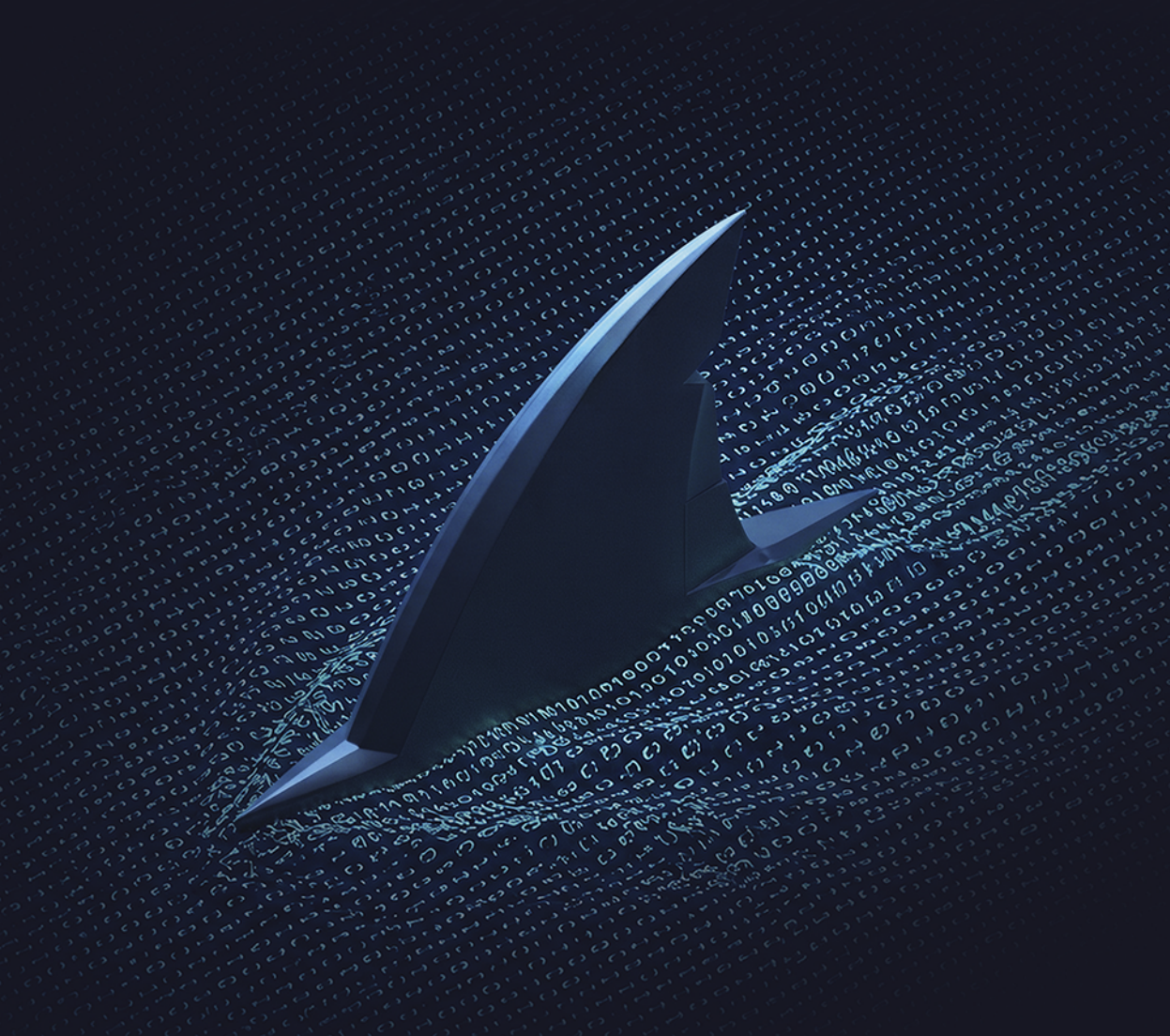


Adam Józefiok

WIRESHARK

Analiza ruchu sieciowego i wykrywanie włamań

Helion

Redaktor prowadzący: Małgorzata Kulik

Projekt okładki: Studio Gravite/Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

Grafika na okładce została wykorzystana za zgodą AdobeStock.com.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
helion.pl/user/opinie/wirana
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-289-2639-4

Printed in Poland.

Spis treści

ROZDZIAŁ 4.

Analiza włamań i bezpieczeństwo w warstwie drugiej OSI

Skanowanie sieci

Zwykle każde włamanie do sieci komputerowej zaczyna się od rozpoznania terenu. W profesjonalnym języku nazywa się to rekonesansem sieci. Rekonesans jest pierwszym krokiem do włamania. Istnieje ktoś (bo za maszyną zawsze kryje się człowiek, przynajmniej na razie), kto nie będąc częścią Twojej sieci, nagle zaczyna próbować ją zrozumieć. Przeprowadza rekonesans.

Jeśli jest intruz, któremu udało się podłączyć do sieci, to w zasadzie ten czuje się poniekąd zdezorientowany. Dzieje się tak, ponieważ sieć (szczególnie w większych przedsiębiorstwach) jest bardzo dynamiczna. Mamy w niej wiele hostów, które potrafią pojawiać się i znikać nagle, tysiące zapytań różnego rodzaju i tyle samo odpowiedzi. Zmieniające się usługi i trudny do przewidzenia pozostały ruch nic potencjalnemu włamywaczowi nie mówią. Więc żeby mógł wykonać kolejny krok, musi zdobyć chociaż szczątkowe informacje o urządzeniach.

I tu dochodzimy do sedna, czyli technik skanowania i ich wykrywania podczas rutynowych audytów czy monitoringu sieci. Technik jest wiele i niektóre działają typowo w warstwie drugiej, inne badają porty, np. TCP czy UDP, jeszcze inne używają ICMP. Oczywiście wszystkie one mają wspólny mianownik, jakim jest poznanie sieci. Prawda jest taka, że każdy rekonesans zostawia pewne ślady, trzeba tylko być czujnym i umieć je rozpoznawać. W tym rozdziale postaram się nieco ten temat przybliżyć.

