SNORT v3

Intrusion Detection & Prevention System

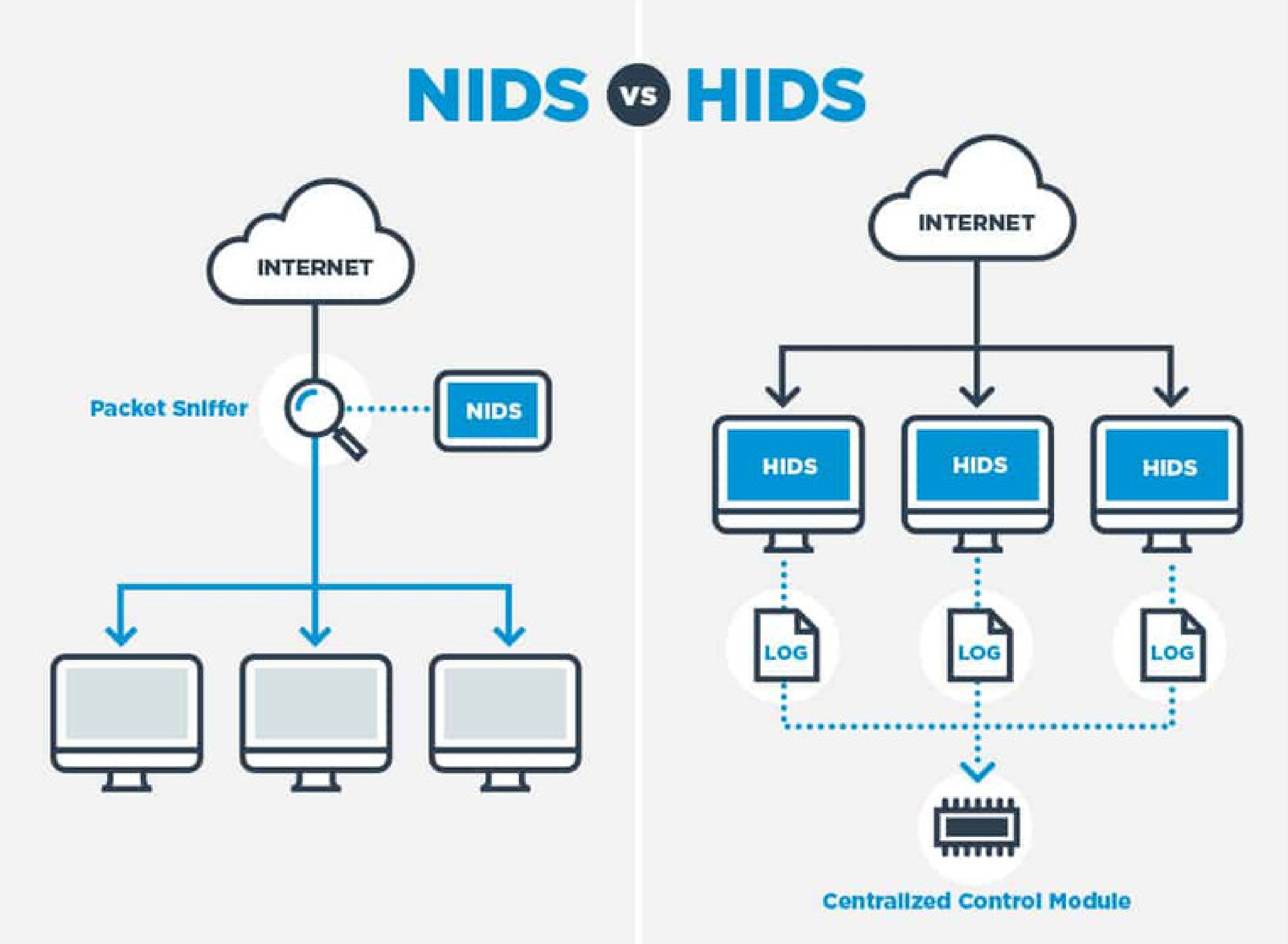
Edoardo Mich
Davide Moletta
Tefera Addis Sisay

