 → 22 [FIN] Seq=1 Win=
731	8.7031...	192.168.1.104	192.168.1.105	TCP	54	39393 → 22 [FIN] Seq=1 Win=
5787	42.777...	192.168.1.105	224.0.0.251	IGMP	60	Membership Report group 224

Come back to over your target machine where snort is capturing all incoming traffic here you will observe that it is generating an alert for NMAP FIN scan. Hence you can block the attacker's IP to protect your network from further scanning.

```
sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
```

```
pentest@ubuntu:~$ sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
12/20-09:00:12.048515  [**] [1:1000008:1] Nmap FIN Scan [**] [Priority: 0] {TCP}
192.168.1.104:39392 -> 192.168.1.105:22
12/20-09:00:12.149434  [**] [1:1000008:1] Nmap FIN Scan [**] [Priority: 0] {TCP}
192.168.1.104:39393 -> 192.168.1.105:22
```

Identify NMAP NULL Scan

Instead of using SYN, SYN/ACK and ACK flag to established TCP connection with the target machine may attacker choose NULL scan to connect with the target by sending data packets through NONE flags only.



Let assume attacker may choose NULL scanning for network enumeration then in that situation we can apply the following rule in snort local rule file.

```
alert tcp any any -> 192.168.1.105 22 (msg:"Nmap NULL Scan"; flags:0;  
sid:1000009; rev:1; )
```

Again, above rule is only applicable for port 22 which will listen for incoming traffic when packets come from NONE Flags. So, if you want to scan any other port then replace 22 from the port you want to scan else you can also use “any” to analysis all ports. Enable the NIDS mode of snort as done above.

Now again using the attacker machine execute the given below command for the NULL scan on port 22.

```
nmap -sN -p22 192.168.1.105
```

```
root@kali:~# nmap -sN -p22 192.168.1.105 ↩  
  
Starting Nmap 7.60 ( https://nmap.org ) at 2017-12-20 12:03 EST  
Nmap scan report for 192.168.1.105  
Host is up (0.00017s latency).  
  
PORT      STATE      SERVICE  
22/tcp    open|filtered ssh  
MAC Address: 00:0C:29:6B:71:A7 (VMware)  
  
Nmap done: 1 IP address (1 host up) scanned in 0.47 seconds
```

From given below image you can observe that Wireshark is showing 2 packets from attacker machine to target machine has been sending using NONE flags.

ip.addr == 192.168.1.105							
No.	Time	Source	Destination	Protoc	Length	Info	
889	14.701...	192.168.1.104	192.168.1.105	TCP	54	47435 → 22	[<None>] Seq=1 W
896	14.801...	192.168.1.104	192.168.1.105	TCP	54	47436 → 22	[<None>] Seq=1 W

Come back to over your target machine where snort is capturing all incoming traffic here you will observe that it is generating an alert for NMAP Null scan. Hence you can block the attacker's IP to protect your network from further scanning.

```
sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
```




```
pentest@ubuntu:~$ sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
12/20-09:03:15.568255  [**] [1:1000009:1] Nmap NULL Scan [**] [Priority: 0] {TCP
} 192.168.1.104:47435 -> 192.168.1.105:22
12/20-09:03:15.668675  [**] [1:1000009:1] Nmap NULL Scan [**] [Priority: 0] {TCP
} 192.168.1.104:47436 -> 192.168.1.105:22
```

Identify NMAP UDP Scan

In order to Identify open UDP port and running services attacker may choose NMAP UDP scan to establish a connection with target machine for network enumeration then in that situation, we can apply the following rule in snort local rule file.

```
alert udp any any -> 192.168.1.105 any ( msg:"Nmap UDP Scan"; sid:1000010; rev:1; )
```

Again, above rule is applicable for every UDP port which will listen for incoming traffic when packets are coming over any UDP port, so if you want to capture traffic for any particular UDP port then replace “any” from that specific port number as done above. Enable the NIDS mode of snort as done above.