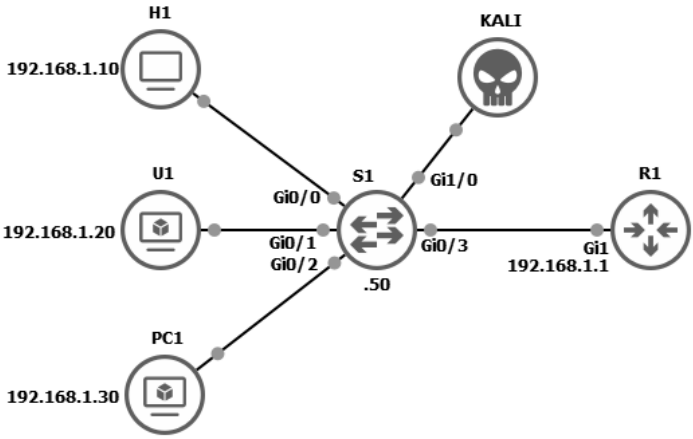
Pewne znaki w sieci mogą zwiastować próby jej skanowania, a nawet włamania. Strumienie ARP, bardzo wiele komunikatów SYN czy odpowiedzi ICMP mogą wzbudzić Twoje podejrzenia. Dlatego w dalszej części przeprowadzimy kilka analiz, ale na rzeczywistych przykładach. Będzie do tego potrzebny GNS3 (lub rzeczywisty sprzęt) oraz, co oczywiste, program Wireshark. Dzięki temu zobaczymy, jak z perspektywy analityka wygląda początek włamania i jak wcześnie można je dostrzec, zanim zmieni się w poważny incydent.

Przeprowadzanie skanowania sieci za pomocą nmap

No to zaczynamy zabawę. Jak już wspomniałem, nie chcę tylko pisać, co się pojawi, wklejać obrazki i tyle. Idea tej książki jest taka, że robię wszystko, abyś mógł powtórzyć testy u siebie w laboratorium.

W pierwszej kolejności budujemy niewielką sieć. Spójrz na rysunek 4.1. W sieci znajdują się trzy stacje robocze (Windows, Ubuntu i VPCS), przełącznik oraz router. Oczywiście jest też w sieci ten zły, czyli Kali Linux.

RYSUNEK 4.1. Niewielka sieć ze stacjami roboczymi, przełącznikiem i routerem



Zaczniemy od pierwszego rozpoznania wstępnego, czyli od narzędzia `arp-scan`. To proste narzędzie wysyła zapytania ARP na całą przestrzeń adresową. Można powiedzieć, że jest to technika bardzo prymitywna i będzie w tym trochę racji. Ale jest bardzo skuteczna. Dlaczego? Generalnie warstwa druga w sieciach z IPv4 działa bez żadnego mechanizmu, który ukrywałby adres MAC. Tak więc jeśli jakaś stacja zapyta, to cały segment usłyszy to pytanie, a najzwyczajniej w świecie właściciel tego adresu po prostu odpowie. Jak się za chwilę przekonasz, te zapytania w sieci widać, ale jeżeli w sieci jest naprawdę bardzo dużo ruchu, to mogą one po prostu „wpełznąć" niezauważone.

Przechodzimy do prostej symulacji, czegoś w rodzaju rozpoznania sieci. Na rysunku 4.2 znajduje się cały przeprowadzony skan sieci i otrzymany wynik. Użyto komendy `sudo arp-scan - - localnet`. Jeśli atakowana sieć jest znana, to możesz zamiast parametru `- - localnet` podać bezpośrednio adres sieci lub prefiks.

```
┌──(kali㉿kali)-[~]
└─$ sudo arp-scan --localnet
Interface: eth0, type: EN10MB, MAC: 00:0c:29:8a:c8:d1, IPv4: 192.168.1.100
WARNING: Cannot open MAC/Vendor file ieee-oui.txt: Permission denied
WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Permission denied
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.1.1     0c:7a:f1:08:00:00       (Unknown)
192.168.1.1     0c:7a:f1:08:00:00       (Unknown) (DUP: 2)
192.168.1.10    12:34:12:34:12:34       (Unknown: locally administered)
192.168.1.20    00:0c:29:96:82:48       (Unknown)
192.168.1.30    00:50:79:66:68:00       (Unknown)
192.168.1.30    00:50:79:66:68:00       (Unknown) (DUP: 2)
192.168.1.50    0c:49:37:a3:80:01       (Unknown)
192.168.1.50    0c:49:37:a3:80:01       (Unknown) (DUP: 2)
192.168.1.50    0c:49:37:a3:80:01       (Unknown) (DUP: 3)

9 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 1.855 seconds (138.01 hosts/sec). 5 responded

┌──(kali㉿kali)-[~]
└─$
```

RYSUNEK 4.2. Wynik przeprowadzonego skanowania sieci

Rozpoczęcie skanowania od razu wywołało lawinę zapytań ARP. Na rysunku 4.3 widać wiele zapytań ARP, ale co ciekawe, nie mija nawet sekunda od uruchomienia skanowania. Zauważ, że każde takie zapytanie trafia pod adres rozgłoszeniowy `ff:ff:ff:ff:ff:ff`. Tak więc przełącznik przekazał zapytania na swoje wszystkie aktywne interfejsy należące do tego samego VLAN-u. Intruz tym zapytaniem poniekąd zmusił całą sieć do odsłonięcia części struktury. Odpowiedziały wszystkie aktywne stacje.

