Snort

open source IPS controlled by cisco , invented in 1989 , uses a series of rules that help define malicious network activity and uses those rules to find packets that match against them and generates alerts for users

Snort modes

1- sniffer mode

read network packets and display them on the console , this is very similar to `tcpdump` or Wireshark with a bit more functionality

```
#only display the traffic
sudo snort -i <interface>

09/23-09:44:37.838793 192.168.1.4:33782 -> 93.184.215.14:80
TCP TTL:64 TOS:0x0 ID:5453 IpLen:20 DgmLen:40 DF
```

```
***A**** Seq: 0x3C619323  Ack: 0x36FC12  Win: 0xFAF0  TcpLen: 2
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

09/23-09:44:37.839038 192.168.1.4:33782 -> 93.184.215.14:80
TCP TTL:64 TOS:0x0 ID:5454 IpLen:20 DgmLen:114 DF
***AP*** Seq: 0x3C619323  Ack: 0x36FC12  Win: 0xFAF0  TcpLen: 2
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=


#display the traffic with more headers
sudo snort -i <interface> -e

09/23-09:45:39.064366 08:00:27:A8:E9:8C -> 52:54:00:12:35:00 ty
192.168.1.4:53396 -> 93.184.215.14:80 TCP TTL:64 TOS:0x0 ID:471
***A**** Seq: 0x9CCCA20B  Ack: 0x38172A  Win: 0xFAF0  TcpLen: 2
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

09/23-09:45:39.064750 08:00:27:A8:E9:8C -> 52:54:00:12:35:00 ty
192.168.1.4:53396 -> 93.184.215.14:80 TCP TTL:64 TOS:0x0 ID:471
***AP*** Seq: 0x9CCCA20B  Ack: 0x38172A  Win: 0xFAF0  TcpLen: 2
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=


#display the traffic with data in hexa and ascii with -d or -X
sudo snort -i <interface> -d

9/23-09:47:47.990180 192.168.1.4:52094 -> 93.184.215.14:80
TCP TTL:64 TOS:0x0 ID:2655 IpLen:20 DgmLen:114 DF
***AP*** Seq: 0x14ADB86F  Ack: 0x393287  Win: 0xFAF0  TcpLen: 2
47 45 54 20 2F 20 48 54 54 50 2F 31 2E 31 0D 0A  GET / HTTP/1.1
48 6F 73 74 3A 20 65 78 61 6D 70 6C 65 2E 63 6F  Host: example.
6D 0D 0A 55 73 65 72 2D 41 67 65 6E 74 3A 20 63  m..User-Agent:
75 72 6C 2F 38 2E 35 2E 30 0D 0A 41 63 63 65 70  url/8.5.0..Acce
74 3A 20 2A 2F 2A 0D 0A 0D 0A                    t: */*....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
```

```
WARNING: No preprocessors configured for policy 0.
09/23-09:47:48.110310 93.184.215.14:80 -> 192.168.1.4:52094
TCP TTL:255 TOS:0x0 ID:5248 IpLen:20 DgmLen:40
***A**** Seq: 0x393287  Ack: 0x14ADB8B9  Win: 0x7FB6  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

WARNING: No preprocessors configured for policy 0.
09/23-09:47:48.126605 93.184.215.14:80 -> 192.168.1.4:52094
TCP TTL:255 TOS:0x0 ID:5249 IpLen:20 DgmLen:1500
***A**** Seq: 0x393287  Ack: 0x14ADB8B9  Win: 0x7FB6  TcpLen: 20
48 54 54 50 2F 31 2E 31 20 32 30 30 20 4F 4B 0D  HTTP/1.1 200 OK
0A 41 67 65 3A 20 33 30 32 37 32 34 0D 0A 43 61  .Age: 302724..Ca
63 68 65 2D 43 6F 6E 74 72 6F 6C 3A 20 6D 61 78  che-Control: ma
```