Network Security Lab 22/23

CISCO

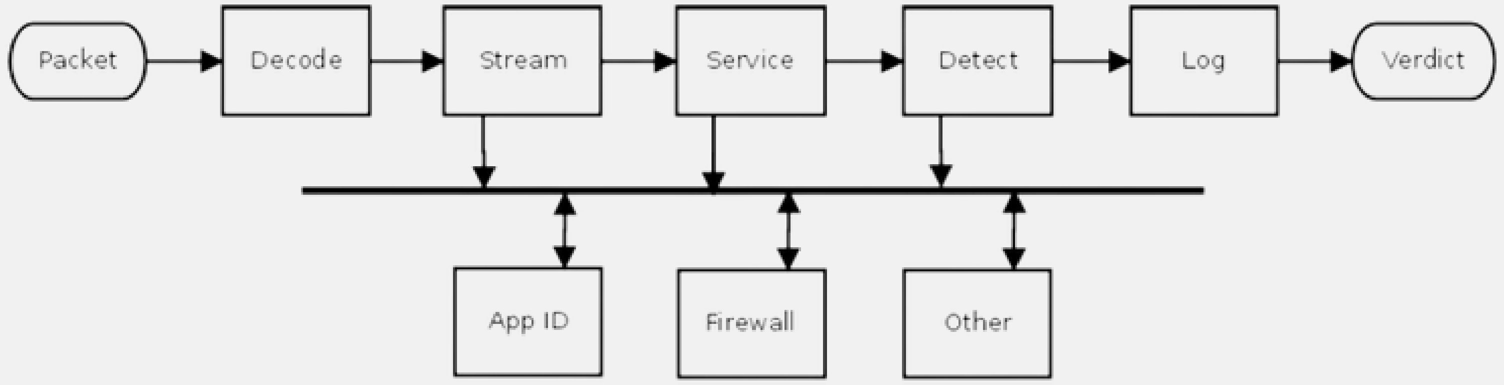
Snort as NIDS

SNORT3



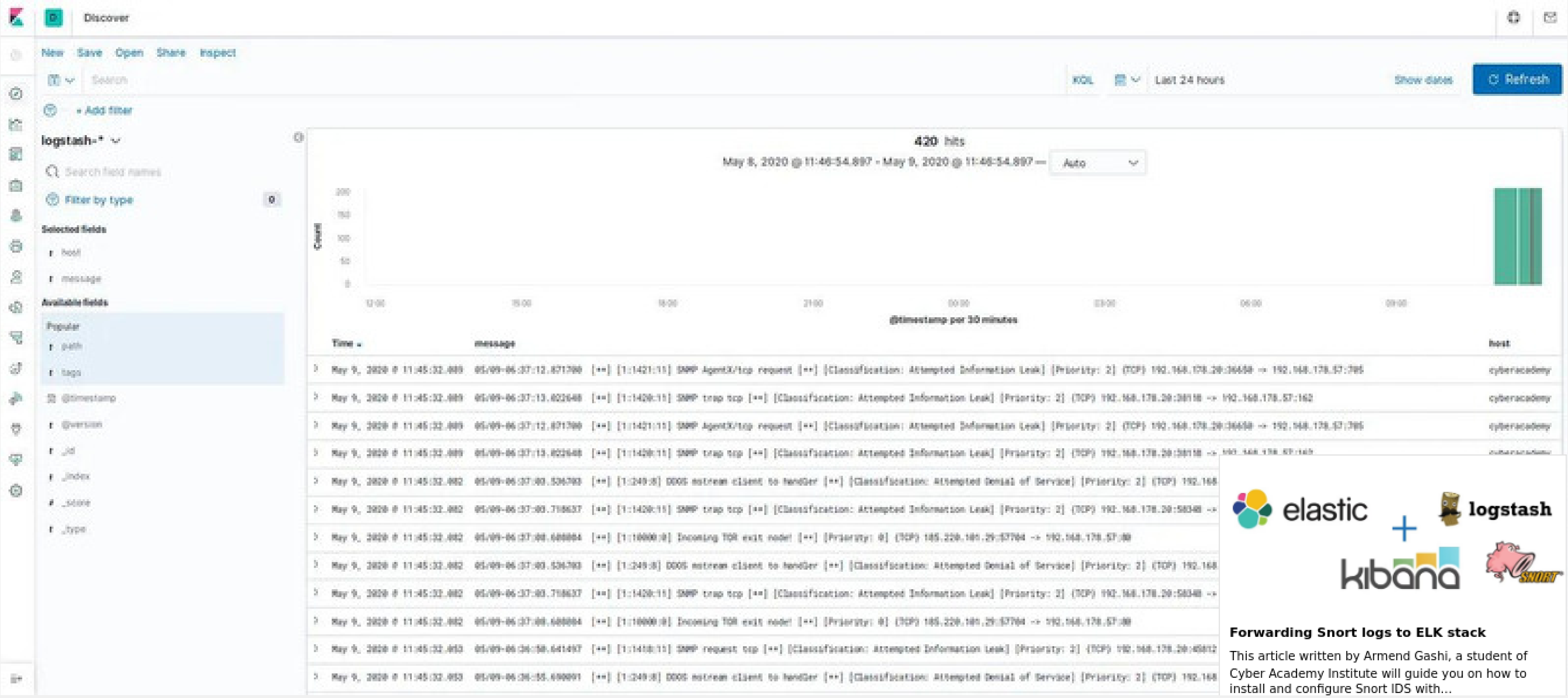
SURICATA

zeek



Snort as a service

Use parameter
`-A alert_json`
for easy transfer to SIEM software like Splunk, ELK stack...

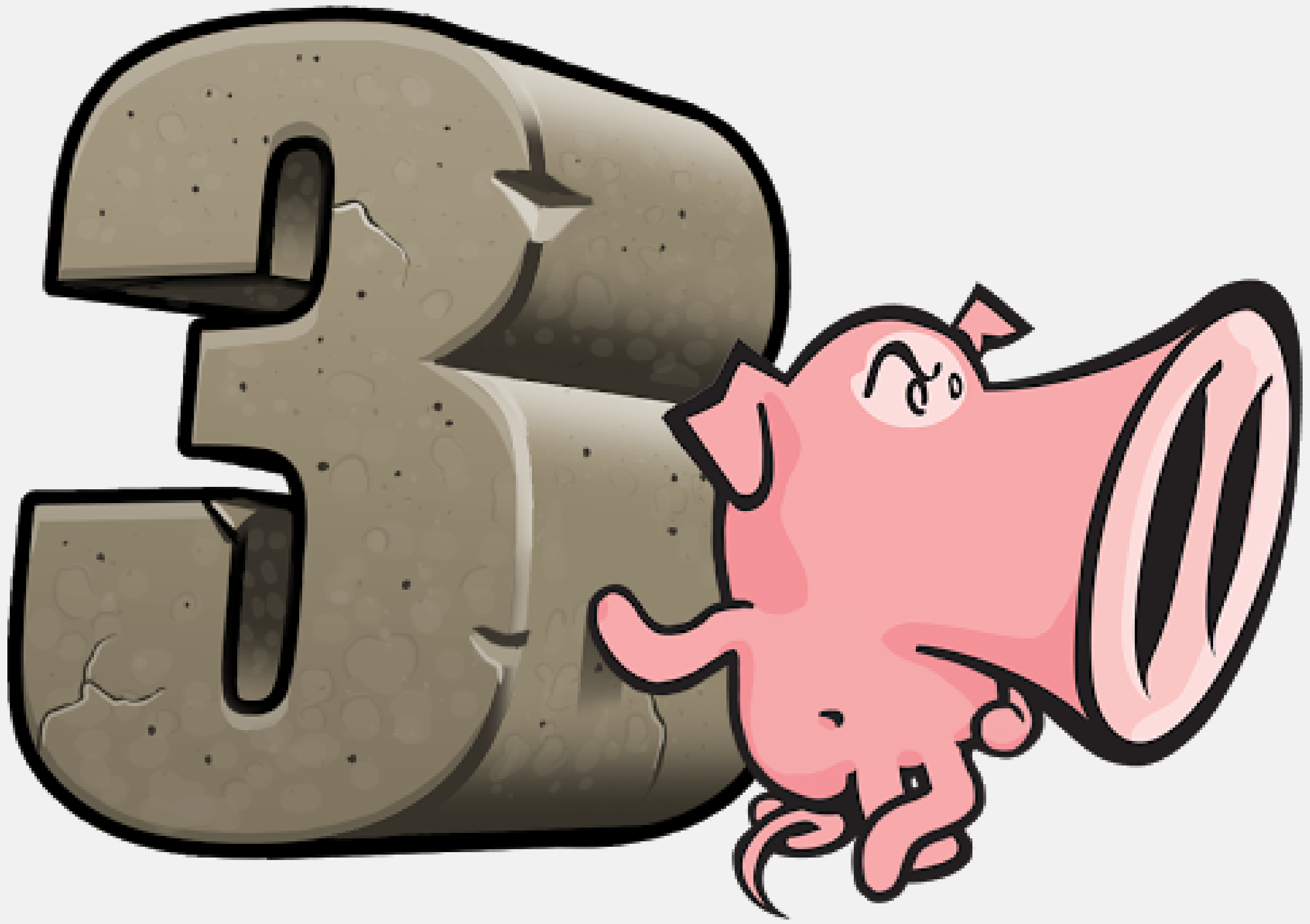


```
Time ▾                        message                                                                                                                                                                          host
> May 9, 2020 @ 11:45:32.089  05/09-06:37:12.871700 [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.178.20:36650 -> 192.168.178.57:705  cyberacademy
> May 9, 2020 @ 11:45:32.089  05/09-06:37:13.022648 [**] [1:1420:11] SNMP trap tcp [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.178.20:38118 -> 192.168.178.57:162           cyberacademy
> May 9, 2020 @ 11:45:32.089  05/09-06:37:12.871700 [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.178.20:36650 -> 192.168.178.57:705  cyberacademy
> May 9, 2020 @ 11:45:32.089  05/09-06:37:13.022648 [**] [1:1420:11] SNMP trap tcp [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.178.20:38118 -> 192.168.178.57:162           cyberacademy