Now again using the attacker machine execute the given below command for a NULL scan on port 22.

```
nmap -sU -p68 192.168.1.105
```

```
root@kali:~# nmap -sU -p68 192.168.1.105
Starting Nmap 7.60 ( https://nmap.org ) at 2017-12-22 05:06 EST
Nmap scan report for 192.168.1.105
Host is up (0.00041s latency).
PORT      STATE      SERVICE
68/udp    open|filtered dhcpc
MAC Address: 00:0C:29:6B:71:A7 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.47 seconds
```



From given below image you can observe that Wireshark is showing 2 packets from attacker machine to target machine has been sending over UDP Port.

ip.addr == 192.168.1.105						
No.	Time	Source	Destination	Protoc	Length	Info
12	19.111...	192.168.1.105	224.0.0.251	IGMP	60	Membership Report group 224
15	19.182...	192.168.1.104	192.168.1.105	UDP	42	49451 → 68 Len=0
16	19.282...	192.168.1.104	192.168.1.105	UDP	42	49452 → 68 Len=0

Come back to over your target machine where snort is capturing all incoming traffic here you will observe that it is generating an alert for NMAP UDP scan. Hence you can block the attacker's IP to protect your network from further scanning.

```
pentest@ubuntu:~$ sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
12/22-02:06:19.669158  [**] [1:1000010:1] NMAP UDP scan [**] [Priority: 0] {UDP}
192.168.1.104:49451 -> 192.168.1.105:68
12/22-02:06:19.769495  [**] [1:1000010:1] NMAP UDP scan [**] [Priority: 0] {UDP}
192.168.1.104:49452 -> 192.168.1.105:68
```

Detect SQL Injection Attack using Snort IDS

Requirement

IDS: Snort (Ubuntu)

Web application: Dhakkan

You can configure your own web server if you want.

Identify Error Based SQL Injection

As we know in Error based SQL injections the attacker uses **single quotes** (') or **double quotes** (") to break down SQL query for identify its vulnerability. Therefore, be smart and add a rule in snort which will analyst Error based SQL injection on the server when someone try to execute SQL query in your network for unprivileged access of database.



Execute given below command in ubuntu's terminal to open snort local rule file in text editor.

```
sudo gedit /etc/snort/rules/local.rules
```

Now add given below line which will capture the incoming traffic coming on any network IP via port 80.

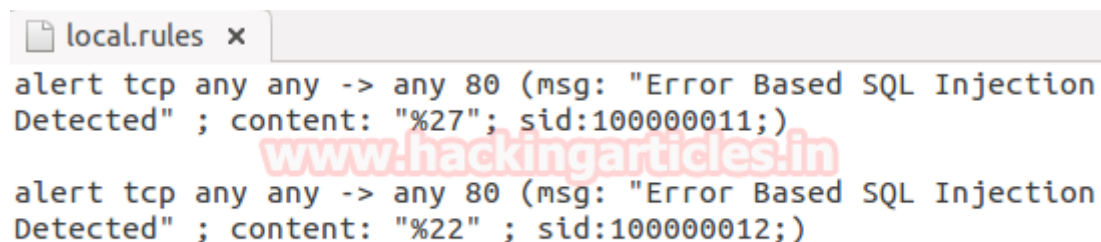
```
alert tcp any any -> any 80 (msg: "Error Based SQL Injection Detected";  
content: "%27" ; sid:100000011; )
```

```
alert tcp any any -> any 80 (msg: "Error Based SQL Injection Detected";  
content: "%22" ; sid:100000012; )
```

If you read above rule you can notice that I had applied filter for content “%27” and %22 is URL encoded format use in browser for single quotes (‘) and double quotes (“) respectively at the time of execution of URL.