> 🚩all these packets are tested using traffic with curl tool

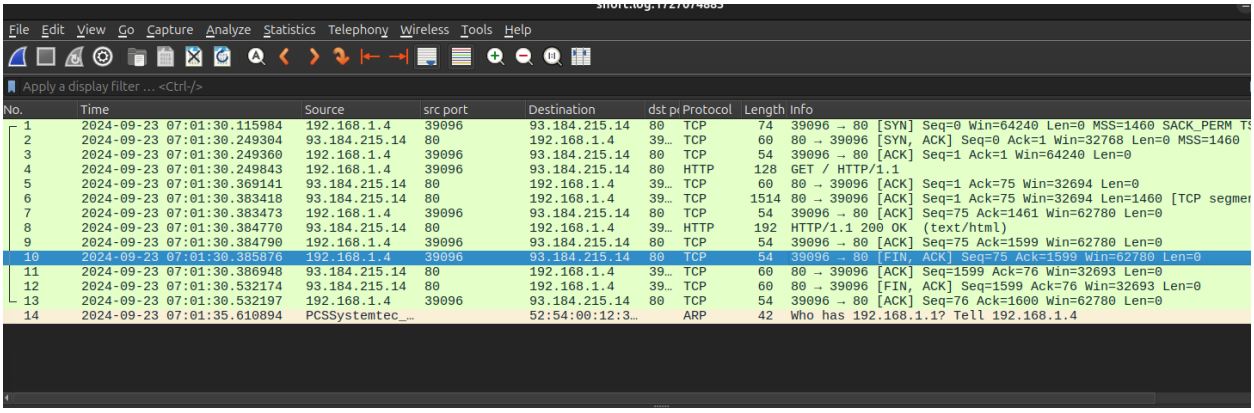
2- packet logger mode

like `tcpdump` also , snort c an log the packets into files on a disk ,this is useful for us if we want to do offline analysis later on such as `tcpdump`

```
sudo snort -i enp0s3 -l /vr/log/snort  #this is the default path

#create a separate folder for snort logging packets
mkdir logs
cd logs

#run snort
sudo snort -i enp0s3 -l .
```

```
#read the packet with snort
sudo snor -r <file name>
```

or read it with Wireshark



3- network intrusion detection and prevention mode

this where snort actively monitors the network traffic against a defined rules and can take actions like generate alerts or block or drop traffic top to prevent potential intrusion

these rules could be defined through community or threat intelligence capability or developed in house with custom rules which we are going to take a look at it

> 💡so rules really are the main core of how snort operate

Installation

```
sudo apt install snort

ctrl + shift + t -> to open new terminal amnd get your current s

sniort --version
```

```
   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.20 GRE (Build 82)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.c
           Copyright (C) 2014-2022 Cisco and/or its affiliates.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.10.4 (with TPACKET_V3)
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.3

#explain the libraries used with snort
libpcap  -> gives snort capability to inspect and analyze the tr
used this library from usingtxpdump and wiireshark

PCRE -> give us theability to use things like pattern matching a

ZLIB -> used fro data compression across the packet
```

Configuration

```
omar@ubuntu:/etc/snort$ cd /etc/snort/

omar@ubuntu:/etc/snort$ ls -l

total 360
-rw-r--r-- 1 root root    1281 Apr 20  2022 attribute_table.dtd
-rw-r--r-- 1 root root    3757 Apr 20  2022 classification.confi
-rw-r--r-- 1 root root   82469 Apr 19 14:32 community-sid-msg.ma
-rw-r--r-- 1 root root   23654 Apr 20  2022 file_magic.conf
-rw-r--r-- 1 root root   33339 Apr 20  2022 gen-msg.map
-rw-r--r-- 1 root root     687 Apr 20  2022 reference.config
drwxr-xr-x 2 root root    4096 Sep 22 20:08 rules
```

```
-rw-r----- 1 root snort  29773 Apr 19 14:32 snort.conf
-rw------- 1 root root     806 Sep 22 20:08 snort.debian.conf
-rw-r--r-- 1 root root    2335 Apr 20  2022 threshold.conf
-rw-r--r-- 1 root root  160606 Apr 20  2022 unicode.map
```

