

Wykrywanie zagrożeń i reakcja na incydenty

Laboratorium 5

Tomasz Jarząbek 272279
Wiktoria Migasiewicz 272177

21.04.2025

Spis treści

1	Opis laboratorium	3
2	Rozwiązania	3
2.1	Przygotowanie	3
2.1.1	Przygotowanie teoretyczne	3
2.2	Sniffer i Logger	4
2.2.1	Instalacja Snorta	4
2.2.2	Sniffer	5
2.2.3	Logger	8
2.3	NIDS	11
2.4	Wnioski	18

1 Opis laboratorium

Laboratorium polegało na analizie i użyciu narzędzia badającego ruch sieciowy Snort w środowisku dwóch maszyn wirtualnych: Kali Linux (Kali VM)/Kali Live oraz Ubuntu. Narzędzia zostały najpierw zbadane i wypisano ich pewne właściwości, a używano ich do odrzucania ruchu między maszynami wirtualnymi i rejestracji wszelkich czynności w postaci logów systemowych wypisywanych na konsoli.

2 Rozwiązania

2.1 Przygotowanie

W celu rozpoczęcia laboratorium, najpierw przygotowano odpowiednie środowisko pracy, które składało się z dwóch maszyn wirtualnych: Kali Linux oraz Ubuntu. Ta wcześniejsza była już zainstalowana, a ta ostatnia została pobrana z oficjalnej strony i zainicjowana z odpowiednimi parametrami hardware'u. Wszystkie działają na systemie operacyjnym z rodziny Linux. Maszyny zostały umieszczone w tej samej sieci w celu zapewnienia komunikacji między nimi.

2.1.1 Przygotowanie teoretyczne

Merytoryczne przygotowanie do zajęć polegało na odpowiedzeniu na kilka konkretnych pytań dotyczących teorii wykorzystywanych narzędzi.

Proszę zapoznać się z dokumentacją narzędzia Snort i opisać:

- **tryby pracy Snorta: Sniffer, Packet Logger, NIDS**
- Wyróżnia się parę trybów pracy Snorta:
 - **Sniffer:** Po prostu wyczytuje pakiety z sieci i wypisuje je na konsoli w czasie rzeczywistym.
 - **Packet Logger:** Zapisuje przechwycone pakiety na dysku w postaci logów.
 - **NIDS:** Odbywa detekcję i analizę ruchu sieciowego oraz spośród nich wszystkich jest najbardziej rozbudowanym trybem i zezwala na największą konfigurację.
- **dyrektywy Snorta: alert, log, pass, drop, reject**
- Jedne z licznych dyrektyw (rules) Snorta to m.in.:
 - **Alert:** Załącza ostrzeżenie, gdy wykryta zostaje podejrzana aktywność.

- **Log:** Zapisuje przechwycone pakiety, które odpowiadają ustanowionym zasadom na dysku w postaci logów ku późniejszej analizie.
- **Pass:** Pozwala na przejście bez analizy ruchowi, który spełnia pewne wymagania.
- **Drop:** Natychmiast wyrzuca pakiet, bez dalszego wdawania się w jakąkolwiek interakcję ze źródłem ("upuszcza" pakiet, nie dając nadawcy informacji o tym, że pakiet najpierw dotarł)
- **Reject:** działa podobnie do drop, ale dodatkowo wysyła pakiet resetujący połączenie (TCP RST) lub ICMP Unreachable w przypadku protokołów innych niż TCP, zanim pakiet zostanie odrzucony, czyli wysyła odpowiedź (RST/ICMP) do nadawcy o odrzuceniu pakietu.

2.2 Sniffer i Logger

2.2.1 Instalacja Snorta

Polecenie:

Proszę zainstalować narzędzie Snort:

Wykorzystana komenda:

```
sudo apt install snort
```

Wynik:



```
kali@buntugit:~$ sudo snort -v
Running in packet dump mode

--== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
Acquiring network traffic from "enp0s3".
Decoding Ethernet

--== Initialization Complete ==--

,,_  -*> Snort! <*-
o" )~ Version 2.9.20 GRE (Build 82)
' '  By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
     Copyright (C) 2014-2022 Cisco and/or its affiliates. All rights reserved.
```

Rysunek 1: Instalacja Snorta.

Komentarz:

Instalacja przebiegła pomyślnie. Wersja 2.9.20 jest stabilnym wydaniem Snorta, odpowiednim do lokalnych testów i edukacji. Podczas instalacji system pyta o interfejs sieciowy – można go później zmienić ręcznie w konfiguracji lub przy uruchomieniu Snorta z parametrem -i.

2.2.2 Sniffer

Polecenie:

Proszę Skonfigurować Snorta w trybie sniffer.

- Wynik ma zawierać dane z warstwy aplikacji.
- Wynik ma zawierać dane z warstwy łącza danych.

Wykorzystane komendy:

ip addr – w celu sprawdzenia interfejsów maszyny

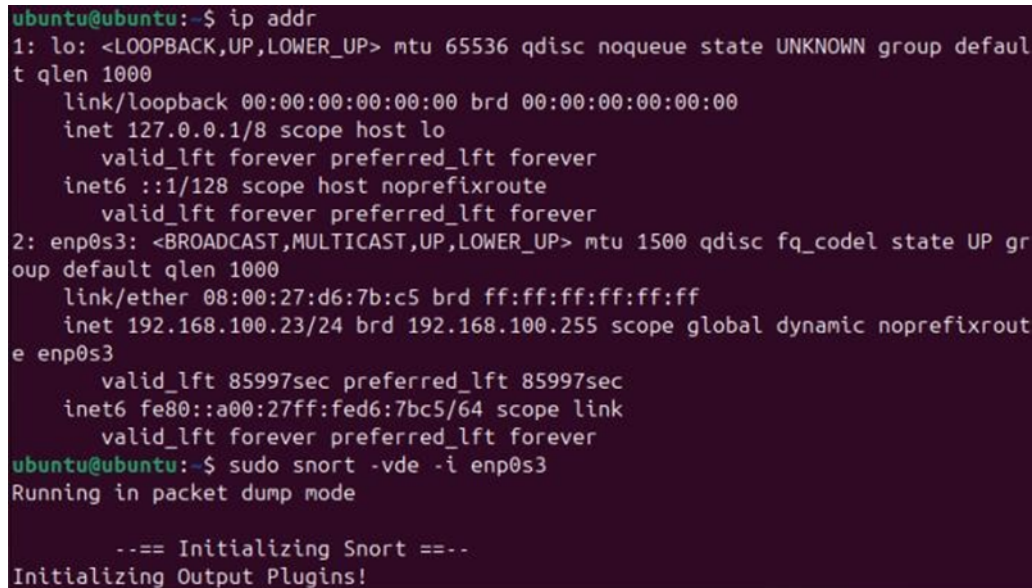
sudo snort -vde -i enp0s3 – uruchomienie Snorta

Komentarz:

- -v – pokazuje dane pakietów
- -d – pokazuje dane z warstwy aplikacji
- -e – pokazuje nagłówki warstwy łącza danych (Ethernet)
- -i – interfejs sieciowy enp0s3

Tryb sniffera jest przydatny przy analizie pakietów w czasie rzeczywistym. Dane są wypisywane bezpośrednio do terminala i nie są zapisywane na dysku.

Wynik:



```
ubuntu@ubuntu:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:d6:7b:c5 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.23/24 brd 192.168.100.255 scope global dynamic noprefixroute
        valid_lft 85997sec preferred_lft 85997sec
    inet6 fe80::a00:27ff:fed6:7bc5/64 scope link
        valid_lft forever preferred_lft forever
ubuntu@ubuntu:~$ sudo snort -vde -i enp0s3
Running in packet dump mode

--== Initializing Snort ==--
Initializing Output Plugins!
```

Rysunek 2: Wynik Sniffera.

Polecenie:

Uruchomić Snorta i odwiedzić trzy dowolne witryny internetowe

Odwiedzone strony to:

- www.wp.pl
- www.facebook.com

- www.wikipedia.pl

Polecenie:

Przedstawić fragment przechwyconego ruchu oraz podsumowanie

Podsumowanie:

```
=====
Run time for packet processing was 176.815317 seconds
Snort processed 23430 packets.
Snort ran for 0 days 0 hours 2 minutes 56 seconds
  Pkts/min:      11715
  Pkts/sec:       133
=====
Memory usage summary:
  Total non-mmapped bytes (arena):      790528
  Bytes in mapped regions (hblkhd):    23216128
  Total allocated space (uordblks):     687616
  Total free space (fordblks):          102912
  Topmost releasable block (keepcost): 101184
=====
Packet I/O Totals:
  Received:      23432
  Analyzed:      23430 ( 99.991%)
  Dropped:        0 (  0.000%)
  Filtered:       0 (  0.000%)
  Outstanding:   2 (  0.009%)
  Injected:       0
=====
```

Rysunek 3: Wynik Snorta na stronach 1.

```
Bad TTL:      0 (  0.000%)
SS G 1:       0 (  0.000%)
SS G 2:       0 (  0.000%)
Total:       23430
=====
Memory Statistics for File at:Sat Apr 12 17:54:10 2025
Total buffers allocated:      0
Total buffers freed:         0
Total buffers released:      0
Total file mempool:          0
Total allocated file mempool: 0
Total freed file mempool:    0
Total released file mempool: 0
Heap Statistics of file:
  Total Statistics:
    Memory in use:            0 bytes
    No of allocs:             0
    No of frees:              0
=====
Snort exiting
ubuntu@ubuntu:~$
```

Rysunek 4: Wynik Snorta na stronach 2.

Komentarz:

Dzięki opcji -d widoczne były niektóre dane z warstwy aplikacji (np. hosty, user-agenty). Połączenia szyfrowane (HTTPS) ograniczają widoczność pełnej treści – widoczna jest tylko faza ustanawiania połączenia (TLS handshake). Podczas odwiedzania stron zaobserwowano pakiety zawierające protokoły HTTP, TLS oraz DNS. Dane widoczne w terminalu zawierały adresy MAC, IP, porty

źródłowe i docelowe, oraz fragmenty zapytań HTTP i odpowiedzi. Domeny stron internetowych zostały sprawdzone za pomocą narzędzia nslookup.

Fragmenty przechwyconego ruchu:

Strona 1:

```
WARNING: No preprocessors configured for policy 0.
04/12-18:06:46.245087 64:C3:94:E7:14:EC -> 08:00:27:D6:7B:C5 type:0x800 len:0x71
212.77.98.9:443 -> 192.168.100.23:36616 TCP TTL:55 TOS:0x0 ID:51300 IpLen:20 DgmLen:99 DF
***A*** Seq: 0xD2D249AE Ack: 0xB24CA8CA Win: 0x15 TcpLen: 32
TCP Options (3) => NOP NOP TS: 1556919766 896558479
17 03 03 00 2A 5F B7 6B 7E 6C E6 9A AF A5 E1 88 ....*_k-l.....
94 D4 F5 65 35 71 40 EB 5B 37 50 02 BB 1F E3 0B ...e5q0.[7P.....
E7 D4 FC 1B 4E 7D 8D BE 64 32 5F 6F A2 67 57 ....N}..d2_o.gW

=====

04/12-18:06:46.346066 08:00:27:D6:7B:C5 -> 64:C3:94:E7:14:D8 type:0x800 len:0x42
192.168.100.23:36616 -> 212.77.98.9:443 TCP TTL:64 TOS:0x0 ID:22345 IpLen:20 DgmLen:52 DF
***A*** Seq: 0xD2D249AE Ack: 0xB24CA8F9 Win: 0x1877 TcpLen: 32
TCP Options (3) => NOP NOP TS: 896558499 1556919764

=====
```

Rysunek 5: Wynik Snorta na stronie 1.

```
ubuntu@ubuntu:~$ nslookup 212.77.98.9
9.98.77.212.in-addr.arpa          name = www.wp.pl.

Authoritative answers can be found from:

ubuntu@ubuntu:~$
```

Rysunek 6: Wynik Snorta na stronie 2.