Turn on IDS mode of snort by executing given below command in terminal:

```
sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
```



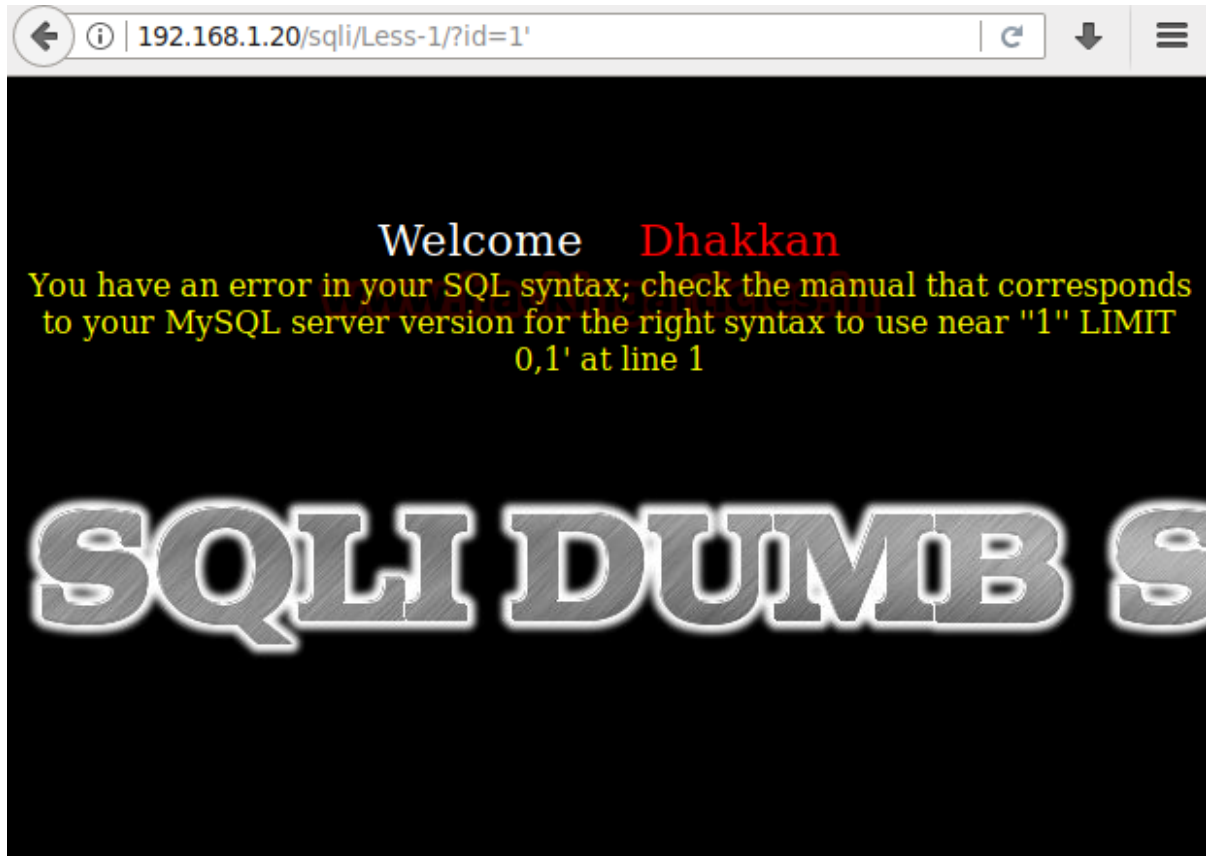
```
local.rules x  
alert tcp any any -> any 80 (msg: "Error Based SQL Injection  
Detected" ; content: "%27"; sid:100000011;)  
alert tcp any any -> any 80 (msg: "Error Based SQL Injection  
Detected" ; content: "%22" ; sid:100000012;)
```

Now test your above rule by making Error based sql injection attack on web application “Dhakkan”, therefore open the server IP in web browser and use single quotes (‘) for identify SQL injection vulnerability as shown below.

```
192.168.1.20/sqli/Less-1/?id=1'
```



Now when attacker will execute malicious quotes in browser for testing Error Base SQL injection then the IDS of the network should also capture this content and will generate the alert.



As per our prediction from given image you can observe the snort has generated alert for Error Based sql injection when capture malicious quotes.