> May 9, 2020 @ 11:45:32.082  05/09-06:37:03.536703 [**] [1:249:8] DDOS mstream client to handler [**] [Classification: Attempted Denial of Service] [Priority: 2] {TCP} 192.168
> May 9, 2020 @ 11:45:32.082  05/09-06:37:03.718637 [**] [1:1420:11] SNMP trap tcp [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.178.20:58348 ->
> May 9, 2020 @ 11:45:32.082  05/09-06:37:08.688004 [**] [1:10000:0] Incoming TOR exit node! [**] [Priority: 0] {TCP} 185.220.101.29:57704 -> 192.168.178.57:80
> May 9, 2020 @ 11:45:32.082  05/09-06:37:03.536703 [**] [1:249:8] DDOS mstream client to handler [**] [Classification: Attempted Denial of Service] [Priority: 2] {TCP} 192.168
> May 9, 2020 @ 11:45:32.082  05/09-06:37:03.718637 [**] [1:1420:11] SNMP trap tcp [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.178.20:58348 ->
> May 9, 2020 @ 11:45:32.082  05/09-06:37:08.688004 [**] [1:10000:0] Incoming TOR exit node! [**] [Priority: 0] {TCP} 185.220.101.29:57704 -> 192.168.178.57:80
> May 9, 2020 @ 11:45:32.053  05/09-06:36:58.641497 [**] [1:1418:11] SNMP request tcp [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.178.20:45812
> May 9, 2020 @ 11:45:32.053  05/09-06:36:55.690091 [**] [1:249:8] DDOS mstream client to handler [**] [Classification: Attempted Denial of Service] [Priority: 2] {TCP} 192.168
```