Strona 2:

```
=====
WARNING: No preprocessors configured for policy 0.
04/12-18:12:39.503182 64:C3:94:E7:14:EC -> 08:00:27:D6:7B:C5 type:0x800 len:0x4FA
57.144.110.1:443 -> 192.168.100.23:33838 UDP TTL:53 TOS:0x0 ID:1 IpLen:20 DgmLen:1260 DF
***A*** Seq: 0xD2D249AE Ack: 0xB24CA8F9 Win: 0x1877 UdpLen: 1260
52 9C 24 4C BD C6 29 12 20 3C D5 CB 92 51 6D 06 R.$L..). <...Qm.
73 D9 46 68 E1 F9 1C C1 4C E9 2E DF 35 60 37 9B s.Fh....L...S`7.
9A 25 9E 7A DD 5A 9F CA 34 32 F8 5D A3 3C 30 41 .%.z.Z..42.].<0A
98 9C D9 FC AD 3F 20 4F E0 EF 89 58 EB 4D 2D FC .....? O...X.M-.
9F 03 3D 84 3C 9D 8F 67 E8 AD 79 1C 44 9F 00 4E ..=<..g..y.D..N
```

Rysunek 7: Wynik Snorta na stronie 2.

```
ubuntu@ubuntu:~$ nslookup 57.144.110.1
1.110.144.57.in-addr.arpa      name = edge-star-mini-shv-01-waw2.facebook.com.

Authoritative answers can be found from:

ubuntu@ubuntu:~$
```

Rysunek 8: Wynik Snorta na stronie 2.

Strona 3:

```
WARNING: No preprocessors configured for policy 0.
04/12-18:40:31.720847 64:C3:94:E7:14:EC -> 08:00:27:D6:7B:C5 type:0x800 len:0x32
185.15.59.224:443 -> 192.168.100.23:34536 TCP TTL:54 TOS:0x0 ID:14851 IpLen:20 D
***AP*** Seq: 0xD0CD0CB5 Ack: 0xFD6B14B6 Win: 0x53 TcpLen: 32
TCP Options (3) => NOP NOP TS: 3148244140 2925506248
17 03 03 02 F0 CD 5A 67 72 C2 8B 6E 50 F5 89 C6
```

Rysunek 9: Wynik Snorta na stronie 3.

```
ubuntu@ubuntu:~$ nslookup 185.15.59.224
224.59.15.185.in-addr.arpa      name = text-lb.esams.wikimedia.org.

Authoritative answers can be found from:

ubuntu@ubuntu:~$
```

Rysunek 10: Wynik Snorta na stronie 3.

2.2.3 Logger

Polecenie:

Utworzyć katalog dla plików logów **Wykorzystane komendy:**

```
sudo mkdir /var/log/snort
```

```
sudo chmod 777 /var/log/snort
```

Wynik:

```
kali@ubuntugit:~$ sudo mkdir /var/log/snort
mkdir: cannot create directory '/var/log/snort': File exists
kali@ubuntugit:~$ sudo chmod 777 /var/log/snort
kali@ubuntugit:~$
```

Rysunek 11: Stworzenie folderów.

Komentarz:

Folder `/var/log/snort` jest domyślnym miejscem zapisu logów Snorta. Nadano mu pełne prawa, aby uniknąć problemów z dostępem do zapisu podczas działania Snorta. W rzeczywistym środowisku zaleca się bardziej restrykcyjne uprawnienia.

Polecenie:

Skonfigurować Snorta w trybie packet logger.

- Wynik ma zostać zapisany w postaci czytelnej dla programu Wireshark
- Uruchomić Snorta i odwiedzić trzy dowolne witryny internetowe

Odwiedzone strony internetowe:

- www.wp.pl
- www.onet.pl
- www.facebook.com

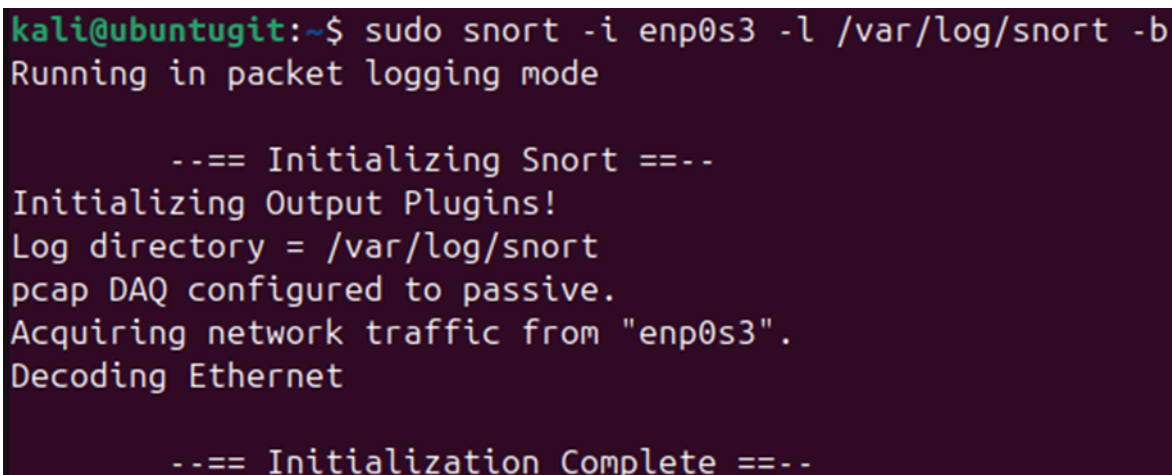
Wykorzystana komenda:

```
sudo snort -i enp0s3 -l /var/log/snort -b
```