No.	Time	DELTA	Source	Destination	Protocol	Length	Info
28	12.899797	0.881223	0c:49:37:a3:00:00	01:80:c2:00:00:00	STP	60	Conf. Root = 32768/1/0c:49:37:a3:00:00 Cost = 0 Port = 0x8001
29	13.020261	0.120464	192.168.154.129	192.168.154.1	TCP	141	80 → 63390 [PSH, ACK] Seq=1132 Ack=1 Win=501 Len=87
30	14.021897	1.001636	192.168.154.129	192.168.154.1	TCP	141	80 → 63390 [PSH, ACK] Seq=1219 Ack=1 Win=501 Len=87
31	14.907454	0.885557	0c:49:37:a3:00:00	01:80:c2:00:00:00	STP	60	Conf. Root = 32768/1/0c:49:37:a3:00:00 Cost = 0 Port = 0x8001
32	15.024040	0.116586	192.168.154.129	192.168.154.1	TCP	141	80 → 63390 [PSH, ACK] Seq=1306 Ack=1 Win=501 Len=87
33	15.873399	0.849359	00:0c:29:8a:c8:d1	ff:ff:ff:ff:ff:ff	ARP	60	Who has 192.168.1.0? Tell 192.168.1.100
34	15.875629	0.002230	00:0c:29:8a:c8:d1	ff:ff:ff:ff:ff:ff	ARP	60	Who has 192.168.1.1? Tell 192.168.1.100
35	15.878218	0.002589	00:0c:29:8a:c8:d1	ff:ff:ff:ff:ff:ff	ARP	60	Who has 192.168.1.2? Tell 192.168.1.100
36	15.878964	0.000746	0c:7a:f1:08:00:00	00:0c:29:8a:c8:d1	ARP	60	192.168.1.1 is at 0c:7a:f1:08:00:00
37	15.879384	0.000420	00:0c:29:8a:c8:d1	ff:ff:ff:ff:ff:ff	ARP	60	Who has 192.168.1.3? Tell 192.168.1.100
38	15.880480	0.001096	00:0c:29:8a:c8:d1	ff:ff:ff:ff:ff:ff	ARP	60	Who has 192.168.1.4? Tell 192.168.1.100
39	15.882109	0.001629	0c:7a:f1:08:00:00	00:0c:29:8a:c8:d1	ARP	60	192.168.1.1 is at 0c:7a:f1:08:00:00
40	15.882560	0.000451	00:0c:29:8a:c8:d1	ff:ff:ff:ff:ff:ff	ARP	60	Who has 192.168.1.5? Tell 192.168.1.100
41	15.885616	0.003056	00:0c:29:8a:c8:d1	ff:ff:ff:ff:ff:ff	ARP	60	Who has 192.168.1.6? Tell 192.168.1.100
42	15.888356	0.002740	00:0c:29:8a:c8:d1	ff:ff:ff:ff:ff:ff	ARP	60	Who has 192.168.1.7? Tell 192.168.1.100
43	15.890424	0.002068	00:0c:29:8a:c8:d1	ff:ff:ff:ff:ff:ff	ARP	60	Who has 192.168.1.8? Tell 192.168.1.100
44	15.891042	0.000618	00:0c:29:8a:c8:d1	ff:ff:ff:ff:ff:ff	ARP	60	Who has 192.168.1.9? Tell 192.168.1.100
45	15.892166	0.001124	00:0c:29:8a:c8:d1	ff:ff:ff:ff:ff:ff	ARP	60	Who has 192.168.1.10? Tell 192.168.1.100
46	15.892206	0.000040	12:34:12:34:12:34	00:0c:29:8a:c8:d1	ARP	42	192.168.1.10 is at 12:34:12:34:12:34
47	15.894666	0.002460	00:0c:29:8a:c8:d1	ff:ff:ff:ff:ff:ff	ARP	60	Who has 192.168.1.11? Tell 192.168.1.100
48	15.897880	0.003214	00:0c:29:8a:c8:d1	ff:ff:ff:ff:ff:ff	ARP	60	Who has 192.168.1.12? Tell 192.168.1.100

RYSUNEK 4.3. Przechwycone ramki ARP

Na rysunku widać zapisy:

```
192.168.1.1 is at 0c:7a:f1:08:00:00
192.168.1.10 is at 12:34:12:34:12:34
```

Czyli: wystąpił w sumie prosty atak i intruz ma już listę aktywnych urządzeń i ich adresy IP oraz adresy MAC. Dzięki temu wie, jaki to sprzęt i producent. To cenna wiedza jak na początek.

Na co warto zwrócić uwagę podczas analizy? Dla tego typu rekonesansu charakterystyczne jest to, że ten rodzaj skanowania tworzy bardzo wyraźny wzorzec czasowy. Zatem od razu zobaczysz serię ramek z tym samym adresem MAC źródłowym w bardzo krótkich odstępach czasu. Mało tego; zwykle będzie to *ARP Request* skierowany pod adres MAC rozgłoszeniowy. Dla analityka charakterystyczna jest seria ramek *ARP Request* wysyłanych z jednego adresu MAC w bardzo krótkich odstępach czasu, zwykle pod adres rozgłoszeniowy `ff:ff:ff:ff:ff:ff`. Taki wzorzec często wskazuje na rekonesans lub automatyczne sondowanie sieci, choć sam w sobie nie stanowi jeszcze niepodważalnego dowodu ataku. Ostateczna ocena powinna uwzględniać kontekst sieci, profil normalnego ruchu oraz zachowanie pozostałych hostów.

Wiedza ta jest kluczowa w praktyce śledczej. Jeśli w logach przełącznika lub przechwyconym ruchu sieciowym występuje rytmiczna sekwencja protokołu ARP w ciągu kilku sekund, to można z dużym prawdopodobieństwem przyjąć, że w tym czasie trwało aktywne rozpoznanie lub automatyczne sondowanie sieci.

Skan TCP SYN

Intruz w poprzednim kroku poznał, z jaką siecią ma do czynienia. Zna już adresy IP, ale jeszcze nie wie, jaka usługa jest uruchomiona na jakich urządzeniach. Teraz musi więc sprawdzić, jakie usługi są otwarte, aby wiedzieć, które można zaatakować i na jakich urządzeniach.

Chciałbym na tym etapie wspomnieć, że użyte komendy celowo skanują całą sieć, ale musisz mieć świadomość, że intruz może wykonać skan tylko jednego konkretnego urządzenia oraz usługi. Wtedy nie zobaczysz aż tylu wiadomości. Natomiast chodzi tutaj o pokazanie Ci pewnego wzorca działania i rodzaju komunikatów, jakie się pojawiają w takim przypadku.