Forwarding Snort logs to ELK stack

This article written by Armend Gashi, a student of Cyber Academy Institute will guide you on how to install and configure Snort IDS with...

Snort2 vs Snort3



- New rule parser and rule syntax.
- Support for multiple packet-processing threads, which frees up more memory for packet processing.
- Use of a shared configuration and attribute table.
- Access to more than 200 plugins.
- Rewritten TCP handling.
- Improved shared object rules, including the ability to add rules for zero-day vulnerabilities.
- New performance monitor.
- New rule remarks and comments that are inside of the rule itself.

https://s3.amazonaws.com/snort-org-site/production/document_files/files/000/004/341/original/snort3_information.pdf?1612471639

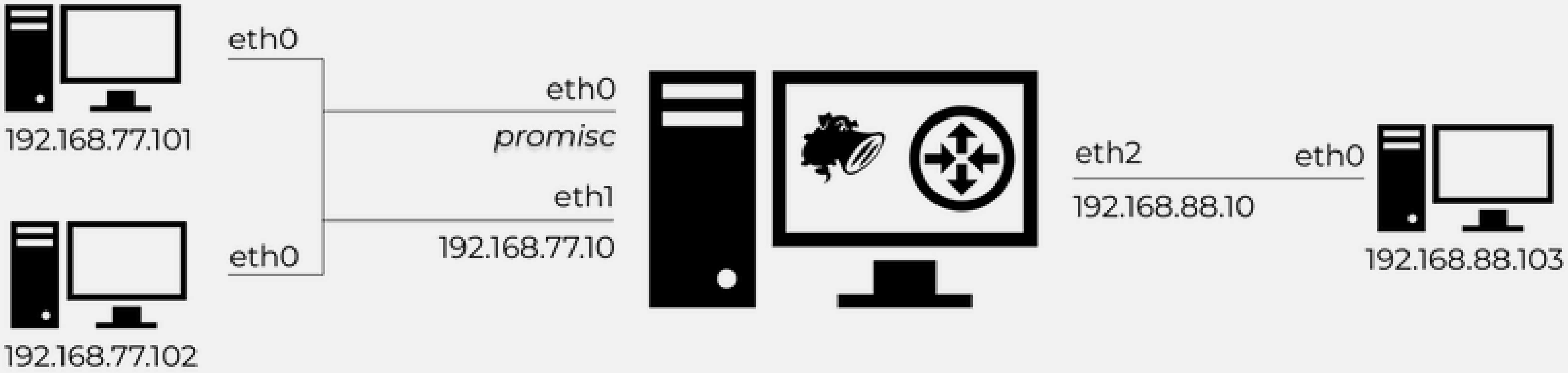
Snort3 installation

snort3/libdaq

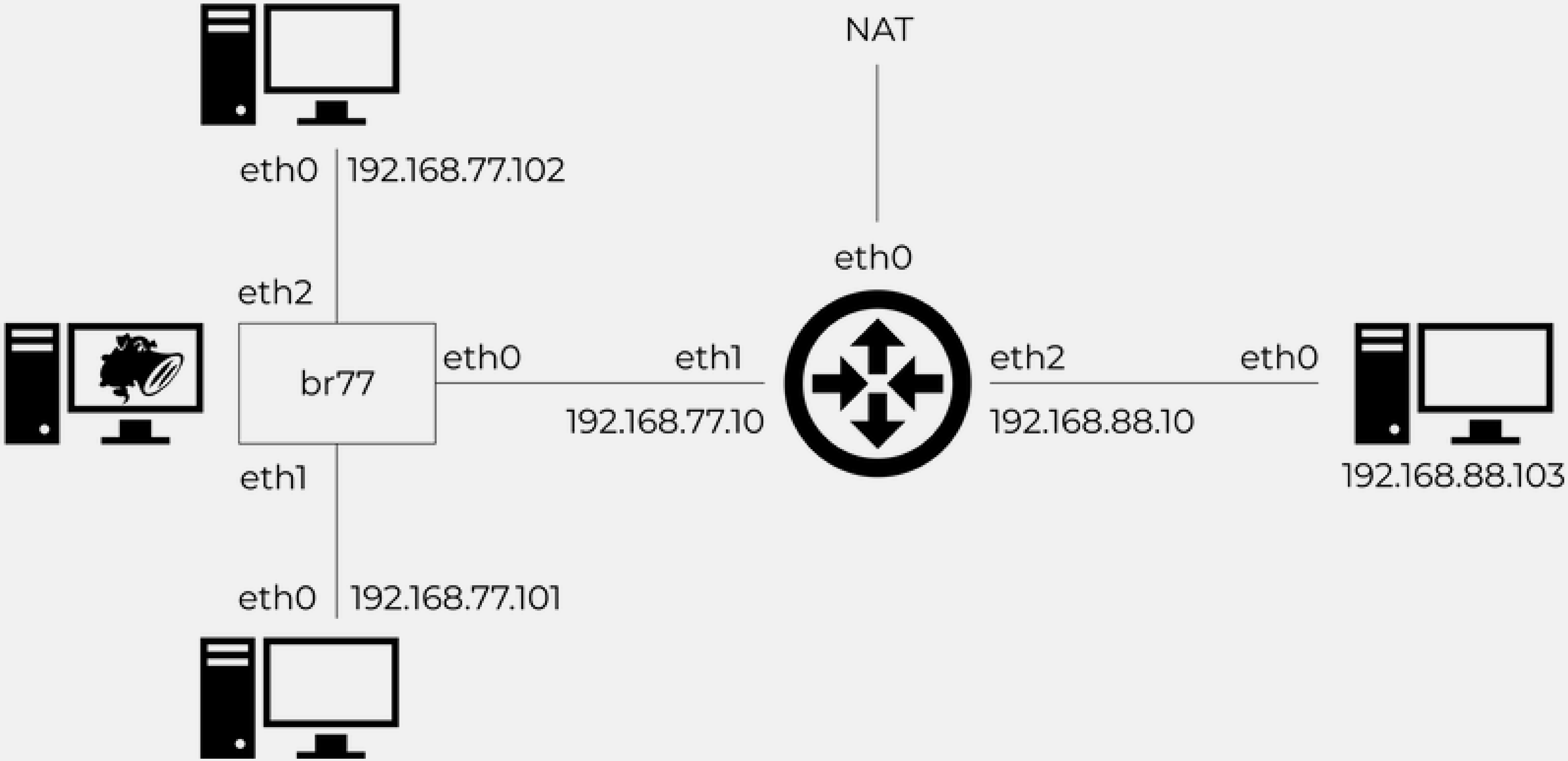
snort3/snort3

```
pkgname=snort3
pkgver=3.1.61.0
pkgrel=0
pkgdesc="Snort 3 is the next generation Snort IPS (Intrusion Prevention System)"
url="https://github.com/snort3/snort3"
arch="all"
license="GPL V2"
depends="daq"
makedepends="cmake build-base autoconf automake cpputest flex-dev libuuid
libtool hwloc-dev libnet-dev libdnet-dev libpcap-dev libpcre32 libtirpc-dev luajit-dev
openssl-dev libssl3 libnetfilter_queue-dev libmnl-dev libunwind-dev zlib-dev pcre-dev
xz-dev gperftools-dev"
builddir="$srcdir/$pkgname-$pkgver"

build() {
   ./configure_cmake.sh --prefix=/usr/local --enable-tcmalloc
}

package() {
   cd ./build
   make
   make DESTDIR="$pkgdir" install
}
```

Old Network setup



Network setup



Rules syntax

```
action protocol IPaddr port# -> IPaddr port#
(options)
```

Illustration by Jessie Hall

From the 2023 Snort parody calendar. Snort, and the Snort pig are registered trademarks of Cisco. All rights reserved.

Rules syntax

SNORT actions

```
action protocol IPaddr port# -> IPaddr port#
(options)
```

Basic actions:

- *Alert*: generate an alert on the single packet
- *Block*: block the current packet and all the packets in this flow
- *Drop*: drop the current packet
- *Log*: log the current packet
- *Pass*: mark the current packet as passed

Active responses:

- *React*: send response to client and terminate session.
- *Reject*: terminate session with TCP reset or ICMP unreachable
- *Rewrite*: enables overwrite packet contents based on a "replace" option in the rules

Rules syntax

SNORT Protocols and Services

```
action protocol IPaddr port# -> IPaddr port#
(options)
```

Protocols:

- IP
- ICMP
- TCP
- UDP

Services:

- SSL
- HTTP
- SMTP
- POP3

Rules syntax

IP addresses

`action protocol` **`IPaddr`** `port# ->` **`IPaddr`** `port#`
`(options)`

IP addresses on which the rule should be applied, can be defined in four ways:

- Numeric IP address: 192.168.77.101, 192.168.77.0/24
- Variable (using $) defined in the Snort config: $HOME_NET
- The keyword *any*, meaning any IP address
- List of IP addresses or subnets: [192.168.77.0/24,10.1.1.0/24]

Rules syntax

Ports

```
action protocol IPaddr port# -> IPaddr port#
(options)
```

Ports on which the traffic will be filtered, can be defined in five different ways:

- The keyword any, meaning any port
- Static port: 8080
- Variable (using $) defined in the Snort config: $HTTP_PORTS
- Port ranges indicated with the range operator: 1:1024
- List of static ports: [1:1024,4444,5555]

Rules syntax

Direction operators

```
action protocol IPaddr port# -> IPaddr port#
(options)
```

There are two valid direction operators that indicates the direction of the traffic that the rule should apply to:

- ->: The first IP is the source and the second is the destination
- <>: Both addresses are source and destination