> 💡 rules directory contains all predefined rules that snort is going to use to detect specific patterns , `snort.conf` is the main configuration file

```
ls rules/


attack-responses.rules          community-mail-client.rules    co
backdoor.rules                  community-misc.rules           co
bad-traffic.rules               community-nntp.rules           co
chat.rules                      community-oracle.rules         do
community-bot.rules             community-policy.rules         de
community-deleted.rules         community-sip.rules            dn
community-dos.rules             community-smtp.rules           do
community-exploit.rules         community-sql-injection.rules  ex
community-ftp.rules             community-virus.rules          ex
community-game.rules            community-web-attacks.rules    fi
community-icmp.rules            community-web-cgi.rules        ft
community-imap.rules            community-web-client.rules     ic
community-inappropriate.rules   community-web-dos.rules        ic
```

> 💡 `/etc/snort/rules/local.rules` file is the file where we add our custom rules

before we do anything it’s good practice to take copy of the configuration file in case we make any mistakes

```
sudo cp snort.conf snort.conf.bak
```

1- setup the internal network that you want to protect and monitor

```
#from
ipvar HOME_NET any

#to
ipvar HOME_NET 192.168.1.0/24
```

2- here you can put the IP address of the critical component in the network such as DNS server , and web server etc.

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

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

# List of web servers on your network
ipvar HTTP_SERVERS $HOME_NET

# List of sql servers on your network
ipvar SQL_SERVERS $HOME_NET

# List of telnet servers on your network
ipvar TELNET_SERVERS $HOME_NET

# List of ssh servers on your network
ipvar SSH_SERVERS $HOME_NET
```

3- here you can find the path to the rules directory that snort take rules from as we mentioned

```
# Path to your rules files (this can be a relative path)
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\rules
var RULE_PATH /etc/snort/rules
var SO_RULE_PATH /etc/snort/so_rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules
```

rules that commented are deactivated , so that snort will not look at them

```
#include $RULE_PATH/app-detect.rules
include $RULE_PATH/attack-responses.rules
include $RULE_PATH/backdoor.rules
include $RULE_PATH/bad-traffic.rules
#include $RULE_PATH/blacklist.rules
#include $RULE_PATH/botnet-cnc.rules
#include $RULE_PATH/browser-chrome.rules
#include $RULE_PATH/browser-firefox.rules
#include $RULE_PATH/browser-ie.rules
#include $RULE_PATH/browser-other.rules
```

for practicing we will comment all rules so that we would create our custom rules

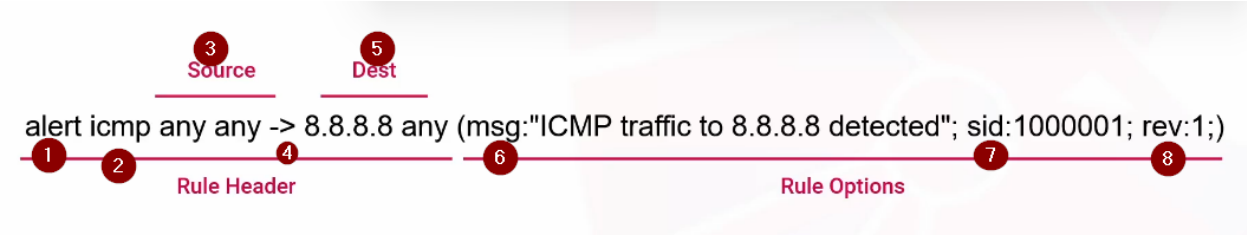
```
put the mouse in the first line you want to comment

shift + arrow down

till you reach the last rule before step 8

esc + 3
```

test snort after any configuration change to see if you broke anything

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

#navigate to the output , you will find that all rules are deact
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Initializing rule chains...
0 Snort rules read
```

```
    0 detection rules
    0 decoder rules
    0 preprocessor rules
0 Option Chains linked into 0 Chain Headers


#will find at the end of output

Total snort Fixed Memory Cost - MaxRss:50944
Snort successfully validated the configuration!
Snort exiting
```

Creating Rules



- The **rule header** contains the basic network-matching conditions: what action to take, which protocol to inspect, and what source and destination IPs/ports to match.
- The **rule options** provide additional information and actions, such as setting a message (`msg`), defining a rule identifier (`sid`), and keeping track of revisions (`rev`).