Najbardziej typowym i najczęściej używanym skanem w analizie sieci jest *SYN scan*. *SYN scan* uruchamiany opcją `-sS` jest szybki i relatywnie mało inwazyjny, ponieważ nie dochodzi w nim do pełnego ustanowienia połączenia TCP. Z tego powodu część usług aplikacyjnych może nie odnotować takiego kontaktu jako pełnego połączenia, jednak ruch ten nadal może być rejestrowany przez zapory, systemy IDS/IPS, mechanizmy audytu systemowego lub inne narzędzia monitorujące.

Spójrz na rysunek 4.4, na którym widać użycie polecenia `sudo nmap -sS 192.168.1.0/24` oraz fragment wyniku. Wydane polecenie oznacza, że dla każdego hosta w podsieci zostaną wysłane pakiety `SYN` na wybrane porty TCP, `nmap` sprawdzi wynik i wyświetli odpowiednie informacje.

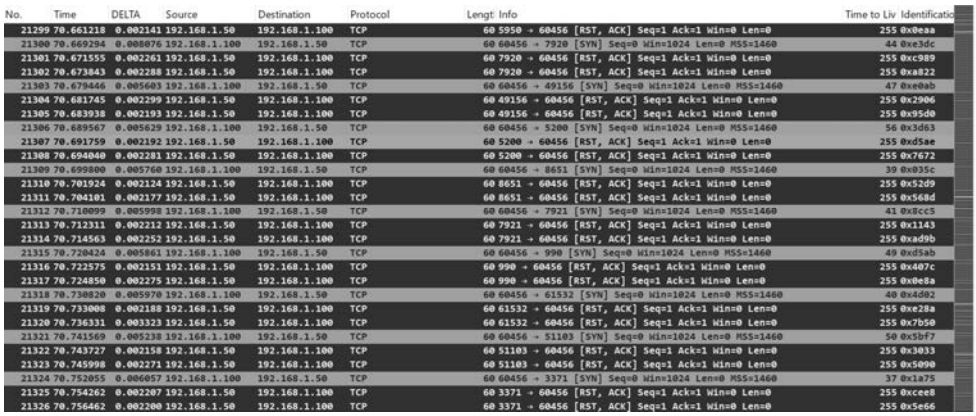
RYSUNEK 4.4. Wynik skanowania całej sieci

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sS 192.168.1.0/24
sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-11 14:56 EST
Nmap scan report for 192.168.1.1
Host is up (0.18s latency).
Not shown: 996 closed tcp ports (reset)
PORT    STATE SERVICE
22/tcp  open  ssh
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https
MAC Address: 0C:7A:F1:08:00:00 (Unknown)

Nmap scan report for 192.168.1.10
Host is up (0.00017s latency).
Not shown: 992 closed tcp ports (reset)
PORT     STATE SERVICE
80/tcp   open  http
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3300/tcp open  ceph
3389/tcp open  ms-wbt-server
5000/tcp open  upnp
8088/tcp open  radan-http
MAC Address: 12:34:12:34:12:34 (Unknown)

Nmap scan report for 192.168.1.20
Host is up (0.00010s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE
```

A teraz spójrz na rysunek 4.5, na którym znajdziesz prawdziwe zamieszanie. Tylko na pierwszy rzut oka wydaje się to nie mieć sensu. Kiedy sam przeprowadzisz taki atak, zauważysz, że w Wiresharku pojawia się bardzo charakterystyczny wzorzec ramek. Schemat dla portu zamkniętego wygląda zwykle następująco: skaner wysyła `SYN`, a host odpowiada `RST/ACK`. Dla portu otwartego odpowiedź hosta ma postać `SYN/ACK`, po czym skaner zwykle odsyła `RST`, aby nie kończyć pełnego połączenia. To właśnie powtarzalność tych sekwencji pozwala analitykowi rozpoznać półotwarty skan `SYN`.



No.	Time	DELTA	Source	Destination	Protocol	Lengt	Info	Time to Liv	Identificatio
21299	70.661218	0.002141	192.168.1.50	192.168.1.100	TCP	60	5950 → 60456 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	255	0x0eaa
21300	70.669294	0.008076	192.168.1.100	192.168.1.50	TCP	60	60456 → 7920 [SYN] Seq=0 Win=1024 Len=0 MSS=1460	44	0xe3dc
21301	70.671555	0.002261	192.168.1.50	192.168.1.100	TCP	60	7920 → 60456 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	255	0xc989
21302	70.673843	0.002288	192.168.1.50	192.168.1.100	TCP	60	7920 → 60456 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	255	0xa822
21303	70.679446	0.005603	192.168.1.100	192.168.1.50	TCP	60	60456 → 49156 [SYN] Seq=0 Win=1024 Len=0 MSS=1460	47	0xe0ab
21304	70.681745	0.002299	192.168.1.50	192.168.1.100	TCP	60	49156 → 60456 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	255	0x2906
21305	70.683938	0.002193	192.168.1.50	192.168.1.100	TCP	60	49156 → 60456 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	255	0x95d0
21306	70.689567	0.005629	192.168.1.100	192.168.1.50	TCP	60	60456 → 5200 [SYN] Seq=0 Win=1024 Len=0 MSS=1460	56	0x3d63
21307	70.691759	0.002192	192.168.1.50	192.168.1.100	TCP	60	5200 → 60456 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	255	0xd5ae
21308	70.694040	0.002281	192.168.1.50	192.168.1.100	TCP	60	5200 → 60456 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	255	0x7672
21309	70.699800	0.005760	192.168.1.100	192.168.1.50	TCP	60	60456 → 8651 [SYN] Seq=0 Win=1024 Len=0 MSS=1460	39	0x035c
21310	70.701924	0.002124	192.168.1.50	192.168.1.100	TCP	60	8651 → 60456 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	255	0x52d9
21311	70.704101	0.002177	192.168.1.50	192.168.1.100	TCP	60	8651 → 60456 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	255	0x568d
21312	70.710099	0.005998	192.168.1.100	192.168.1.50	TCP	60	60456 → 7921 [SYN] Seq=0 Win=1024 Len=0 MSS=1460	41	0x8cc5
21313	70.712311	0.002212	192.168.1.50	192.168.1.100	TCP	60	7921 → 60456 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	255	0x1143
21314	70.714563	0.002252	192.168.1.50	192.168.1.100	TCP	60	7921 → 60456 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	255	0xad9b
21315	70.720424	0.005861	192.168.1.100	192.168.1.50	TCP	60	60456 → 990 [SYN] Seq=0 Win=1024 Len=0 MSS=1460	40	0xd5ab
21316	70.722575	0.002151	192.168.1.50	192.168.1.100	TCP	60	990 → 60456 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	255	0x407c
21317	70.724850	0.002275	192.168.1.50	192.168.1.100	TCP	60	990 → 60456 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	255	0x0e8a
21318	70.730820	0.005970	192.168.1.100	192.168.1.50	TCP	60	60456 → 61532 [SYN] Seq=0 Win=1024 Len=0 MSS=1460	40	0x4d02
21319	70.733008	0.002188	192.168.1.50	192.168.1.100	TCP	60	61532 → 60456 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	255	0xe28a
21320	70.736331	0.003323	192.168.1.50	192.168.1.100	TCP	60	61532 → 60456 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	255	0x7b50
21321	70.741569	0.005238	192.168.1.100	192.168.1.50	TCP	60	60456 → 51103 [SYN] Seq=0 Win=1024 Len=0 MSS=1460	50	0x5bf7
21322	70.743727	0.002158	192.168.1.50	192.168.1.100	TCP	60	51103 → 60456 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	255	0x3033
21323	70.745998	0.002271	192.168.1.50	192.168.1.100	TCP	60	51103 → 60456 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	255	0x5090
21324	70.752055	0.006057	192.168.1.100	192.168.1.50	TCP	60	60456 → 3371 [SYN] Seq=0 Win=1024 Len=0 MSS=1460	37	0x1a75
21325	70.754262	0.002207	192.168.1.50	192.168.1.100	TCP	60	3371 → 60456 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	255	0xcee8
21326	70.756462	0.002200	192.168.1.50	192.168.1.100	TCP	60	3371 → 60456 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	255	0x5e66

RYSUNEK 4.5. Przechwycony ruch TCP

Rysunek obrazuje ten schemat:

```
192.168.1.50 → 192.168.1.100   [SYN]
192.168.1.100 → 192.168.1.50   [RST, ACK]
192.168.1.50 → 192.168.1.100   [SYN]
192.168.1.100 → 192.168.1.50   [RST, ACK]
```

Zauważ, że wynik jest niemalże geometryczny, powtarzalny i trudny do pomylenia z naturalnym ruchem sieci.

Dla analityka *SYN scan* to jeden z najbardziej jednoznacznych sygnałów, że w sieci dzieje się coś niepokojącego. Układ opcji TCP, wartości okna i charakter sekwencji pakietów mogą być pomocne w rozpoznaniu narzędzia, jednak nie stanowią jednoznacznego dowodu, ponieważ zależą od wersji `nmap`, systemu operacyjnego, typu skanu oraz urządzeń pośredniczących. Od razu rzuca się w oczy wysoka częstotliwość ramek. Zauważ, że kolejne komunikaty `SYN` wysyłane są co 1 – 3 ms. Charakterystyczny jest również bardzo powtarzalny wzorzec, przedstawiający identyczny host źródłowy, różne porty docelowe, następnie brak dokończenia handshake'u, bo host odpowiada `SYN/ACK`, ale intruz nie finalizuje handshake'u. Warto również zwrócić uwagę, jeśli w sieci pojawia się ruch do portów, które normalnie nigdy nie są używane.

W wielu scenariuszach `nmap` wykazuje powtarzalny wzorzec kolejności odpytywania portów, ale nie należy zakładać, że zawsze skanuje je liniowo ani że sama kolejność pozwala pewnie odtworzyć intencję atakującego. Pomaga to jednak przewidzieć jego kolejny krok. Tego typu cechy bywają pomocne w rozpoznaniu narzędzia lub systemu źródłowego, ale nie należy traktować ich jako jednoznacznego dowodu, gdyż podobny układ może zależeć również od stosu TCP/IP, wersji narzędzia i urządzeń pośredniczących.

Skan UDP

Skanowanie UDP jest nieco inne. Dzieje się tak dlatego, że protokół UDP nie daje atakującemu wygodnych sygnałów, tak jak to było w TCP. W TCP pary `SYN–RST` lub `SYN–SYN/ACK` formują czytelny język stanu portu. W UDP reakcje są bardziej rozproszone, a większość sensu bierze się nie z samego datagramu, ale z tego, co system operacyjny robi w odpowiedzi na jego brak.

Tym razem do przeprowadzenia skanu UDP użyto w KALI polecenia `nmap -sU 192.168.1.0/24`. Spójrz na rysunek 4.6. Widać na nim wydane polecenie oraz fragment wyniku. Zatrzymajmy się na chwilę przy tym wyniku.

RYSUNEK 4.6. Wynik skanowania UDP

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sU 192.168.1.0/24
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-11 15:50 EST
Nmap scan report for 192.168.1.1
Host is up (0.0062s latency).
All 1000 scanned ports on 192.168.1.1 are in ignored states.
Not shown: 1000 closed udp ports (port-unreach)
MAC Address: 0C:7A:F1:08:00:00 (Unknown)

Nmap scan report for 192.168.1.10
Host is up (0.00026s latency).
Not shown: 989 closed udp ports (port-unreach)
PORT     STATE         SERVICE
137/udp  open          netbios-ns
138/udp  open|filtered netbios-dgm
162/udp  open|filtered snmptrap
500/udp  open|filtered isakmp
514/udp  open|filtered syslog
1900/udp open|filtered upnp
3389/udp open|filtered ms-wbt-server
4500/udp open|filtered nat-t-ike
5050/udp open|filtered mmcc
5353/udp open|filtered zeroconf
5355/udp open|filtered llmnr
MAC Address: 12:34:12:34:12:34 (Unknown)

Nmap scan report for 192.168.1.20
Host is up (0.00034s latency).
Not shown: 998 closed udp ports (port-unreach)
PORT     STATE         SERVICE
631/udp  open|filtered ipp
5353/udp open|filtered zeroconf
MAC Address: 00:0C:29:96:82:48 (VMware)

Nmap scan report for 192.168.1.30
Host is up (0.069s latency).
```