So, when the network admin get alert from IDS on the basis of it, he can take action against attacking IP, as shown in given image the malicious traffic is coming form 192.168.1.21 on port 80.

```
zed@ubuntu:/etc/snort/rules$ sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
01/11-03:21:25.445355  [**] [1:100000011:0] Error Based SQL Injection Detected [
**] [Priority: 0] {TCP} 192.168.1.21:37486 -> 192.168.1.20:80
```

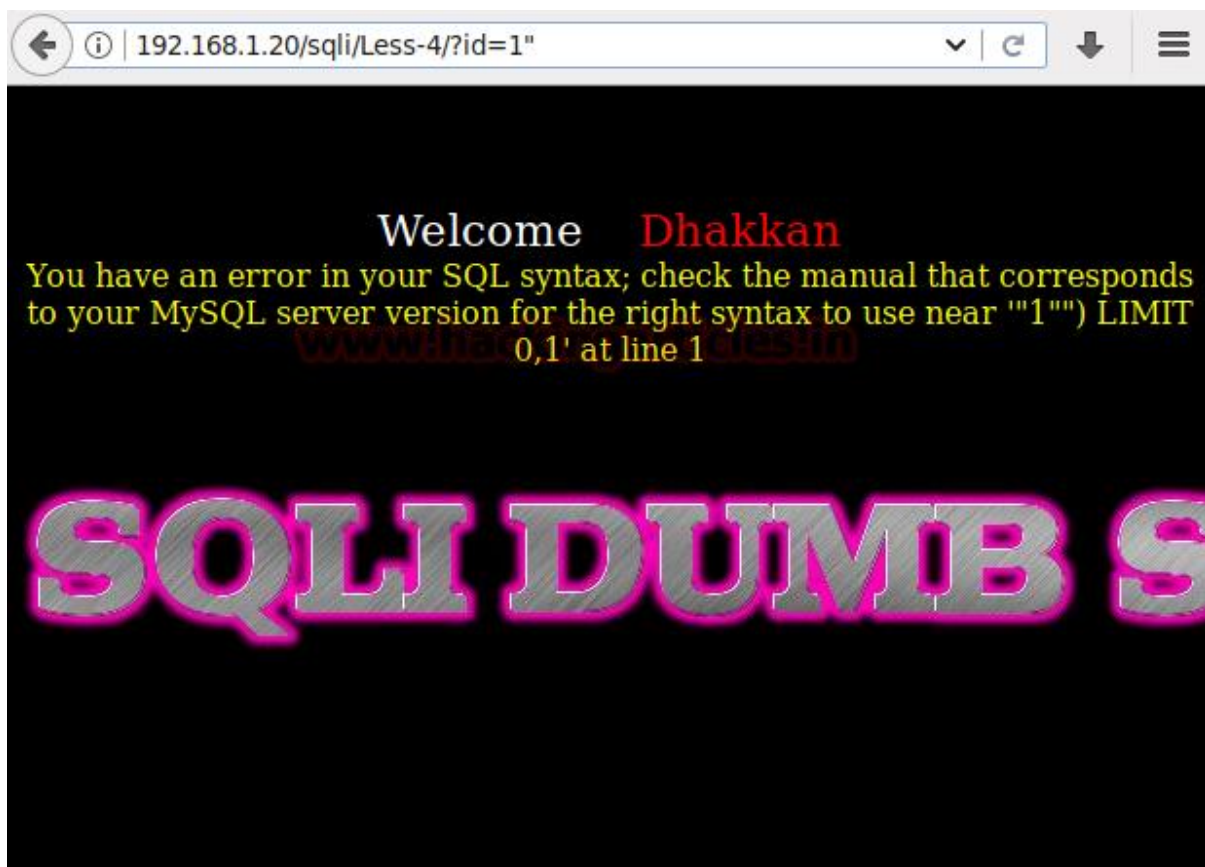
Testing Double Quotes Injection



Now again open the server IP in web browser and use double quotes (") for identify SQL injection vulnerability as shown below.

```
192.168.1.20/sqli/Less-4/?id=1"
```

Now when attacker will execute malicious quotes in browser for testing Double quotes SQL injection then the IDS of the network should also capture this content and will generate the alert.



From given image you can observe the snort has generated alert for Error Based sql injection when capture malicious quotes.