1. Action (Rule Header):

- This specifies what action the rule should take when it matches traffic. Common actions are:
  - `alert`: Generates an alert but allows the traffic.
  - `log`: Logs the traffic.

- `pass`: Ignores the matching traffic (essentially whitelisting).
- `drop` or `reject`: Blocks the traffic.

- **Summary of Rule Actions:**

Action	Blocks Traffic	Sends Alert	Sends Response
`alert`	No	Yes	No
`log`	No	No (logs only)	No
`pass`	No	No (whitelists)	No
`drop`	Yes	Yes	No
`reject`	Yes	Yes	Yes

In this case, the action is `alert`, which means it will generate an alert when the rule matches.

2. Protocol (Rule Header):

- This defines the network protocol to inspect, such as:
  - `tcp`: Transmission Control Protocol.
  - `udp`: User Datagram Protocol.
  - `icmp`: Internet Control Message Protocol (used for pings).
  - `ip`: Any IP protocol.

In this example, the protocol is `icmp`, which is used for ping traffic and error messages in networking.

3. Source IP (Rule Header):

- This defines the source of the network traffic. It can be:
  - A specific IP address (e.g., `192.168.1.1`).
  - A network range in CIDR notation (e.g., `192.168.1.0/24`).
  - `any`: This matches any IP address.

The rule in the image has `any` as the source IP, meaning it will match traffic from any source.

3.Source Port (Rule Header):

- This field specifies the source port for `tcp` or `udp` traffic.
- For `icmp` traffic, which doesn't have ports, this field is typically set to `any`.
- Ports are specified as a number, a range, or `any` for all ports.

In this case, the source port is set to `any`, since ICMP doesn't have ports.

4. Directions

- `>` **(unidirectional)**:
  - This arrow specifies that the rule applies to traffic going from the **source** to the **destination**.
  - Example:

    ```
    alert tcp 192.168.1.0/24 any -> 10.0.0.1 80 (msg:"HTTP traffic detected"; sid:1000002; rev:1;)
    ```

    - This rule will only match TCP traffic going from the `192.168.1.0/24` network to `10.0.0.1` on port 80 (unidirectional).
- `<>` **(bidirectional)**:
  - This symbol specifies that the rule applies to traffic in **both directions**, meaning the rule will trigger if the source and destination are swapped.
  - Example:

    ```
    alert tcp any any <> 192.168.1.0/24 80 (msg:"Bidirectional HTTP traffic"; sid:1000003; rev:1;)
    ```

    - This rule will match any TCP traffic involving the `192.168.1.0/24` network on port 80, regardless of whether the traffic is from the `192.168.1.0/24`

network to another host or from another host to the `192.168.1.0/24` network (bidirectional).

5. Destination IP (Rule Header):

- This defines the destination IP address or network range.
- Like the source IP, it can be a specific IP, a CIDR range, or `any`.

In this example, the destination IP is `8.8.8.8`, which is a public DNS server (used here as an example).

5. Destination Port (Rule Header):

- Specifies the destination port for `tcp` or `udp` traffic.
- For `icmp`, this field is typically set to `any` because ICMP doesn't use port numbers.

6-Rule Options (msg, SID, rev, etc.):

- **6- msg**: A human-readable message that will be included in the alert when the rule triggers. It is enclosed in double quotes and typically provides information about what the rule is detecting.
- **7- SID (Snort ID)**: A unique identifier assigned to the rule. It is used to identify the rule in logs and alerts. Numbers below 1,000,000 are reserved for Snort, while custom rules typically use numbers starting from 1,000,001.
- **8- rev (Revision)**: Specifies the revision number of the rule. This allows you to track changes to rules over time. When you update or modify a rule, you increment the revision number to indicate that it has been updated.