Zwróć uwagę na to, że wynik skanowania UDP ma zupełnie inny charakter niż skan TCP. W raporcie od razu rzuca się w oczy, że router `192.168.1.1` nie zwrócił żadnych widocznych otwartych portów UDP. Zatem otrzymujesz informację, że wszystkie porty znajdują się w stanie `ignored states`. Jeżeli skanowany port UDP jest zamknięty, system może odesłać *ICMP Destination unreachable/Port unreachable*, co pozwala `nmap` oznaczyć port jako `closed`. W praktyce takie odpowiedzi mogą jednak podlegać filtrowaniu lub ograniczaniu szybkości, zwłaszcza na urządzeniach sieciowych, dlatego brak odpowiedzi nie musi oznaczać portu otwartego i często skutkuje stanem `open/filtered`. Dla atakującego oznacza to, że router `R1` nie jest atrakcyjnym celem na tym etapie. Dla analityka natomiast jest to sygnał, że router zachowywał się przewidywalnie, bo odpowiedzi ICMP spełniają swoje zadanie.

Dalej na rysunku znajdują się wyniki dla hostów. Dosyć szczegółowo o otwartych portach informuje `192.168.1.10`. Widać tam cały szereg portów oznaczonych jako `open/filtered`. Można powiedzieć, że jest to stan, który wprowadza pewien rodzaj niejednoznaczności. Dlaczego tak jest? W zasadzie `nmap` nie wie, czy port faktycznie jest otwarty, czy ruch został odfiltrowany. Dzieje się tak dlatego, że w UDP brakuje odpowiednika tego, co świetnie działa w TCP, czyli po prostu odpowiedzi.

Jak już wiesz, protokół UDP ma bardzo prostą naturę. Urządzenie wysyła pakiet i nie dostaje potwierdzenia, czy dotarł, czy nie. Milczenie nie ma w UDP żadnego specjalnego znaczenia. Ale w skanowaniu `nmap` to milczenie jest źródłem całego problemu. Kiedy program `nmap` wysyła pakiet UDP na jakiś port, mogą wydarzyć się trzy rzeczy. Pierwsza możliwość: host odpowiada *ICMP Port unreachable*. W takim wypadku `nmap` wie, że port jest zamknięty. Druga możliwość: host odpowie jakąś prawdziwą odpowiedzią UDP. W takim wypadku `nmap` wie, że port jest otwarty. I trzecia możliwość: host nie odpowie. Tutaj właśnie pojawia się problem. To milczenie można interpretować na dwa sposoby. Pierwsza interpretacja jest taka, że port faktycznie jest otwarty i usługa przyjmuje pakiet mimo braku ICMP. Druga interpretacja to, że firewall po drodze odrzuca pakiety bez słowa, celowo myląc intruza. `nmap` nie ma jak tego rozróżnić. Dlatego oznacza te porty jako `open/filtered`.

Po zakończonym ataku przyjrzyjmy się temu, co udało się przechwycić w trakcie jego trwania. Rysunek 4.7 przedstawia przechwycony ruch.

No.	Time	DELTA	Source	Destination	Protocol	Lengt	Info
3961	60.668481	0.001022	192.168.1.100	192.168.1.20	UDP	42	36871 → 113 Len=0
3962	60.668731	0.000250	192.168.1.100	192.168.1.20	UDP	42	36871 → 113 Len=0
3963	60.763683	0.094952	0c:49:37:a3:00…	01:80:c2:00:00…	STP	60	Conf. Root = 32768/1/0c:49:37:a3:00:00 Cost = 0 Po
3964	60.767504	0.003821	192.168.1.100	192.168.1.1	UDP	82	36871 → 40019 Len=40
3965	60.768701	0.001197	192.168.1.1	192.168.1.100	ICMP	70	Destination unreachable (Port unreachable)
3966	60.979669	0.210968	192.168.1.100	192.168.1.50	UDP	82	36873 → 40019 Len=40
3967	60.981966	0.002297	192.168.1.50	192.168.1.100	ICMP	70	Destination unreachable (Port unreachable)
3968	61.068153	0.086187	192.168.1.100	192.168.1.20	UDP	60	36873 → 113 Len=0
3969	61.068272	0.000119	192.168.1.20	192.168.1.100	ICMP	70	Destination unreachable (Port unreachable)
3970	61.069969	0.001697	192.168.1.100	192.168.1.20	UDP	42	36873 → 113 Len=0
3971	61.071095	0.001126	192.168.1.20	192.168.1.100	ICMP	70	Destination unreachable (Port unreachable)
3972	61.072777	0.001682	192.168.1.50	192.168.1.100	ICMP	70	Redirect (Redirect for host)
3973	61.074131	0.001354	192.168.1.100	192.168.1.20	UDP	60	36873 → 113 Len=0
3974	61.075206	0.001075	192.168.1.100	192.168.1.10	UDP	60	36873 → 20380 Len=0
3975	61.075324	0.000118	192.168.1.10	192.168.1.100	ICMP	70	Destination unreachable (Port unreachable)
3976	61.076194	0.000870	192.168.1.100	192.168.1.10	UDP	42	36873 → 20380 Len=0
3977	61.076479	0.000285	192.168.1.10	192.168.1.100	ICMP	70	Destination unreachable (Port unreachable)
3978	61.077991	0.001512	192.168.1.10	192.168.1.100	ICMP	70	Destination unreachable (Port unreachable)
3979	61.090089	0.012098	192.168.154.129	192.168.154.1	TCP	141	80 → 49777 [PSH, ACK] Seq=5312 Ack=1 Win=501 Len=87
3980	61.380109	0.290020	192.168.1.100	192.168.1.50	UDP	82	36871 → 34038 Len=40
3981	61.568187	0.188078	192.168.1.100	192.168.1.1	UDP	82	36871 → 34038 Len=40
3982	61.570421	0.002234	192.168.1.1	192.168.1.100	ICMP	70	Destination unreachable (Port unreachable)
3983	61.781161	0.210740	192.168.1.100	192.168.1.50	UDP	82	36873 → 34038 Len=40
3984	61.783170	0.002009	192.168.1.50	192.168.1.100	ICMP	70	Destination unreachable (Port unreachable)
3985	61.868250	0.085080	192.168.1.100	192.168.1.20	UDP	82	36871 → 50612 Len=40
3986	61.868930	0.000680	192.168.1.100	192.168.1.20	UDP	82	36871 → 50612 Len=40
3987	61.869196	0.000266	192.168.1.100	192.168.1.20	UDP	82	36871 → 50612 Len=40