Rules syntax

Snort options

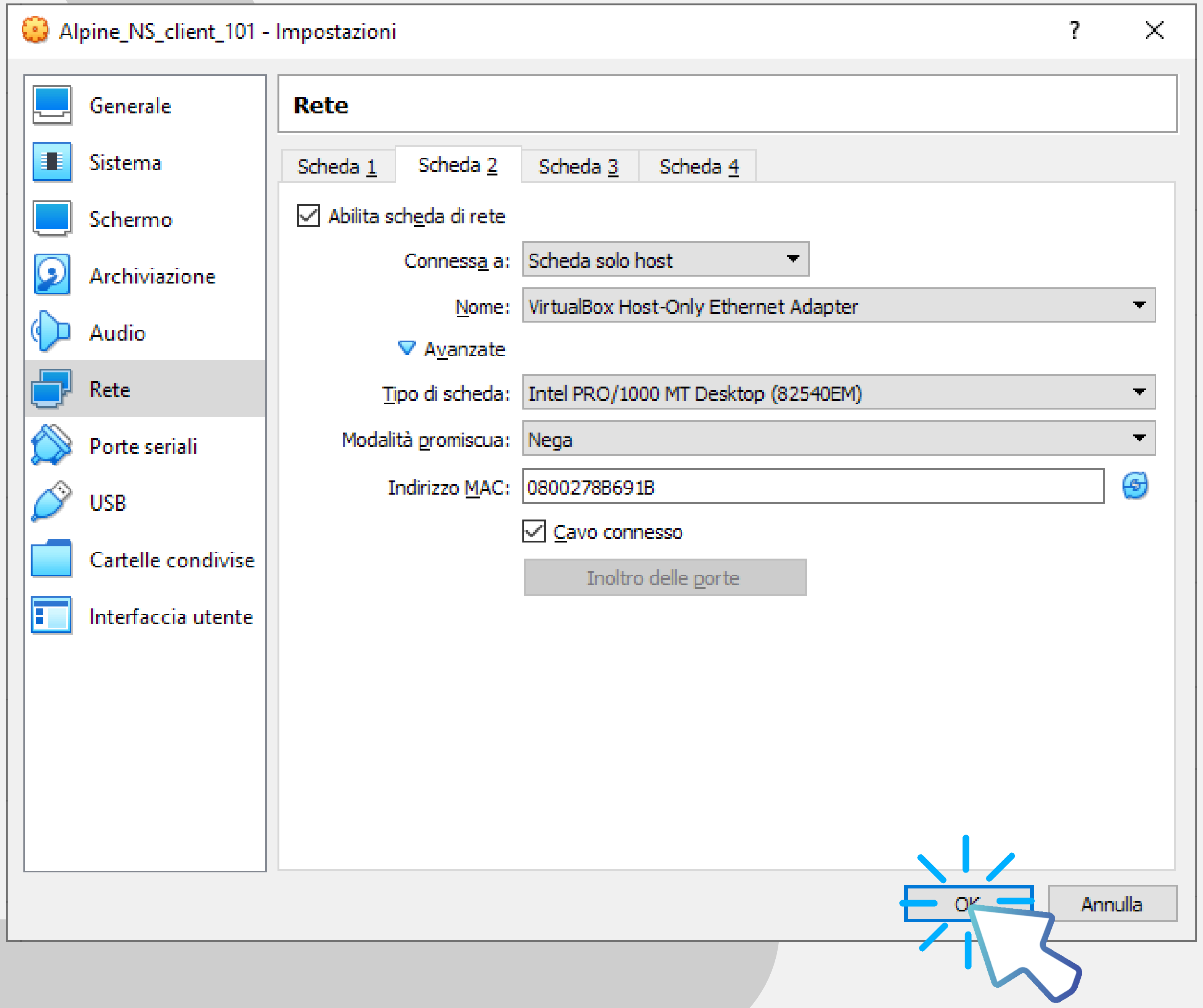
```
action protocol IPaddr port# -> IPaddr port#
(options)
```

All rule options are enclosed in parentheses after the end of the rule.
Each rule option is declared as key-value: <name>: <specific criteria>;
Some examples of rule options are:

- *msg*: set a message to be displayed when the rule fires
- *sid*: set a numeric identifier to the rule
- *priority*: set a priority to the rule
- *content*: used to perform basic pattern matching against packet data

DISCALMER

```
ssh root@<ip>
```



Alpine_NS_client_101 - Impostazioni

Rete

☑ Abilita scheda di rete

Connessa a: Scheda solo host

Nome: VirtualBox Host-Only Ethernet Adapter

Tipo di scheda: Intel PRO/1000 MT Desktop (82540EM)