- `-i enp0s3` – interfejs sieciowy
- `-l /var/log/snort` – miejsce zapisu logów
- `-b` – tryb binarny, czyli .pcap

Ten tryb umożliwia zapisywanie ruchu sieciowego do pliku binarnego (.pcap), który można później otworzyć w Wiresharku. To bardziej użyteczne niż tryb sniffera, ponieważ umożliwia dokładną analizę po fakcie oraz eksport danych.

Wynik:



```
kali@ubuntugit:~$ sudo snort -i enp0s3 -l /var/log/snort -b
Running in packet logging mode

--== Initializing Snort ==--
Initializing Output Plugins!
Log directory = /var/log/snort
pcap DAQ configured to passive.
Acquiring network traffic from "enp0s3".
Decoding Ethernet

--== Initialization Complete ==--
```

Rysunek 12: Wynik Snorta zapisywany do Wireshark.

Polecenie:

Zaprezentować fragment rezultatu w programie Wireshark

Wykorzystana komenda:

```
sudo wireshark /var/log/snort/snort.log.*
```

Wynik:

```
kali@ubuntugit:~$ sudo wireshark /var/log/snort/snort.log.*
```

Rysunek 13: Włączenie Wiresharka z plikiem logów.

Przechwycone pakiety:

snort.log.1744532144							
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help							
http							
No.	Time	Source	Destination	Protocol	Length	Info	
48	1.909735	192.168.100.23	13.227.146.122	HTTP	408	GET /	
52	1.922980	13.227.146.122	192.168.100.23	HTTP	1332	HTTP/	
56	2.009912	192.168.100.23	13.227.146.122	HTTP	425	GET /	
58	2.023938	13.227.146.122	192.168.100.23	HTTP	1332	HTTP/	
184	12.767934	192.168.100.23	13.227.146.66	HTTP	351	GET /	
215	12.803995	13.227.146.66	192.168.100.23	HTTP	597	HTTP/	
426	13.224302	192.168.100.23	104.81.99.218	OCSP	507	Reque	
429	13.239398	104.81.99.218	192.168.100.23	OCSP	940	Respc	
783	31.450604	192.168.100.23	57.144.110.141	HTTP	408	GET /	
785	31.464380	57.144.110.141	192.168.100.23	HTTP	264	HTTP/	
804	31.520148	192.168.100.23	104.81.99.218	OCSP	505	Reque	
806	31.539939	104.81.99.218	192.168.100.23	OCSP	941	Respc	
820	31.998999	192.168.100.23	34.107.221.82	HTTP	367	GET /	
822	32.014041	34.107.221.82	192.168.100.23	HTTP	364	HTTP/	
831	32.038315	192.168.100.23	34.107.221.82	HTTP	384	GET /	
833	32.052527	34.107.221.82	192.168.100.23	HTTP	282	HTTP/	
889	43.740849	192.168.100.23	212.77.98.9	HTTP	405	GET /	

Rysunek 14: Przechwycone pakiety w Wiresharku.

Weryfikacja adresów odwiedzonych stron internetowych:

```
kali@ubuntugit:~$ nslookup 212.77.98.9
9.98.77.212.in-addr.arpa      name = www.wp.pl.

Authoritative answers can be found from:

kali@ubuntugit:~$ nslookup 57.144.110.141
141.110.144.57.in-addr.arpa   name = edge-star-shv-01-waw2.facebook.com.

Authoritative answers can be found from:

kali@ubuntugit:~$ nslookup 13.227.146.122
122.146.227.13.in-addr.arpa   name = server-13-227-146-122.waw51.r.cloudfront.net.

Authoritative answers can be found from:
```

Rysunek 15: Zweryfikowane adresy IP.

```
kali@ubuntugit:~$ nslookup onet.pl
Server:      127.0.0.53
Address:     127.0.0.53#53

Non-authoritative answer:
Name:   onet.pl
Address: 13.227.146.122
```

Rysunek 16: Zweryfikowane adresy IP.

Komentarz:

Wireshark umożliwia zaawansowaną analizę pakietów – filtrację po adresach IP, protokołach, portach, a nawet treści (jeśli pakiet jest nieszyfrowany). Ułatwia również identyfikację odwiedzanych witryn poprzez analizę zapytań DNS lub nagłówek SNI w pakietach TLS.

2.3 NIDS

Należało skonfigurować Snorta w trybie NIDS.

Stworzono plik `/etc/snort/rules/custom.rules` oraz do pliku `snort.conf`, dodając do niego linię: `.`

Poprawność zmienionej konfiguracji sprawdzono komendą `"snort -T -c /etc/snort/snort.conf"`.

```
Total snort Fixed Memory Cost - MaxRss:105168
Snort successfully validated the configuration!
Snort exiting
ubuntu@ubuntu-VirtualBox:/etc/snort$
```

Rysunek 17: Sprawdzona konfiguracja Snorta.

Utworzono i przetestowano konkretne reguły. Testy reguł wykonano z maszyny Kali Live. Adres maszyny Ubuntu to 192.168.100.37.

- detekcja skanowania portów od 100 do 1000, test można wykonać w oparciu o narzędzie `nmap`.