So, when the network admin get alert from IDS on the basis of it, he can take action against attacking IP, as shown in given image the malicious traffic is coming from 192.168.1.21 on port 80.

```
zed@ubuntu:/etc/snort/rules$ sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
01/11-03:22:14.485463  [**] [1:100000012:0] Error Based SQL Injection Detected [
**] [Priority: 0] {TCP} 192.168.1.21:37488 -> 192.168.1.20:80
```



Boolean Based SQL Injection

As we know in Boolean based SQL injections the attacker uses **AND /OR operators** where attacker will try to confirm if the database is vulnerable to Boolean SQL Injection by evaluating the results of various queries which return either TRUE or FALSE.

Now add a rule in snort which will analyse Boolean based SQL injection on the server when someone try to execute SQL query in your network for unprivileged access of database. Here **I had applied filter for content “and” & “or” to be captured. Here nocase denotes not case sensitive it can be as AND/and, OR/or.**

```
alert tcp any any -> any 80 (msg: "AND SQL Injection Detected"; content: "and" ; nocase; sid:100000060; )
```

```
alert tcp any any -> any 80 (msg: "OR SQL Injection Detected"; content: "or" ; nocase; sid:100000061; )
```

Turn on IDS mode of snort by executing given below command in terminal:

```
sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
```

local.rules x

```
alert tcp any any -> any 80 (msg: "AND SQL Injection Detected" ; content: "and"; nocase ; sid:100000060;)
```

```
alert tcp any any -> any 80 (msg: "OR SQL Injection Detected" ; content: "or"; nocase ; sid:100000061;)
```

Again, open the server IP in web browser and use AND operator for identify Boolean SQL injection vulnerability as shown below.

```
192.168.1.20/sqli/Less-8/?id=1' AND 1=1 --+
```



Now when attacker will execute malicious quotes in browser for testing Boolean Base SQL injection then the IDS of the network should also capture this content and will generate the alert.



Testing OR Operator

As per our calculation from given image you can observe the snort has generated alert for Boolean Based sql injection when captured content AND.

So, when the network admin get alert from IDS on the basis of it, he can take action against attacking IP, as shown in given image the malicious traffic is coming form 192.168.1.21 on port 80.

```
zed@ubuntu:/etc/snort/rules$ sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
01/11-03:25:02.114714  [**] [1:100000060:0] AND SQL Injection Detected [**] [Priority: 0] {TCP} 192.168.1.21:37492 -> 192.168.1.20:80
```




Again, open the server IP in web browser and use OR operator to identify Boolean SQL injection vulnerability as shown below.

```
192.168.1.20/sqli/Less-8/?id=1' OR 1=1 --+
```

Now when attacker will execute malicious quotes in browser for testing Boolean Base SQL injection then the IDS of the network should also capture this content and will generate the alert.



As per our calculation from given image you can observe the snort has generated alert for Boolean Based sql injection when captured content OR.

So, when the network admin get alert from IDS on the basis of it, he can take action against attacking IP, as shown in given image the malicious traffic is coming form 192.168.1.21 on port 80.

```
zed@ubuntu:/etc/snort/rules$ sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
01/11-03:11:36.167866  [**] [1:100000061:0] OR SQL Injection Detected [**] [Priority: 0] {TCP} 192.168.1.21:37474 -> 192.168.1.20:80
```




Encoded AND/OR

Similarly, in given below rule I had applied filter for content “%26%26” and “%7c%7c” are URL encoded format use in browser for && and || respectively at the time of execution of URL.

```
alert tcp any any -> any 80 (msg: "AND SQL Injection Detected"; content: "and" ; nocase; sid:100000008; )
```

```
alert tcp any any -> any 80 (msg: "OR SQL Injection Detected"; content: "or" ; nocase; sid:100000009; )
```

Turn on IDS mode of snort by executing given below command in terminal:

```
sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
```

```
local.rules x
alert tcp any any -> any 80 (msg: "AND SQL Injection Detected" ;
content: "%26%26"; sid:100000008;)
alert tcp any any -> any 80 (msg: "OR SQL Injection Detected" ;
content: "%7C%7C"; sid:100000009;)
```

Again, open the server IP in web browser and use **&& operator** for identify Boolean SQL injection vulnerability as shown below.

```
192.168.1.20/sqli/Less-25/?id=1' %26%26 1==1 --+
```



Now when attacker will execute malicious quotes in browser for testing Boolean Base SQL injection then the IDS of the network should also capture this content and will generate the alert.