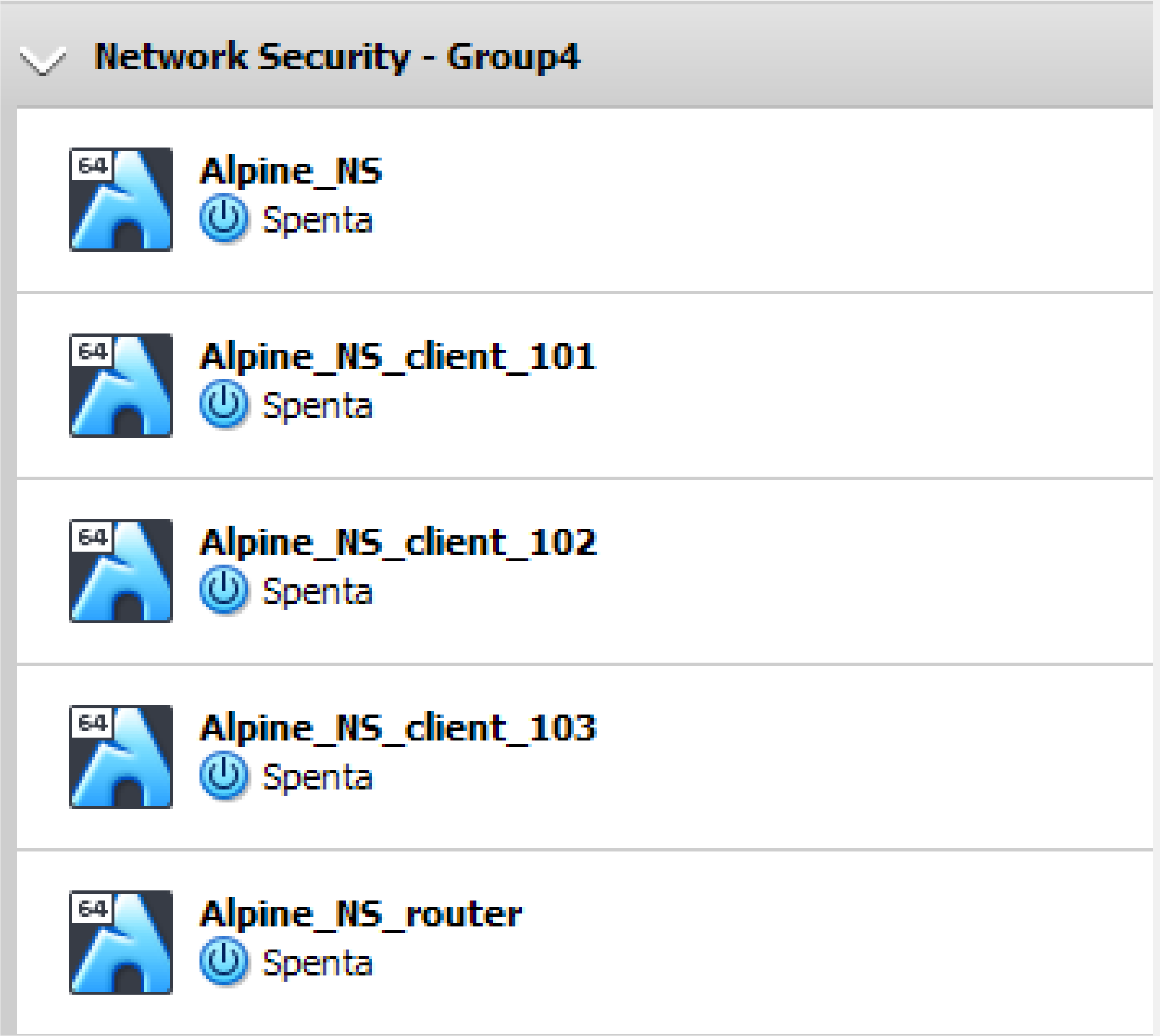
Modalità promiscua: Nega

Indirizzo MAC: 0800278B691B

☑ Cavo connesso

Virtual Machines

Credentials



Network Security - Group4

- Alpine_NS
  Spenta
- Alpine_NS_client_101
  Spenta
- Alpine_NS_client_102
  Spenta
- Alpine_NS_client_103
  Spenta
- Alpine_NS_router
  Spenta

👤 root

🔒 admin

Directories

Where to find things

```
snort
├── binaries
├── confs
│   ├── snort_ids.lua
│   └── snort_ips_nfq.lua
├── dumps
│   ├── IDS_exercise
│   ├── sqlinj.pcap
│   └── Qakbot.zip
├── enableNFQ.sh
└── rules
    └── snort3-community-rules
```

Aliases

```
snort3="snort --daq afpacket -A alert_full"

snort3_ips="snort --daq-dir /usr/local/lib/daq
            -c /root/snort/confs/snort_ips_nfq.lua -Q -A alert_full"

snort3_pcap="snort --daq pcap -A alert_fast -k none"

snort3_qakbot="snort -c /usr/local/etc/snort/snort.lua -A alert_full
               --lua \"alert_full = { file=true }\""

snort3_portscan="snort --daq afpacket -A alert_fast -i br77
                -c /usr/local/etc/snort/snort.lua
                --lua "port_scan = default_low_port_scan
                ips = {enable_builtin_rules = true, variables = default_variables}""
```

Exercise 1

Write a rule to detect ping from any IP to any IP

Run snort with the following command

```
snort3 -i br77 -R snort/rules/exercise01.rules
```

Hint: use the Snort option *sid:1*
to give an identifier to your rule

Exercise 1 solution

Write a rule to detect ping from any IP to any IP

```
alert icmp any any -> any any
(
    sid:1
)
```

Exercise 2

Write a rule to detect ping from .77.101 to .77.102 and write a message when the rule fires

Run snort with the following command

```
snort3 -i br77 -R snort/rules/exercise02.rules
```

Hint: use the Snort option *msg:""*
to set a message for your rule

Exercise 2 test

Execute those commands:

```
On client 101 (alerted)
$ ping 192.168.77.102

On client 102 (only replies from 101 alerted)
$ ping 192.168.77.101
```

Exercise 2 solution

Write a rule to detect ping from .77.101 to .77.102 and write a message when the rule fires

```
alert icmp 192.168.77.101 any -> 192.168.77.102 any
(
   sid:2;
   msg:"ping from client 101 to client 102"
)
```

Exercise 3

Write a rule to detect ping replies from .77.101 to .77.102

Run snort with the following command

```
snort3 -i br77 -R snort/rules/exercise03.rules
```

Hint: use Wiresharsk to inpect the packets and use the Snort option *itype:* to write the rule

Exercise 3 extra

Write a rule to detect ping replies from .77.101 to .77.102

The **itype** rule option is used to compare a packet's ICMP type to a specified integer value. It is a non-payload detection option and it is used as follows

```
action icmp IPaddr port# -> IPaddr port#
(
   itype:X
)
```

Exercise 3 test

Execute those commands:

```
On client 101
$ ping 192.168.77.102

On client 102 (replies from client 101 alerted)
$ ping 192.168.77.101
```

Exercise 3 solution

Write a rule to detect ping replies
from .77.101 to .77.102

```
alert icmp 192.168.77.101 any -> 192.168.77.102 any
(
    itype:0;
    sid:3;
    msg:"ping reply from client 101 to client 102"
)
```

Exercise 4

Write a rule to detect TCP packets from .77.101 to .88.103

Run snort with the following command

```
snort3 -i br77 -R snort/rules/exercise04.rules
```

Hint: every client machine has a python script that sends TCP packets to the specified IP address.
Use *cat packet.py* to inspect the packet

Exercise 4 test

Execute those commands:

```
On client 101 (alerted)
$ python3 packet.py (input: 192.168.88.103)

On client 102
$ python3 packet.py (input: 192.168.88.103/.77.101)

On client 103 (reply from client 101 alerted)
$ python3 packet.py (input: 192.168.77.101)
```

Exercise 4 solution

Write a rule to detect TCP packets
from .77.101 to .88.103

```
alert tcp 192.168.77.101 any -> 192.168.88.103 any
(
    sid:4;
    msg:"A TCP packet from client 101 to client 103"
)
```

Exercise 5

Write a rule to detect TCP packets with ACK flag from .77.101 to .88.103

Run snort with the following command

```
snort3 -i br77 -R snort/rules/exercise05.rules
```

Hint: use Wiresharsk to inpect the packets and
use the Snort option *flags:* to write the rule

Exercise 5 extra

Write a rule to detect TCP packets with ACK flag from .77.101 to .88.103

The **flags** rule option is used to check if flag bits are set in the TCP header.
Additionally, one of the following modifiers can be added:

+ match the specified flags plus any others,

* match if any of the specified flags are set,

! match if the specified flags are not set.

```
action tcp IPaddr port# -> IPaddr port#
(
   flags:[+*!] F || S || R || P || A || U || C || E || 0
)
```