End of each Rule Option :

- Each rule option must end with a semicolon (`;`), indicating the end of the rule's logic and options.

> you can use this site to create rules for you : http://snorpy.cyb3rs3c.net/

Alert Mode

1- Alert with Full mode (default) : Display alert in a file with all details

```
snort -A full -c /etc/snort/snort.conf -i eth0 -l /var/log/snort
```

2- Alert with fast mode : display alerts with few details such as IP's , ports , protocols , msg , in the file specified

```
snort -A fast -c /etc/snort/snort.conf -i eth0 -l /var/log/snort
```

3- Alert with console : display alert directly in console in fast mode however we can also specify file to log

```
snort -A console -c /etc/snort/snort.conf -i eth0 -l /var/log/sn
```

example 1

```
omar@ubuntu:/logs$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=116 time=49.1 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=48.2 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=46.2 ms


omar@ubuntu:/etc/snort/rules$ sudo snort -i enp0s3 -A console -l
```

```
[sudo] password for omar:
09/23-12:17:10.544475  [**] [1:1000001:1] ICMP traffic to 8.8.8
09/23-12:17:11.545571  [**] [1:1000001:1] ICMP traffic to 8.8.8
09/23-12:17:12.551391  [**] [1:1000001:1] ICMP traffic to 8.8.8
09/23-12:17:13.559143  [**] [1:1000001:1] ICMP traffic to 8.8.8
```

example 2

another generic example but with fast mode , the rule is to block any remote connection on specific port such as 4444 which is most `msf` exploits and malware using it

rule

```
alert tcp any any -> any 4444 (msg:"remote connection on 4444  (
```

trigger the rule

```
mar@ubuntu:/var/log/snort$ sudo hping3 -c 1 -p 4444 -S example.c
HPING example.com (enp0s3 93.184.215.14): S set, 40 headers + 0
```

snort

```
omar@ubuntu:/etc/snort/rules$ sudo snort -i enp0s3 -A fast -l /v
```

detection

```
cd /var/log/snort

omar@ubuntu:/var/log/snort$ ls
```

```
alert  snort.log.1727084663


#read the alert file with cat , this alerts could be dinamicall
omar@ubuntu:/var/log/snort$ cat alert
09/23-12:44:30.642523  [**] [1:1000002:1] remote connection on


#logs file contains the raw data of the packet for further anal
omar@ubuntu:/var/log/snort$ sudo wireshark snort.log.1727084663
```

> 💡 here where network traffic analysis and network traffic monitoring tied together

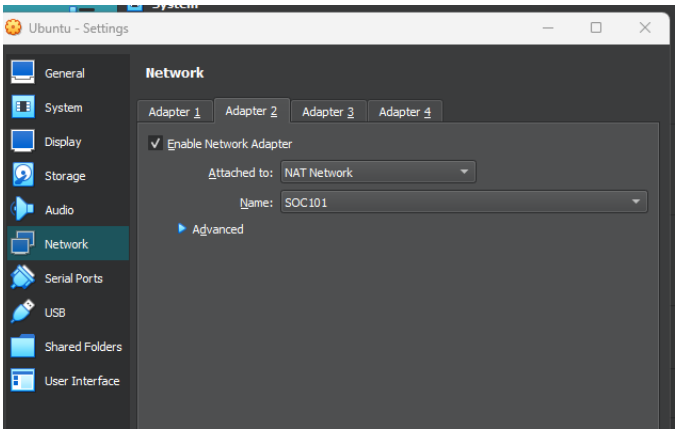
Inline Mode (IPS)

in this mode we can block , drop , reject connection with snort

first we need some configuration to operate in this mode

1- create another interface in the VM within the same subnet and make bridge between them

```
enp0s3:enp0s8
```