As per our calculation from given image you can observe the snort has generated alert for Boolean Based sql injection when captured content **%26%26**.

So, when the network admin get alert from IDS on the basis of it, he can take action against attacking IP, as shown in given image the malicious traffic is coming form 192.168.1.21 on port 80.

```
zed@ubuntu:/etc/snort/rules$ sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
01/11-03:00:28.491295  [**] [1:10000008:0] AND SQL Injection Detected [**] [Priority: 0] {TCP} 192.168.1.21:37454 -> 192.168.1.20:80
```

Testing Encoded OR Operator



Again, open the server IP in web browser and use || **operator** for identify Boolean SQL injection vulnerability as shown below.

```
192.168.1.20/sqli/Less-25/?id=1' %7C%7C 1==1 --+
```

Now when attacker will execute malicious quotes in browser for testing Boolean Base SQL injection then the IDS of the network should also capture this content and will generate the alert.



As per our calculation from given image you can observe the snort has generated alert for Boolean Based sql injection when captured content %7C %7C.

So, when the network admin get alert from IDS on the basis of it, he can take action against attacking IP, as shown in given image the malicious traffic is coming form 192.168.1.21 on port 80.

```
zed@ubuntu:/etc/snort/rules$ sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
01/11-03:01:34.669653  [*] [1:10000009:0] OR SQL Injection Detected [**] [Priority: 0] {TCP} 192.168.1.21:37456 -> 192.168.1.20:80
```



Identify Form Based SQL Injection

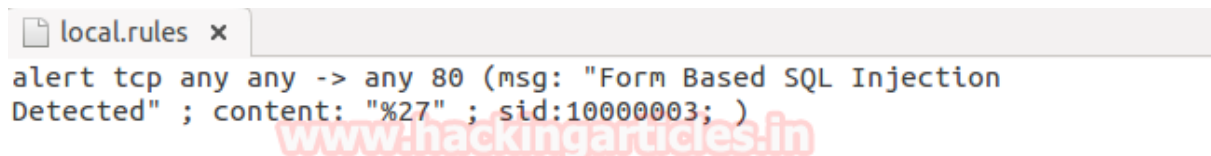
The Form Based SQL injection also known as “Post Error based SQL injection” because the attacker executes malicious quotes inside Login form of a web page that contains text field for username and password to login inside web server.

Therefore, now add a rule in snort which will analyst Form based SQL injection on the server when someone try to execute SQL query in your network for unprivileged access of database.

```
alert tcp any any -> any 80 (msg: "Form Based SQL Injection Detected";  
content: "%27" ; sid:1000003; )
```

If you read above rule you can notice that I had applied filter for content “%27” to be captured; turn on IDS mode of snort by executing given below command in terminal:

```
sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
```



```
local.rules x  
alert tcp any any -> any 80 (msg: "Form Based SQL Injection  
Detected" ; content: "%27" ; sid:1000003; )
```

I had used single quotes (‘) to break the query inside the text field of username then click on **submit**.

Username: ‘

From the given screenshot you can see we have **got error message (in blue color)** which means the database is vulnerable to SQL injection.

For more detail on Form Based SQL injection read our [previous](#) article.

Now when attacker will execute malicious quotes in browser for testing Form Base SQL injection then the IDS of the network should also capture this content and will generate the alert.



As per our prediction from given image you can observe the snort has generated alert for Form Based sql injection when capture malicious quotes.