Exercise 5 test

Execute those commands:

```
On client 103
$ nc -lp <port#>

On client 101 (alerted)
$ nc 192.168.88.103 <port#>

On client 102
$ nc 192.168.88.103 <port#>
```

Exercise 5 solution

Write a rule to detect TCP packets with
ACK flag from .77.101 to .88.103

```
alert tcp 192.168.77.101 any -> 192.168.88.103 any
(
    flags:*A;
    sid:5;
    msg:"ACK tcp packet from client 101 to client 103"
)
```

Exercise 6

Write a rule to detect DNS requests from home network to anything but the company DNS server

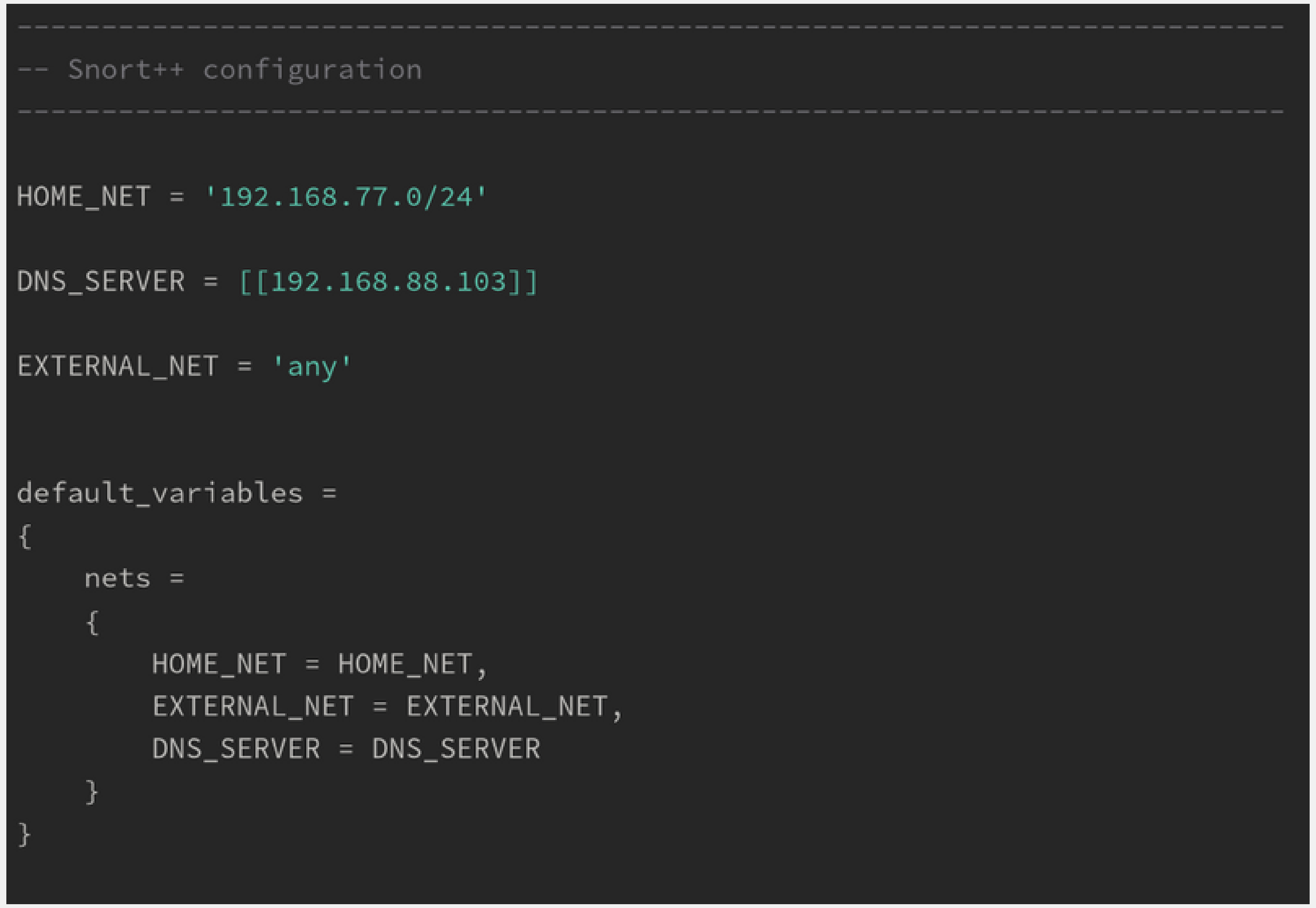
Run snort with the following command

```
snort3 -i br77 -R snort/rules/exercise06.rules
-c snort/confs/snort_ids.lua
```

Hint: rememeber that you can use variables ($VARNAME) instead of static IPs, check the configuration file snort_ids.lua to get further hints

Exercise 6 extra

Write a rule to detect DNS requests from home network to anything but the company DNS server



```
---------------------------------------------------------------------------
-- Snort++ configuration
---------------------------------------------------------------------------

HOME_NET = '192.168.77.0/24'

DNS_SERVER = [[192.168.88.103]]

EXTERNAL_NET = 'any'


default_variables =
{
    nets =
    {
        HOME_NET = HOME_NET,
        EXTERNAL_NET = EXTERNAL_NET,
        DNS_SERVER = DNS_SERVER
    }
}
```

Exercise 6 test

Execute those commands:

```
On client 101/102
$ nslookup google.com 192.168.88.103

On client 101/102 (alerted)
$ nslookup google.com 1.1.1.1
```

Exercise 6 solution

Write a rule to detect DNS requests from home network to anything but the company DNS server

```
alert udp $HOME_NET any -> !$DNS_SERVER 53
(
    sid:6;
    msg:"DNS request to a not standard DNS server"
)
```

Exercise 7

Write a rule to detect HTTP requests with unallowed user-agents from HOME_NET to .88.103

Run snort with the following command

```
snort3 -i br77 -R snort/rules/exercise07.rules
-c snort/confs/snort_ids.lua
```

Hint: the only user agent we want to accept is *wget* family
Use a combination of Snort options *http_header* and *content* to write the rule

Exercise 7 extra

Write a rule to detect HTTP requests with unallowed user-agents from HOME_NET to .88.103

The **http_header** rule option allows to look for content matches in HTTP header (we can specify in which field using **http_header: flied fieldName**).
The **content** rule option is used to perform pattern matching against packet data.
They are payload detection option and are used as follows

```
action http IPaddr port# -> IPaddr port#
(
   http_header:field fieldName;
   content:[!]"content_string", nocase #used to be case insensitive
)
```