2- use another library instead of the default one

```
--daq afpacket
```

example 1: drop any ftp connection either sent or received

rule : added to `local.rules` file

```
drop tcp any any <> any 21 (msg:"ftp packet drop"; sid:1000003;
```

set snort

```
omar@ubuntu:~$ sudo snort -q -A console -i enp0s3:enp0s8 -c /etc
```

trigger alert by connecting to free ftp server that used to testing purpose

```
omar@ubuntu:~$ ftp test.rebex.net
Connected to test.rebex.net.

421 Service not available, remote server timed out. Connection c
```

snort outpet

```
omar@ubuntu:~$ sudo snort -q -A console -i enp0s3:enp0s8 -c /etc
09/23-13:24:07.701143  [Drop] [**] [1:1000003:1] ftp packet drop
09/23-13:25:07.702693  [Drop] [**] [1:1000003:1] ftp packet drop
```

Custom rules for real scenarios

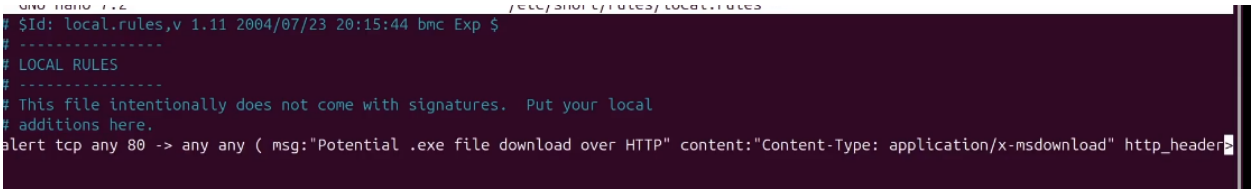
in this section we will create rules and run against PCAP files to extract IOC's and malicious activities based on our traffic analysis

1- detect exe files within URI


```
alert tcp any any -> any 80 (msg:"exe file detected in requested
```

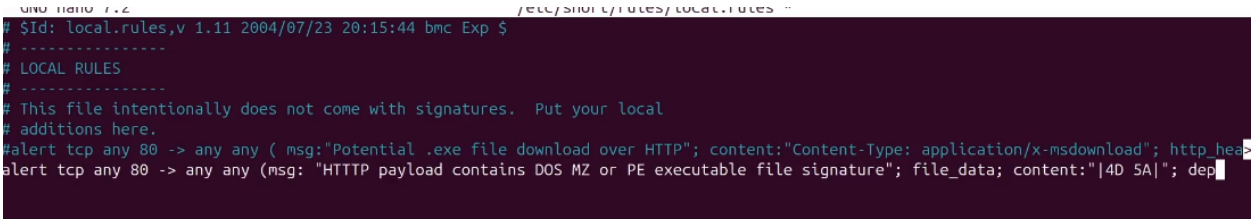
> 💡 `nocase` : means it’s not case sensitive

2-block response if content type “ `application/x-msdownload` ”




```
alert tcp any 80 -> any any (msg:"malicious file detected in hea
```