Dodano do pliku `custom.rules` zasadę dot. skanowania portów od 100 do 1000, czyli:

```
alert tcp any any -> any 100:1000 (msg:"Port scan detected in range 100-1000";
flags:S; sid:1000001; rev:1;), gdzie:
```

msg - wiadomość wyświetlana na terminalu

alert - generuje alert

tcp - Reguła dotyczy tylko pakietów TCP

any any - źródłowy adres IP i port: dowolny

-> - Ruch skierowany do...

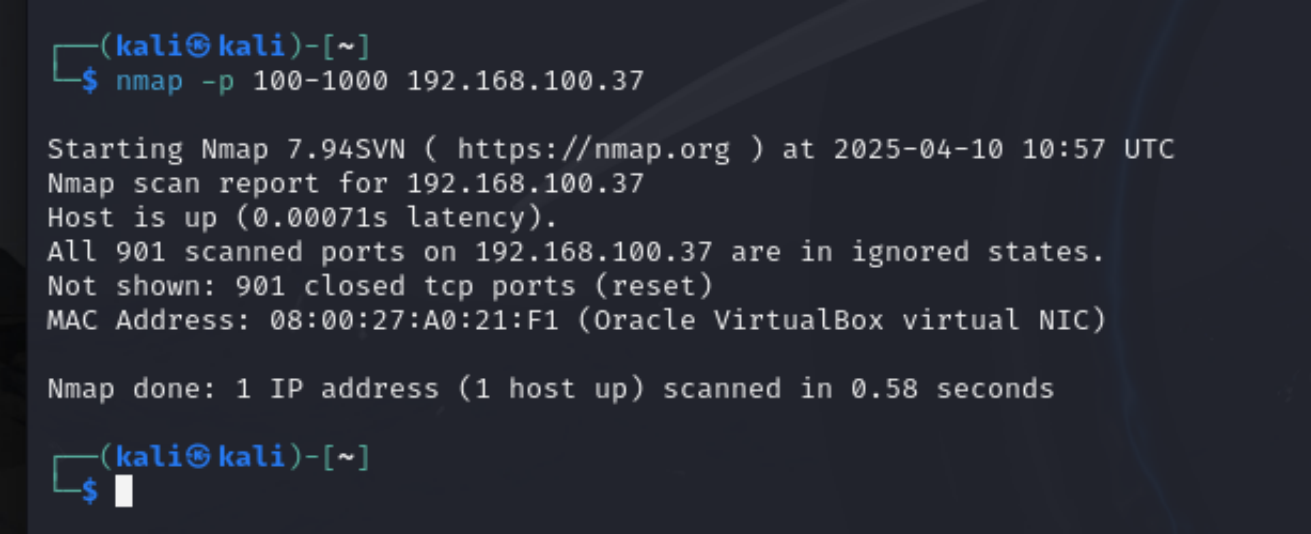
any 100:1000 - docelowy adres IP dowolny, ale port w zakresie 100–1000

msg:"... treść komunikatu alertu

flags:S - filtruje tylko pakiety z flagą SYN – typowe dla skanowania portów

sid:1000001 - ID reguły

rev:1 - wersja reguły



```
(kali㉿kali)-[~]  
$ nmap -p 100-1000 192.168.100.37  
  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-10 10:57 UTC  
Nmap scan report for 192.168.100.37  
Host is up (0.00071s latency).  
All 901 scanned ports on 192.168.100.37 are in ignored states.  
Not shown: 901 closed tcp ports (reset)  
MAC Address: 08:00:27:A0:21:F1 (Oracle VirtualBox virtual NIC)  
  
Nmap done: 1 IP address (1 host up) scanned in 0.58 seconds  
  
(kali㉿kali)-[~]  
$
```

Rysunek 18: Wykonanie nmap na Kali Live w stronę Ubuntu.

```
ubuntu@ubuntu-VirtualBox:/etc/snort/rules$ sudo snort -q -A console -i enp0s3 -c
--custom.rules--
04/10-12:57:11.857326  [**] [1:1000001:1] Port scan detected in range 100-1000 [
**] [Priority: 0] {TCP} 192.168.100.32:55999 -> 192.168.100.37:143
04/10-12:57:11.857331  [**] [1:1000001:1] Port scan detected in range 100-1000 [
**] [Priority: 0] {TCP} 192.168.100.32:55999 -> 192.168.100.37:445
04/10-12:57:11.857332  [**] [1:1000001:1] Port scan detected in range 100-1000 [
**] [Priority: 0] {TCP} 192.168.100.32:55999 -> 192.168.100.37:113
04/10-12:57:11.857334  [**] [1:1000001:1] Port scan detected in range 100-1000 [
**] [Priority: 0] {TCP} 192.168.100.32:55999 -> 192.168.100.37:995
04/10-12:57:11.857335  [**] [1:1000001:1] Port scan detected in range 100-1000 [
**] [Priority: 0] {TCP} 192.168.100.32:55999 -> 192.168.100.37:587
04/10-12:57:11.858799  [**] [1:1000001:1] Port scan detected in range 100-1000 [
**] [Priority: 0] {TCP} 192.168.100.32:55999 -> 192.168.100.37:554
04/10-12:57:11.858800  [**] [1:1000001:1] Port scan detected in range 100-1000 [
**] [Priority: 0] {TCP} 192.168.100.32:55999 -> 192.168.100.37:199
04/10-12:57:11.859831  [**] [1:1000001:1] Port scan detected in range 100-1000 [
**] [Priority: 0] {TCP} 192.168.100.32:55999 -> 192.168.100.37:256
04/10-12:57:11.859832  [**] [1:1000001:1] Port scan detected in range 100-1000 [
```

Rysunek 19: Otrzymane powiadomienia o próbach skanowania portów.

Widać, iż otrzymano informację o skanowaniu określonego portu, a wiadomość formatowana jest tak, jak sami to zrobiliśmy.

- detekcja ataków brute-force na usługę SSH (próg działania: 5-krotne próby podania hasła w czasie 30 sekund), test można wykonać w oparciu o narzędzie **hydra**.