RYSUNEK 4.7. Pakiety UDP przechwycone w trakcie skanowania

Zauważ, że `nmap` wysyła krótkie datagramy UDP do kolejnych portów i czeka na reakcję. Te datagramy są oszczędne, pozbawione treści, bo ich celem nie jest nawiązanie połączenia, lecz wywołanie odpowiedzi systemu operacyjnego po drugiej stronie. W Wiresharku ruch ten od razu wyróżnia się na tle wszystkiego, co znamy z normalnego użytkowania sieci. Jest to bardzo intensywne i równomierne odpytywanie każdego portu. W krótkich, milisekundowych odstępach pojawiają się datagramy UDP wysyłane do hostów w sieci: `192.168.1.10`, `1.20`, `1.30`, a także do routera `192.168.1.1`. Jest to bardzo podejrzany ruch. Dalej mamy odpowiedzi *ICMP Destination unreachable*, które poszczególne systemy wysyłają, gdy dany port UDP jest zamknięty.

Dla analityka odpowiedzi ICMP mogą stanowić cenną wskazówkę interpretacyjną, ponieważ ich typ, kod, częstotliwość i sposób generowania pozwalają wnioskować o zachowaniu hosta lub urządzenia pośredniczącego. Nie należy jednak traktować tych cech jako samodzielnego, jednoznacznego identyfikatora systemu operacyjnego lub roli urządzenia.

Zauważ, że router odpowiada inaczej niż hosty końcowe; widzimy nawet moment, w którym router udziela komunikatu `Redirect`, pokazując, że intruz chwilowo próbował wysłać pakiet przez bramę, mimo że cel znajdował się w tej samej sieci.

To, co najważniejsze dla analityka, to skala i intensywność. Normalna sieć w zasadzie nie generuje strumieni *ICMP Port unreachable* w takich ilościach. Mogą się zdarzyć pojedyncze komunikaty, gdy aplikacja próbuje kontaktować się z usługą, której nie ma, ale nie zobaczymy ciągłej serii odpowiedzi z kilku hostów naraz, z równym, niemal matematycznym rytmem. Tutaj ICMP jest regularny, powtarzalny i bez dłuższych przerw. Jest to typowy znak sugerujący odpytywanie.

Problemy z protokołem ARP (ARP poisoning i MiTM)

W rozdziale 1. omówiłem protokół ARP. Wbrew pozorom protokół ten wciąż pełni niezwykle istotną funkcję w komunikacji. Szczególnie w komunikacji dotyczącej protokołu IPv4, który ciągle jest najczęściej używanym protokołem warstwy sieci. Dlatego zagrożenia, które dotyczą protokołu ARP, są nadal aktualne i mogą być niebezpieczne, jeśli na czas nie zostaną wykryte. Zobaczmy, w jaki sposób możesz wykryć anomalie związane z ARP w programie Wireshark. ARP to mechanizm mapowania adresów IP na adresy łącza (MAC), a jego podstawy opisuje RFC 826.

Atak polega na tym, że osoba niepowołana uruchamia program, który służy do oszukiwania kart sieciowych. Skutkiem działania takiego programu jest to, że stacja robocza, przesyłając zapytanie ARP, otrzymuje w odpowiedzi od atakującego adres MAC jego stacji roboczej (atakującej). Uaktualnia na tej podstawie swoją tablicę ARP. Dzieje się tak ze względu na to, że atakujący zwykle wysyła spreparowane *ARP_REPLY* (*opcode 2*) albo ogłoszenia (*gratuitous ARP*), które powodują nadpisanie wpisów ARP ofiary. Dzięki temu część ruchu ofiary, w szczególności ruch kierowany do podszywanego adresu IP, może zostać skierowana do hosta atakującego. Aby uzyskać pełny scenariusz MiTM, atakujący musi jeszcze zapewnić dalsze przekazywanie pakietów, w przeciwnym razie skutkiem może być jedynie zakłócenie komunikacji.

Pełny atak spoofingowy polega na tym, że atakujący tak przystosowuje sprzęt, że ofiara nie ma nawet świadomości, że jest podsłuchiwana. Wszystkie usługi uruchomione przez ofiarę działają tak, jak gdyby nic niepokojącego się nie działo.

W praktyce należy odróżnić *ARP Probe* od *gratuitous ARP*, czyli *ARP Announcement*. *ARP Probe* służy do sprawdzenia, czy planowany adres IPv4 nie jest już używany w sieci, natomiast *gratuitous ARP* jest rozgłoszeniowym ogłoszeniem własnego mapowania IP/MAC, stosowanym m.in. do odświeżania tablic ARP innych hostów lub do obsługi scenariuszy przełączenia awaryjnego. Samo wystąpienie *gratuitous ARP* nie jest z definicji złośliwe, lecz w analizie bezpieczeństwa należy zwracać uwagę na jego częstotliwość, kontekst i spójność z topologią sieci.