So, when the network admin get alert from IDS on the basis of it, he can take action against attacking IP, as shown in given image the malicious traffic is coming form 192.168.1.21 on port 80.

```
zed@ubuntu:/etc/snort/rules$ sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
01/11-00:30:22.916255  [**] [1:10000003:0] Form Based SQL Injection Detected [**]
[Priority: 0] {TCP} 192.168.1.21:37274 -> 192.168.1.20:80
```

Identify Order by SQL Injection

In order to identify number of columns in database the un-trusted user may use **order by** clause which will arrange the result set in ascending or descending order of the columns used in the query.

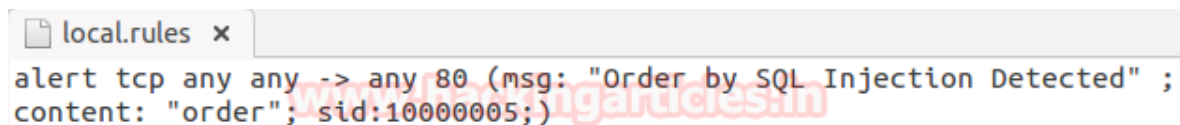


Now add a rule in snort which will analyse order by SQL injection on the server when someone tries to execute SQL query in your network for unprivileged access of database. Here again **that I had applied filter for content "order" to be captured.**

```
alert tcp any any -> any 80 (msg: "Order by SQL Injection"; content: "order" ; sid:1000005; )
```

Turn on IDS mode of snort by executing given below command in terminal:

```
sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
```

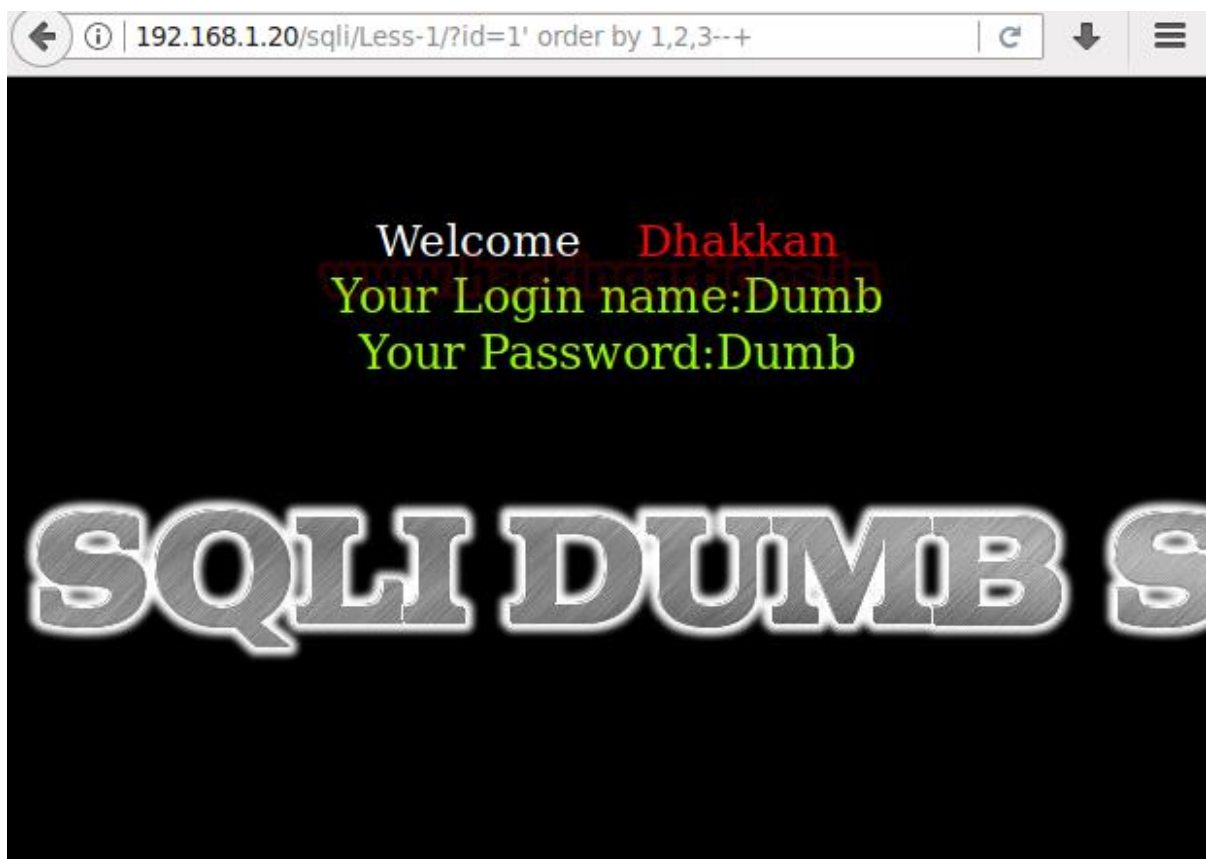


```
local.rules x
alert tcp any any -> any 80 (msg: "Order by SQL Injection Detected" ;
content: "order"; sid:1000005;)
```