Dodano do pliku **custom.rules** zasadę dot. skanowania ewentualnej próby brute-force'a SSH, mianowicie:

alert tcp any any -> any 22 (msg:"SSH brute-force attempt"; detection_filter:track by_src, count 5, seconds 30; sid:1000002; rev:1;), gdzie:

any any -> any 22 - dowolny klient, atakujący port 22 (SSH) serwera.

detection_filter:track by_src, count 5, seconds 30; - jeśli ta sama źródłowa IP zrobi 5 prób w 30 sekund, zgłoś alert.

sid:1000002 - ID.

```
(kali㉿kali)-[/usr/share/wordlists]
$ hydra -l root -P /usr/share/wordlists/rockyou.txt ssh://192.168.100.37

Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in
military or secret service organizations, or for illegal purposes (this is n
on-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-04-10 11:
07:16
[WARNING] Many SSH configurations limit the number of parallel tasks, it is r
ecommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1
/p:14344399), ~896525 tries per task
[DATA] attacking ssh://192.168.100.37:22/
```

Rysunek 20: Atak brute-force SSH na Ubuntu z Kali Live za pomocą Hydra.

```
ubuntu@ubuntu-VirtualBox:/etc/snort/rules$ sudo snort -q -A console -i enp0s3 -c
custom.rules
04/10-13:07:21.726581  [**] [1:1000002:1] SSH brute-force attempt [**] [Priority
: 0] {TCP} 192.168.100.32:60706 -> 192.168.100.37:22
04/10-13:07:21.737128  [**] [1:1000002:1] SSH brute-force attempt [**] [Priority
: 0] {TCP} 192.168.100.32:60604 -> 192.168.100.37:22
04/10-13:07:21.751412  [**] [1:1000002:1] SSH brute-force attempt [**] [Priority
: 0] {TCP} 192.168.100.32:60684 -> 192.168.100.37:22
04/10-13:07:36.486880  [**] [1:1000002:1] SSH brute-force attempt [**] [Priority
: 0] {TCP} 192.168.100.32:36360 -> 192.168.100.37:22
04/10-13:07:36.525820  [**] [1:1000002:1] SSH brute-force attempt [**] [Priority
: 0] {TCP} 192.168.100.32:36420 -> 192.168.100.37:22
```

Rysunek 21: Otrzymane alerty dot. próby brute-force'a z adresu IP Kali Live.

Snort wykrył próbę brute-force'a i wyświetlił źródło hosta atakującego wraz z portami, z których atakowano.

- detekcja odpowiedzi błędu 404 serwera http, test można wykonać w oparciu o połączenie z witryną <http://www.google.com/404>.

Dodano zasadę dotyczącą wykrycia zwrotu błędu 404 przez stronę http.

```
alert tcp any any -> any 80 (msg:"HTTP 404 Not Found detected"; content:"HTTP/1.1
404";content:"Not Found"; sid:1000003; rev:1;)
```

any any -> any 80 - ruch do portu 80 (HTTP).

content:"HTTP/1.1 404";content:"Not Found szukamy wzorca błędu 404 w odpowiedzi.

```

^C^C*** Caught Int-Signal
ubuntu@ubuntu-VirtualBox:/etc/snort/rules$ sudo nano custom.rules
ubuntu@ubuntu-VirtualBox:/etc/snort/rules$ sudo snort -A console -q -i enp0s3 -c /
etc/snort/rules/custom.rules
^C^C*** Caught Int-Signal
ubuntu@ubuntu-VirtualBox:/etc/snort/rules$ sudo nano custom.rules
ubuntu@ubuntu-VirtualBox:/etc/snort/rules$ sudo snort -A console -q -i enp0s3 -c /
etc/snort/rules/custom.rules
^C*** Caught Int-Signal
ubuntu@ubuntu-VirtualBox:/etc/snort/rules$ ^C
ubuntu@ubuntu-VirtualBox:/etc/snort/rules$ sudo nano custom.rules
ubuntu@ubuntu-VirtualBox:/etc/snort/rules$ sudo snort -A console -q -i enp0s3 -c /
etc/snort/rules/custom.rules
04/10-13:50:09.958308  [**] [1:1000006:1] HTTP 404 Not Found detected [**] [Priori
ty: 0] {TCP} 193.239.44.202:80 -> 192.168.100.37:46288
04/10-13:50:10.171115  [**] [1:1000006:1] HTTP 404 Not Found detected [**] [Priori
ty: 0] {TCP} 193.239.44.202:80 -> 192.168.100.37:46300

```

Rysunek 22: Wykryty błąd HTTP 404.

Ponieważ strona google posiadająca błąd 404 jest szyfrowana (HTTPS), żaden błąd nie zostałby przez tę zasadę wykryty, dlatego skorzystano ze strony forum rybackiego, która używa protokołu HTTP. Na niej wprowadzono losowy ciąg znaków w wyszukiwarce stron, powodując wystąpienie owego błędu. W tym momencie, Snort wykrył owy błąd i zaprezentował to na konsoli.

