

W sprawozdaniu zamieść zrzuty ekranu z istotnych etapów oraz odpowiedzi na pytania.



Taki symbol oznacza, że trzeba w sprawozdaniu dodać zrzut ekranu (najczęściej 1) z wyniku działania polecenie.



Taki symbol oznacza, że należy dodać opis (najczęściej 1 zdanie) z wyniku działania polecenia.

Wstęp

Wireshark sam w sobie jest rozbudowanym narzędziem do analizy sieciowej, jednak istnieją sytuacje, w których jego użycie nie jest tak efektywne jak skorzystanie z narzędzi pomocniczych. Dotyczy to zwłaszcza zaawansowanej automatyzacji przechwytywania ruchu lub analizy bardzo dużych zbiorów danych. W praktyce żaden program pracujący bezpośrednio na plikach PCAP nie poradzi sobie w pełni z takimi zadaniami ze względu na ograniczenia mocy obliczeniowej procesora i dostępnej pamięci RAM. W takich przypadkach zejście do poziomu linii poleceń często pozwala uzyskać lepsze rezultaty. To właśnie dlatego wraz z Wiresharkiem instalowane są także narzędzia tekstowe, takie jak TShark, oraz edytor pakietów Editcap, zapewniające większą elastyczność i wydajność w pracy z danymi sieciowymi.

```
root@kali:~# tshark -r packets.pcap -T text
Running as user "root" and group "root". This could be dangerous.
  1 0.000000000 192.168.0.6 → 104.28.6.89 ICMP 98 Echo (ping) request id=0x1b86, seq=1541/12
86, ttl=64
  2 0.209711164 104.28.6.89 → 192.168.0.6 ICMP 98 Echo (ping) reply id=0x1b86, seq=1541/12
86, ttl=54 (request in 1)
  3 1.001906657 192.168.0.6 → 104.28.6.89 ICMP 98 Echo (ping) request id=0x1b86, seq=1542/15
42, ttl=64
  4 1.192770973 104.28.6.89 → 192.168.0.6 ICMP 98 Echo (ping) reply id=0x1b86, seq=1542/15
42, ttl=54 (request in 3)
  5 2.003365632 192.168.0.6 → 104.28.6.89 ICMP 98 Echo (ping) request id=0x1b86, seq=1543/17
98, ttl=64
  6 2.434560259 104.28.6.89 → 192.168.0.6 ICMP 98 Echo (ping) reply id=0x1b86, seq=1543/17
98, ttl=54 (request in 5)
  7 3.003769942 192.168.0.6 → 104.28.6.89 ICMP 98 Echo (ping) request id=0x1b86, seq=1544/20
54, ttl=64
  8 3.347729784 104.28.6.89 → 192.168.0.6 ICMP 98 Echo (ping) reply id=0x1b86, seq=1544/20
54, ttl=54 (request in 7)
  9 4.003967430 192.168.0.6 → 104.28.6.89 ICMP 98 Echo (ping) request id=0x1b86, seq=1545/23
10, ttl=64
 10 4.163455725 104.28.6.89 → 192.168.0.6 ICMP 98 Echo (ping) reply id=0x1b86, seq=1545/23
10, ttl=54 (request in 9)
```

I. Tshark

TShark to narzędzie wiersza poleceń dostarczane razem z Wiresharkiem. Działa podobnie jak Wireshark, lecz nie posiada graficznego interfejsu użytkownika (GUI). Dzięki temu idealnie sprawdza się w środowiskach serwerowych, skryptach automatyzujących oraz podczas przechwytywania ruchu w czasie rzeczywistym.

1. Uruchom maszynę wirtualną z Kali Linux i przejdź do terminala.



2. Sprawdź nazwę swojego interfejsu sieciowego (najczęściej `eth0`):

```
ip addr
```



3. Uruchom TSharka i sprawdź, czy przechwytuje pakiety (wejdź na dowolną stronę internetową):

```
tshark
```

```
(kali@kali)~$ tshark
Capturing on 'eth0'
1 0.000000000 192.168.217.133 → 162.159.200.1 NTP 90 NTP Version 4, client
2 0.019901522 VMware_ec:89:15 → Broadcast ARP 60 Who has 192.168.217.133? Tell 192.168.217.2
3 0.020004171 VMware_23:e2:38 → VMware_ec:89:15 ARP 42 192.168.217.133 is at 00:0c:29:23:e2:38
4 0.020523345 162.159.200.1 → 192.168.217.133 NTP 90 NTP Version 4, server
5 5.192042825 VMware_23:e2:38 → VMware_ec:89:15 ARP 42 Who has 192.168.217.2? Tell 192.168.217.133
6 5.192178792 VMware_ec:89:15 → VMware_23:e2:38 ARP 60 192.168.217.2 is at 00:50:56:ec:89:15
7 5.350207437 192.168.217.133 → 192.168.217.2 DNS 88 Standard query 0xc475d A contile.services.mozilla.com
8 5.350242038 192.168.217.133 → 192.168.217.2 DNS 88 Standard query 0xc353 AAAA contile.services.mozilla.com
9 5.352155024 192.168.217.2 → 192.168.217.133 DNS 417 Standard query response 0xc475d A contile.services.mozilla.com A 34.36.137.203 NS ns-1655.awsdns-14.co.uk NS ns-679.awsdns-20.net NS ns-258.awsdns-32.com NS ns-1471.awsdns-55.org A 205.251.193.2 A 205.251.194.167 A 205.251.197.191 A 205.251.198.119 AAAA 2600:9000:5301:200::1 AAAA 2600:9000:5302:a700::1 AAAA 2600:9000:5305:bf00::1 AAAA 2600:9000:5306:7700::1
10 5.352155322 192.168.217.2 → 192.168.217.133 DNS 169 Standard query response 0xc353 AAAA contile.services.mozilla.com SOA ns-679.awsdns-20.net
11 5.352694774 192.168.217.133 → 34.36.137.203 TCP 74 41852 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2602174091 TSecr=0 WS=128
12 5.383839936 34.36.137.203 → 192.168.217.133 TCP 60 443 → 41852 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
13 5.383878232 192.168.217.133 → 34.36.137.203 TCP 54 41852 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
```