3- detecting magic type for MZ



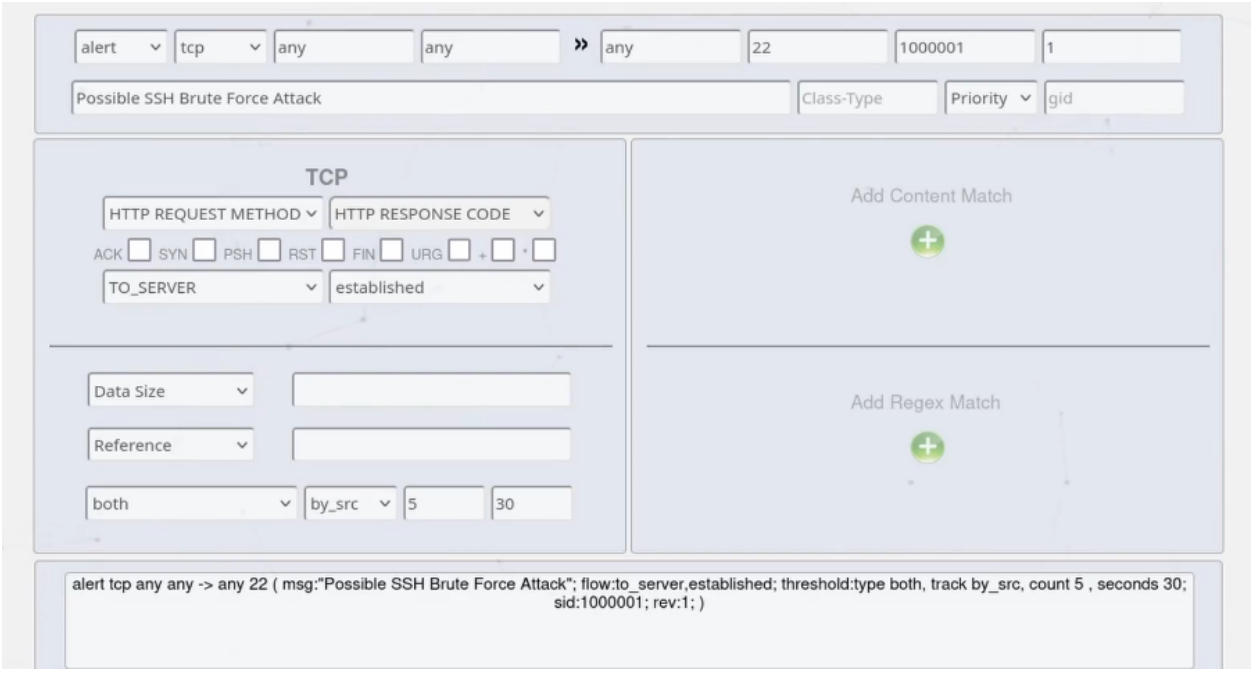

```
alert tcp any 80 -> any any(msg:"MZ magic byte detected in file'
```

> 💡 `depth : 2` means look for first 2 bytes in file data

4-detecting user agent for `SSLoad`

```
alert tcp any any -> any any ( msg:"user agent for ssload "; co
```

5-detect SSH brute force :




```
alert tcp any any -> any 22 (msg:"possible ssh brute force "; fl
```

difference between threshold and limit and both

1-threshold tracking type:

- `threshold` will generate an alert when exceeding the count even if the limit time is not exceeding
- `both` ensures that the alerts will be triggered if count of tries is exceeds within the time limit

2-

`by_src`: This means that Snort will track the connections **by the source IP address**. In this case, it tracks how many connection attempts are made from a single source IP to the SSH server.

3-
`Count 5` This indicates the number of connection attempts required to trigger the rule (key and value separated by space not colon)

4- `seconds 30;` **Time Window:** This specifies the time window in which the connection attempts must occur for the rule to be triggered(key and value separated by space not colon)

5- `flow:to_server , established;` :

- **Only traffic going to the server** is inspected (client requests, not server responses).
- **Only packets from an established TCP connection** are checked, meaning the client and server have already completed the TCP handshake

how to test snort against PCAP file

```
sudo snort -c /etc/snort/snort.conf -q -r <pcap file> -A consol
```

Challenge

1- file captured : 2.49 minutes - 30k packets

2- conversations:

```
internal ip : 192.168.1.6 - 192.168.1.7 (top talkers)
```

3- protocols

```
http : 3600
ssh : 1099
ftp : 18
```

4- http analysis show that attacker was trying to brute force on `/login.php` endpoint

after redirection to `/admin.php` he tries `lfi` and reached to `hosts` and `passwd` and `ssh` private keys

after getting the SSH private keys , attacker connected to SSH server and made what he wants on the server with encrypted session

1-Create a Snort rule to detect if 10 failed login attempts (HTTP 401 response codes) occur within a 30-second period from the same IP address. ?

```
alert tcp any any -> any 80 ( msg:"possible brute force detecte
```

2-create a Snort rule to detect any successful logins

```
alert tcp any 80 -> any any ( msg:"successful login detected ";
```

3-Create a Snort rule to detect any packets with this string in the Request URI

```
alert tcp any any -> any 80 (msg:"LFI attack detected"; content
```

4- Create a Snort rule to detect any outgoing connections to an external FTP server

```
alert tcp any any -> any 21 (msg:"outgoing ftp connection";  si
```