- **detekcja flag w protokole TCP: URG, PSH, FIN (razem), test można wykonać w oparciu o narzędzie nmap.**

Następna zasada dotyczyła wykrycia konkretnych flag w protokole:

```

alert tcp any any -> any any (msg:"Nmap TCP scan with URG+PSH+FIN flags";
flags:UPF; sid:1000004; rev:1;)

```

flags:UPF - wymagane są flagi: URG, PSH, FIN jednocześnie. any any -> any any - działa niezależnie od portów/adresów.


```
(kali㉿kali)-[~]  
$ nmap -sX 192.168.100.37  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-10 11:24 UTC  
Nmap scan report for 192.168.100.37  
Host is up (0.00074s latency).  
Not shown: 999 closed tcp ports (reset)  
PORT      STATE      SERVICE  
22/tcp    open|filtered ssh  
MAC Address: 08:00:27:A0:21:F1 (Oracle VirtualBox virtual NIC)  
  
Nmap done: 1 IP address (1 host up) scanned in 1.63 seconds  
  
(kali㉿kali)-[~]  
$ nmap -sU 192.168.100.37  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-10 11:24 UTC  
  
(kali㉿kali)-[~]  
$ nmap -sP 192.168.100.37  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-10 11:25 UTC  
Nmap scan report for 192.168.100.37  
Host is up (0.00078s latency).  
MAC Address: 08:00:27:A0:21:F1 (Oracle VirtualBox virtual NIC)  
Nmap done: 1 IP address (1 host up) scanned in 0.07 seconds
```

Rysunek 23: Wysyłane z Kali Live skany nmap z flagami.

```
ubuntu@ubuntu-VirtualBox:/etc/snort/rules$ sudo snort -q -A console -i enp0s3 -c  
custom.rules  
04/10-13:24:43.573765  [**] [1:1000004:1] Suspicious TCP Flags: URG PSH FIN [**]  
[Priority: 0] {TCP} 192.168.100.32:47389 -> 192.168.100.37:1025  
04/10-13:24:43.574474  [**] [1:1000004:1] Suspicious TCP Flags: URG PSH FIN [**]  
[Priority: 0] {TCP} 192.168.100.32:47389 -> 192.168.100.37:25  
04/10-13:24:43.574475  [**] [1:1000004:1] Suspicious TCP Flags: URG PSH FIN [**]  
[Priority: 0] {TCP} 192.168.100.32:47389 -> 192.168.100.37:5900  
04/10-13:24:43.575728  [**] [1:1000004:1] Suspicious TCP Flags: URG PSH FIN [**]  
[Priority: 0] {TCP} 192.168.100.32:47389 -> 192.168.100.37:139  
04/10-13:24:43.575730  [**] [1:1000004:1] Suspicious TCP Flags: URG PSH FIN [**]  
[Priority: 0] {TCP} 192.168.100.32:47389 -> 192.168.100.37:993  
04/10-13:24:43.576966  [**] [1:1000004:1] Suspicious TCP Flags: URG PSH FIN [**]  
[Priority: 0] {TCP} 192.168.100.32:47389 -> 192.168.100.37:143  
04/10-13:24:43.576968  [**] [1:1000004:1] Suspicious TCP Flags: URG PSH FIN [**]  
[Priority: 0] {TCP} 192.168.100.32:47389 -> 192.168.100.37:587  
04/10-13:24:43.576968  [**] [1:1000004:1] Suspicious TCP Flags: URG PSH FIN [**]  
[Priority: 0] {TCP} 192.168.100.32:47389 -> 192.168.100.37:80  
04/10-13:24:43.577690  [**] [1:1000004:1] Suspicious TCP Flags: URG PSH FIN [**]  
[Priority: 0] {TCP} 192.168.100.32:47389 -> 192.168.100.37:8888
```

Rysunek 24: Wysłana wiadomość dot. wykrycia skanu z flagami.

Flaga URG (URGent) sygnalizuje, że przesyłane dane są pilne i powinny być natychmiast przetworzone po stronie odbiorcy, bez konieczności oczekiwania na zakończenie przetwarzania wcześniejszych segmentów.

Flaga PSH (Push) pełni podobną funkcję – zapewnia, że dane zostaną od razu przekazane do aplikacji odbiorczej, a nie zatrzymane w buforze TCP.

W skrócie - PSH wymusza natychmiastowe przesłanie danych do aplikacji odbiorczej, URG oznacza dane jako pilne i wykorzystuje pole wskaźnika pilności, a FIN informuje o zakończeniu połączenia.

Ruch z tymi flagami może być podejrzany, ponieważ często jest charakterystyczny dla skanów portów, prób obejścia zabezpieczeń lub tunelowania danych.

Snort wykrył wszystkie przesyłane skany z flagami jednocześnie (na nmap, wiadomości wysyłane były osobno, ponieważ pewne flagi nie współpracowały ze sobą).

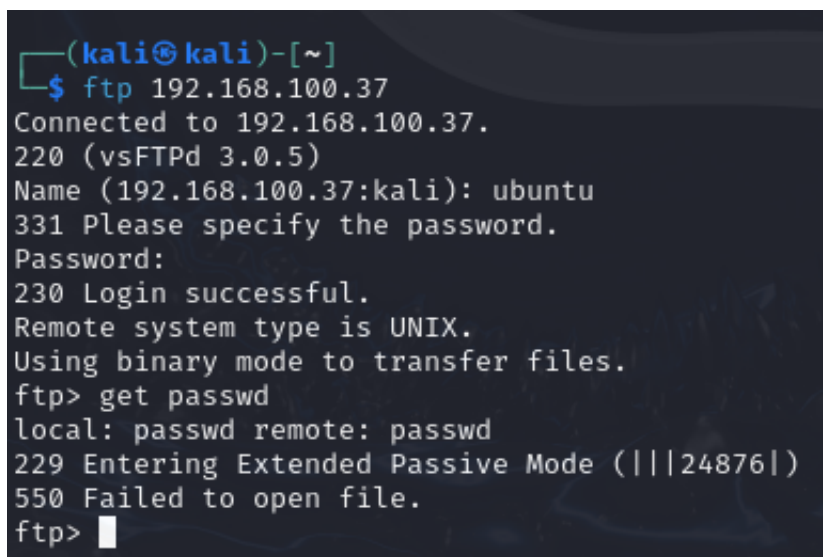
- **detekcja pobrania pliku passwd w oparciu o protokół FTP, test można wykonać przy użyciu narzędzia ftp i serwera vsftpd.**