Zatrzymaj działanie TSharka za pomocą `Ctrl + C`

4. Przechwyć wszystkie pakiety z wybranego interfejsu i zapisz je do pliku `przechwycone.pcap` (w trakcie przechwytywania odwiedź stronę WWW):

```
tshark -i <nazwa_interfejsu> -w przechwycone.pcap
```

Po wykonaniu zadania przerwij działanie TSharka (`Ctrl + C`)

5. Sprawdź zawartość pliku komendą `nano`.



6. Przechwyć pakiety z użyciem filtra przechwytyującego ruch tylko na porcie 9999 (źródłowy lub docelowy):

```
tshark -i <nazwa_interfejsu> -f "port 9999"
```

Wejdź na stronę internetową – żadne pakiety nie powinny zostać przechwycone

7. Ustaw filtr dla portu 80 i powtórz test.

8. Użyj filtra dla portu 21 (FTP):

9. Uruchom drugą instancję terminala i wykonaj komendę:

```
ftp ftp.helion.pl
```

 10. Wróć do poprzedniego terminala i sprawdź, czy przechwycono pakiety FTP. Jeśli tak – zatrzymaj przechwytywanie (Ctrl + C).

11. Odczytaj z pliku `przechwycone.pcap` jedynie pakiety zawierające metodę GET z nagłówka http:

```
tshark -r przechwycone.pcap -Y "http.request.method == GET"
```

```
(kali@kali)-[~]  
$ tshark -r przechwycone.pcap -Y "http.request.method == GET"  
113 0.667847790 192.168.217.133 → 34.107.221.82 HTTP 364 GET /success.txt?ipv4 HTTP/1.1  
382 6.438528685 192.168.217.133 → 34.107.221.82 HTTP 364 GET /success.txt?ipv4 HTTP/1.1
```

Jeśli w pliku nie ma takich pakietów, wróć do punktu 4 i wygeneruj nowy ruch HTTP.

12. Przechwyć cały ruch, ale wyświetlaj tylko pakiety TCP na porcie 443:

```
tshark -i <nazwa_interfejsu> -Y "tcp.port == 443"
```

 13. Przechwyć pierwsze 5 pakietów (odwiedź stronę WWW):

```
tshark -i <nazwa_interfejsu> -c 5
```

14. Przechwyć ruch na interfejsie i zapisz wybrane pola do pliku CSV:

```
tshark -i <nazwa_interfejsu> -T fields -e ip.src -e ip.dst -E header=y -E separator=,  
-E quote=d > output.csv
```

15. Wejdź na dowolną stronę WWW i zatrzymaj działanie TSharka (Ctrl + C).

 16. Wyświetl zawartość utworzonego pliku:

```
cat output.csv
```

II. Tcpdump

Tcpdump to jedno z najbardziej znanych i najczęściej wykorzystywanych narzędzi służących do przechwytywania oraz analizy ruchu sieciowego w systemach unixowych. Jako narzędzie działające w wierszu poleceń pozwala na przechwytywanie pakietów przesyłanych w sieci w czasie rzeczywistym oraz prezentowanie ich w formie tekstowej.

1. Przetestuj działanie Tcpcap:

```
sudo tcpdump
```

2. Przechwyć wszystkie pakiety z wybranego interfejsu i zapisz je do pliku `przechwycone_tcpdump.pcap`:

```
sudo tcpdump -i <nazwa_interfejsu> -w przechwycone_tcpdump.pcap
```



3. Wejdź na dowolną stronę WWW, a następnie sprawdź zawartość utworzonego pliku.

4. Przechwyć pakiety, których adres źródłowy lub docelowy odpowiada adresowi IP Twojego sąsiada:

```
sudo tcpdump -i <nazwa_interfejsu> src <adres_ip_sasiada> or dst <adres_ip_sasiada>
```



5. Wykonaj polecenie `ping` do sąsiada i sprawdź, czy ruch ICMP został przechwycony.

6. Przechwyć pakiety komunikacji TCP na porcie 80:

```
sudo tcpdump -i <nazwa_interfejsu> 'tcp port 80'
```

7. Przechwyć pakiety TCP na porcie 80 oraz wyświetlaj ich zawartość w formacie ASCII:

```
sudo tcpdump -i <nazwa_interfejsu> -A 'tcp port 80'
```



8. Przechwyć 10 pakietów z interfejsu:

```
sudo tcpdump -i <nazwa_interfejsu> -c 10
```