Exercise 7 test

Execute those commands:

```
On client 103
$ python3 -m http.server

On client 101/102
$ wget 192.168.88.103:<port#>

On client 101/102 (alerted)
$ curl 192.168.88.103:<port#>
```

Exercise 7 solution

Write a rule to detect HTTP requests with unallowed user-agents from HOME_NET to .88.103

```
alert http $HOME_NET any -> any any
(
   http_header:field user-agent;
   content:!"wget", nocase;
   sid:7;
   msg:"Unallowed user-agent in HTTP request from HOME_NET"
)
```

Exercise 8

Write a rule to detect SQLi on the pcap file

Run snort with the following command

```
snort3_pcap -c snort/confs/snort_ids.lua
-r snort/dumps/sqlinj.pcap -R snort/rules/exercise08.rules
```

Hint: Use a combination of Snort options
http_uri, *content* and *pcre* to write a regex that match the traffic

Exercise 8 extra

Write a rule to detect SQLi on the pcap file

The **http_uri** rule option allows to look for content matches in HTTP uri.
The **pcre** rule option is used to match regular expression strings against packet data.
They are payload detection option and are used as follows:

```
action http IPaddr port# -> IPaddr port#
(
   http_uri:path || query || fragment || host || port || scheme;
   pcre:[!]"/pcre_string/[flag]"; #use pcrepattern sintax for regex
)
```

Exercise 8 solution

Write a rule to detect SQLi on the pcap file

```
alert http any any -> any 8080
(
   http_uri:query;
   content:"search=",nocase;
   pcre:"/(.*[\"\']\;)+.*/";
   sid:8;
   msg:"SQLi command in search parameter"
)
```

Exercise 9

An example of Snort3 portscan module

Run snort with the following command

```
snort --daq afpacket -A alert_fast -i br77
-c /usr/local/etc/snort/snort.lua
--lua "port_scan = default_low_port_scan
ips = {enable_builtin_rules = true, variables = default_variables}"
```

Hint: use the alias *snort3_portscan*

Exercise 9 test

Execute those commands:

```
On client 101
# TCP SYN port scan
$ nmap -nsS 192.168.77.102
# TCP connect port scan
$ nmap -nsT 192.168.77.102
# UDP port scan
$ nmap -nsU 192.168.77.102
```

Exercise 10

Write a rule to detect QAKbot on the pcap file

Unzip the Quakbot artifact ZIP:

```
unzip snort/dumps/Qakbot.zip -d snort/dumps/Qakbot
```

Password is 'infected'

Exercise 10

Write a rule to detect QAKbot on the pcap file

Run snort with the following command

```
snort3_qakbot -r snort/dumps/Qakbot/2023-05-24-obama264-
Qakbot-infection.pcap -R snort/rules/exercise10.rules
```

You can find the logs of snort in the file "alert_full" on the current directory

Exercise 10 solution

Write a rule to detect QAKbot on the pcap file

You can find the rules in the file /opt/snort_rules/exercise10.rules

IPS

Intrusion Prevention System

Run the following command to enable the IPS capabilities for snort

```
snort/enableNFQ.sh
```

Exercise 11

Write a rule to block ping from any IP to any IP

Run snort with the following command

```
snort3_ips -R snort/rules/exercise11.rules
```

Exercise 11 test

Execute those commands:

```
On client 103
$ tcpdump -ni eth0

On client 101/102
$ ping 192.168.88.103
```

Before running snort pings reach the destination and no alerts are displayed

After running snort pings don't reach the destination and an alert is displayed

Exercise 11 solution

Write a rule to block ping from any IP to any IP

```
block icmp any any -> any any
(
    sid:11
)
```

Exercise 12

Write a rule to block TCP packets with ACK flags from .77.101 to .88.103

Run snort with the following command

```
snort3_ips -R snort/rules/exercise12.rules
```

Exercise 12 test

Execute those commands:

```
On client 103
$ nc -lp <port#>

On client 101
$ nc 192.168.88.103 <port#>
```

Before running snort connection is accepted and messages reach the destination

After running snort connection is refused and messages don't reach the destination

Exercise 12 solution

Write a rule to block TCP packets with ACK flags from .77.101 to .88.103

```
block tcp 192.168.77.101 any -> 192.168.88.103 any
(
   flags:*A;
   sid:12;
   msg:"Blocked an ACK packet from client 101 to client 103"
)
```

Exercise 13

Write a rule to drop DNS requests for unitn.it domains from HOME_NET to not company DNS server

Run snort with the following command

```
snort3_ips -R snort/rules/exercise13.rules
```

Hint: use Wireshark to inspect payload packets and use the Snort option *content* to wirte the rule

Exercise 13 extra

Write a rule to drop DNS requests for unitn.it domains from HOME_NET to not company DNS server

As we have seen the **content** rule option is used to perform pattern matching against packet data.
It allows to match hex bytes when enclosed in **|** characters.

```
action protocol IPaddr port# -> IPaddr port#
(
   content:"|hexvalues|"
)
```

Exercise 13 test

Execute those commands:

```
on client 103
tcpdump -ni eht0
```

```
On client 101/102
$ nslookup google.com 192.168.88.103
$ nslookup google.com 1.1.1.1
$ nslookup unitn.it 192.168.88.103
$ nslookup unitn.it 1.1.1.1
```

Before running snort all commands runs without alerts

After running snort the last command is blocked and snort generates an alert

Exercise 13 solution

Write a rule to drop DNS requests for unitn.it domains to not company DNS server

```
drop udp $HOME_NET any -> !$DNS_SERVER 53
(
    content:"|05 75 6e 69 74 6e 02 69 74 00|";
    sid:13;
    msg:"Blocked DNS request to a not standard DNS server"
)
```

Exercise 14

Write a rule to block sensible SQLi on the pcap file

Run snort with the following command

```
snort3_pcap -c snort/confs/snort_ids.lua
-r snort/dumps/sqlinj.pcap -R snort/rules/exercise14.rules
```

Exercise 14 solution

Write a rule to block sensible SQLi on the pcap file

```
block http any any -> any 8080
(
    http_uri:query;
    content:"search=",nocase;
    pcre:"/(.*[\"\']\;)+.*(DROP|INSERT)+/";
    sid:14;
    msg:"Blocked SQL command in search parameter"
)
```

Conclusions

To learn more about Snort rules

Official Snort 3 Rule Writing Guide:

https://docs.snort.org

Thank you for your kind attention

SNORT3

Edoardo Mich
Davide Moletta
Tefera Addis Sisay

Network Security Lab 22/23