Ostatnia zasada to zasada dotycząca pobierania pliku `/etc/passwd` zawierającego listę kont użytkowników w systemie.

alert tcp any any -> any 21 (msg:"FTP passwd file download attempt"; content:"passwd"; sid:1000005; rev:1;)

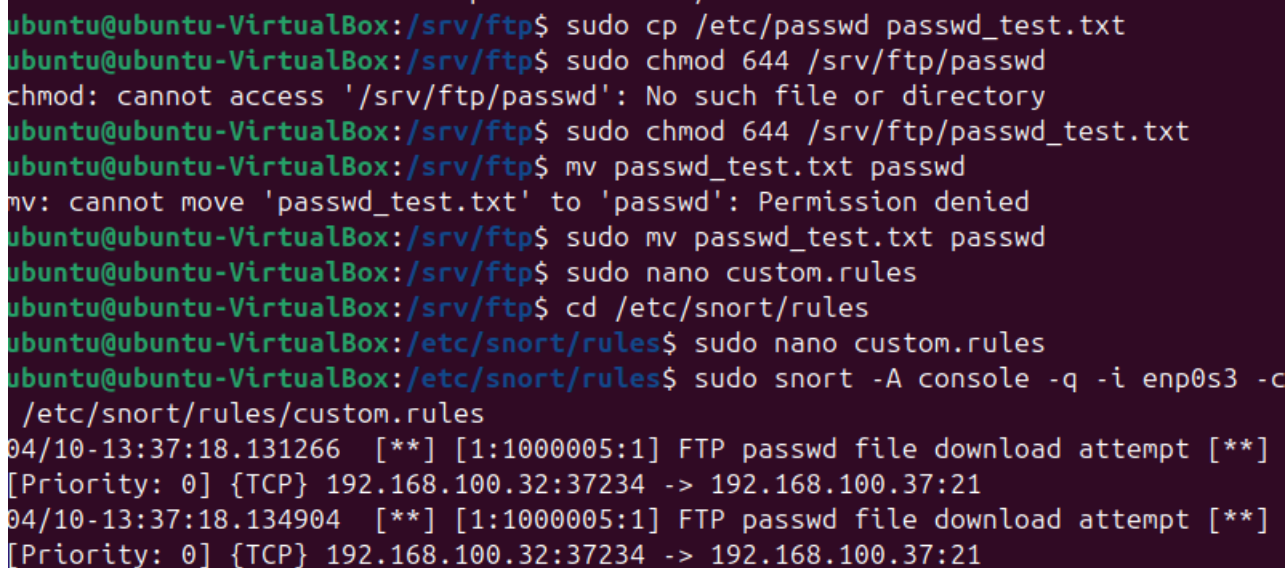
any any -> any 21 - pakiet trafia do portu FTP (21).

content:"passwd" w treści pakietu FTP klient wysyła polecenie pobrania pliku *passwd*.



```
(kali@kali)-[~]
$ ftp 192.168.100.37
Connected to 192.168.100.37.
220 (vsFTPD 3.0.5)
Name (192.168.100.37:kali): ubuntu
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> get passwd
local: passwd remote: passwd
229 Entering Extended Passive Mode (|||24876|)
550 Failed to open file.
ftp>
```

Rysunek 25: Próba pobrania pliku passwd.



```
ubuntu@ubuntu-VirtualBox:/srv/ftp$ sudo cp /etc/passwd passwd_test.txt
ubuntu@ubuntu-VirtualBox:/srv/ftp$ sudo chmod 644 /srv/ftp/passwd
chmod: cannot access '/srv/ftp/passwd': No such file or directory
ubuntu@ubuntu-VirtualBox:/srv/ftp$ sudo chmod 644 /srv/ftp/passwd_test.txt
ubuntu@ubuntu-VirtualBox:/srv/ftp$ mv passwd_test.txt passwd
mv: cannot move 'passwd_test.txt' to 'passwd': Permission denied
ubuntu@ubuntu-VirtualBox:/srv/ftp$ sudo mv passwd_test.txt passwd
ubuntu@ubuntu-VirtualBox:/srv/ftp$ sudo nano custom.rules
ubuntu@ubuntu-VirtualBox:/etc/snort/rules$ sudo nano custom.rules
ubuntu@ubuntu-VirtualBox:/etc/snort/rules$ sudo snort -A console -q -i enp0s3 -c
/etc/snort/rules/custom.rules
04/10-13:37:18.131266  [**] [1:1000005:1] FTP passwd file download attempt [**]
[Priority: 0] {TCP} 192.168.100.32:37234 -> 192.168.100.37:21
04/10-13:37:18.134904  [**] [1:1000005:1] FTP passwd file download attempt [**]
[Priority: 0] {TCP} 192.168.100.32:37234 -> 192.168.100.37:21
```

Rysunek 26: Wykrycie próby pobrania pliku passwd.

Najpierw próbowano z Kali Live pobrać plik `/etc/passwd`, ale nie było to możliwe nawet z komendą `sudo` (co dobrze świadczy o zabezpieczeniach Ubuntu), dlatego skopiowano plik `/etc/passwd` do pliku tymczasowego "passwd" w `/srv/ftp`. Dzięki temu Kali Live mógł próbować pobrać ten plik (co również się nie udało), a zostało to odnotowane przez Snort.

2.4 Wnioski

Snort jest wszechstronnym narzędziem, służy zarówno jako IPS oraz IDS ze względu na możliwości tworzenia własnych reguł w zależności od potrzeb administratora systemu. Pozwala na blokowanie i zezwalanie ruchu na wielu portach oraz wykrywanie podejrzanych zachowań i schematów w wielu protokołach. Dodatkowo wyniki tego narzędzia mogą być zapisywane do pliku `.pcap` w celu późniejszej analizy w programie Wireshark.