Now again open the server IP in web browser and use string order by to identify column of database as shown below.

```
192.168.1.20/sqli/Less-1/?id=1' order by 1,2,3 --+
```

Now when attacker will execute malicious string in browser for testing order by SQL injection then the IDS of the network should also capture this content and will generate the alert



As per our prediction from given image you can observe the snort has generated alert for order by sql injection when capture malicious string.

So, when the network admin get alert from IDS on the basis of it, he can take action against attacking IP, as shown in given image the malicious traffic is coming form 192.168.1.21 on port 80.

```
zed@ubuntu:/etc/snort/rules$ sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
01/11-01:48:06.081322  [**] [1:10000005:0] Order By SQL Injection Detected [**]
[Priority: 0] {TCP} 192.168.1.21:37400 -> 192.168.1.20:80
```

Identify Union Based SQL Injection

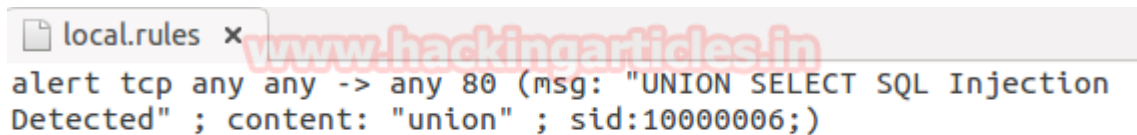
We all know in Error base SQL injection attacker may use the UNION operator to combine the result-set of two or more SELECT statements. Therefore, add a rule in snort which will analyst Union select SQL injection on the server when someone try to execute SQL query in your network for unprivileged access of database. Here again **that I had applied filter for content “union” to be captured.**



```
alert tcp any any -> any 80 (msg: "UNION SELECT SQL Injection"; content: "union"
; sid:1000006; )
```

Turn on IDS mode of snort by executing given below command in terminal:

```
sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
```



```
local.rules x
alert tcp any any -> any 80 (msg: "UNION SELECT SQL Injection
Detected" ; content: "union" ; sid:1000006;)
```

Now again open the server IP in web browser and use string order by for identify column of database as shown below.

```
192.168.1.20/sqli/Less-1/?id=-1' union select 1,2,3 --+
```

Now when attacker will execute malicious string in browser for testing Union select SQL injection then the IDS of the network should also capture this content and will generate the alert.



As per our prediction from given image you can observe the snort has generated alert for union select sql injection when capture malicious string.

So, when the network admin get alert from IDS on the basis of it, he can take action against attacking IP, as shown in given image the malicious traffic is coming form 192.168.1.21 on port 80.

```
zed@ubuntu:/etc/snort/rules$ sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
01/11-02:56:25.682946  [**] [1:10000006:0] UNION SELECT SQL Injection Detected [
**] [Priority: 0] {TCP} 192.168.1.21:37438 -> 192.168.1.20:80
```

Conclusion

Hence, one can make use of these commands as a cybersecurity professional to assess vulnerabilities on systems and keep these systems away from threat.

References

- <https://www.hackingarticles.in/comprehensive-guide-on-snort-part-1/>
- <https://www.hackingarticles.in/detect-nmap-scan-using-snort/>
- <https://www.hackingarticles.in/detect-sql-injection-attack-using-snort-ids/>