Nie każde częste lub nietypowe występowanie zapytań ARP oznacza atak. W wielu sieciach można obserwować wzmożony ruch ARP wynikający z normalnej pracy urządzeń, rekonfiguracji hostów, odświeżania wpisów cache'u, pracy środowisk wirtualnych, mechanizmów wysokiej dostępności albo procesów wykrywania konfliktu adresów. Z drugiej

strony ruch ARP, który nie pasuje do normalnego profilu sieci, powinien zwrócić uwagę analityka. Dlatego ocena nie powinna opierać się na pojedynczym pakiecie ani jednej liczbie, lecz na kontekście, częstotliwości, powtarzalności i spójności mapowań IP/MAC.

Nie istnieje uniwersalny próg liczby pakietów ARP, po przekroczeniu którego można automatycznie stwierdzić atak. Ocena powinna się opierać na profilu normalnego ruchu w danej sieci, typie urządzeń, ich funkcjach oraz obserwowanych wzorcach czasowych. Pamiętaj, że w ocenianiu takich zjawisk musisz być ostrożny. Nawet jeśli masz więcej komunikacji ARP, niż to zaznaczyłem, np. 5, a nawet 10 na sekundę, też nie musi to stanowić problemu. Zawsze warto szukać w takim przypadku podejrzanych wzorców. Sama liczba ARP w sieci zależy od tego, co faktycznie robią urządzenia w sieci. Jeśli wszystkie urządzenia łączą się tylko z jednym urządzeniem (np. z routerem, który łączy je dalej z siecią globalną), to zobaczysz niewielką liczbę ARP. Jeśli jednak urządzenia wysyłają np. okresowe wiadomości do wszystkich swoich sąsiadów, zobaczysz bardzo dużo komunikacji ARP. Pamiętaj jednak, że nie ma standardowego, uniwersalnego progu liczby pakietów ARP, który definiowałby atak. Najlepsza praktyka to zbudowanie typowego profilu ruchu ARP dla danej sieci i wykrywanie anomalii. Nie ma w żadnym RFC dokładnych bezpiecznych wartości, jednak na podstawie swoich doświadczeń podałem je powyżej, abyś miał jakieś rozeznanie.

Nie chcę tylko teoretyzować, dlatego najpierw chciałbym pokazać Ci, jak szybko przeprowadzić eksperyment, w którym możesz wywołać taki atak, a następnie już w domowych warunkach analizować jego skutki. Pamiętaj, by eksperyment przeprowadzić w sieci odseparowanej od rzeczywistej, najlepiej na maszynach wirtualnych z interfejsami ustawionymi na wirtualne `VMnet`, jeśli używasz np. VMware.

Na jednej maszynie wirtualnej uruchom system `KALI`, a na drugiej np. stację z systemem Linux lub Windows. Chodzi o to, aby połączyć te dwie maszyny poprzez interfejs wirtualny. Więcej nie będzie Ci potrzebne. Oczywiście sprawdź wcześniej, czy maszyny się ze sobą komunikują.

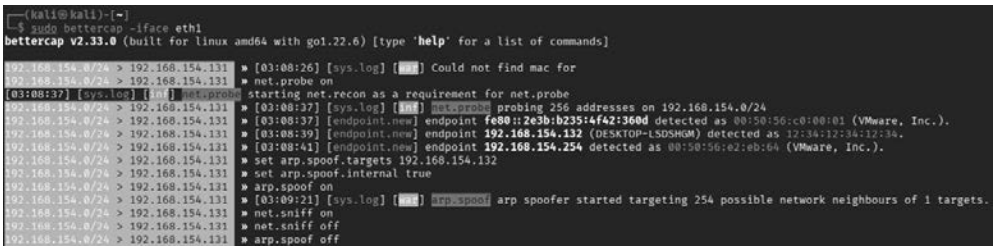
W ramach bardziej rozszerzonych ćwiczeń i do pełnej demonstracji ataku ARP spoofing/MiTM najlepiej przygotować co najmniej trzy węzły, tj. host atakujący, host ofiary oraz bramę lub trzeci host, przez który będzie przechodził ruch. Dwie maszyny wystarczają do obserwacji wybranych zjawisk ARP, ale nie oddają w pełni scenariusza klasycznego pośredniczenia w komunikacji.

Atak przeprowadzisz z maszyny `KALI`, wykorzystując program bettercap. Może się okazać, że program nie jest dostępny, dlatego wcześniej sprawdź to, wpisując w terminalu komendę `sudo bettercap -v`. Jeśli otrzymasz błąd, wówczas pobierz i zainstaluj odpowiedni pakiet, wydając najpierw komendę `sudo apt update`, a potem `sudo apt install bettercap -y`. Oczywiście w tym czasie nie zapomnij o podłączeniu maszyny wirtualnej do sieci Internet.

Po ewentualnej instalacji możesz przejść do sprawdzenia, jaki interfejs jest odpowiedzialny za komunikację z drugą maszyną. Użyj komendy `ip addr show` i zapisz symbol interfejsu. Dalej przejdź komendą `sudo bettercap -iface [`*`interfejs`*`]` do trybu wydawania dodatkowych poleceń programu bettercap.

Najpierw wydaj komendę `net.probe on`, która może wykryć aktywne hosty widoczne w danym segmencie sieci. Następnie wpisz: `set arp.spoof.targets [`*`adres IP stacji lub całej sieci`*`]`. Kolejna komenda jest odpowiedzialna za uruchomienie ataku lokalnie: `set arp.spoof.internal true`. Aby uruchomić ataki, wydaj komendy `arp.spoof on` i `net.sniff on`.

W celu ich przerwania jako parametr wpisz `off`. Rysunek 4.8 przedstawia wydanie opisanych komend.



RYSUNEK 4.8. Atak ARP spoof

Teraz, kiedy wiesz już, jak przeprowadzić atak, możesz zebrać dowolną ilość danych w programie Wireshark. W tym celu w trakcie ataku uruchom program Wireshark np. na stacji atakowanej lub na łączu wirtualnym.

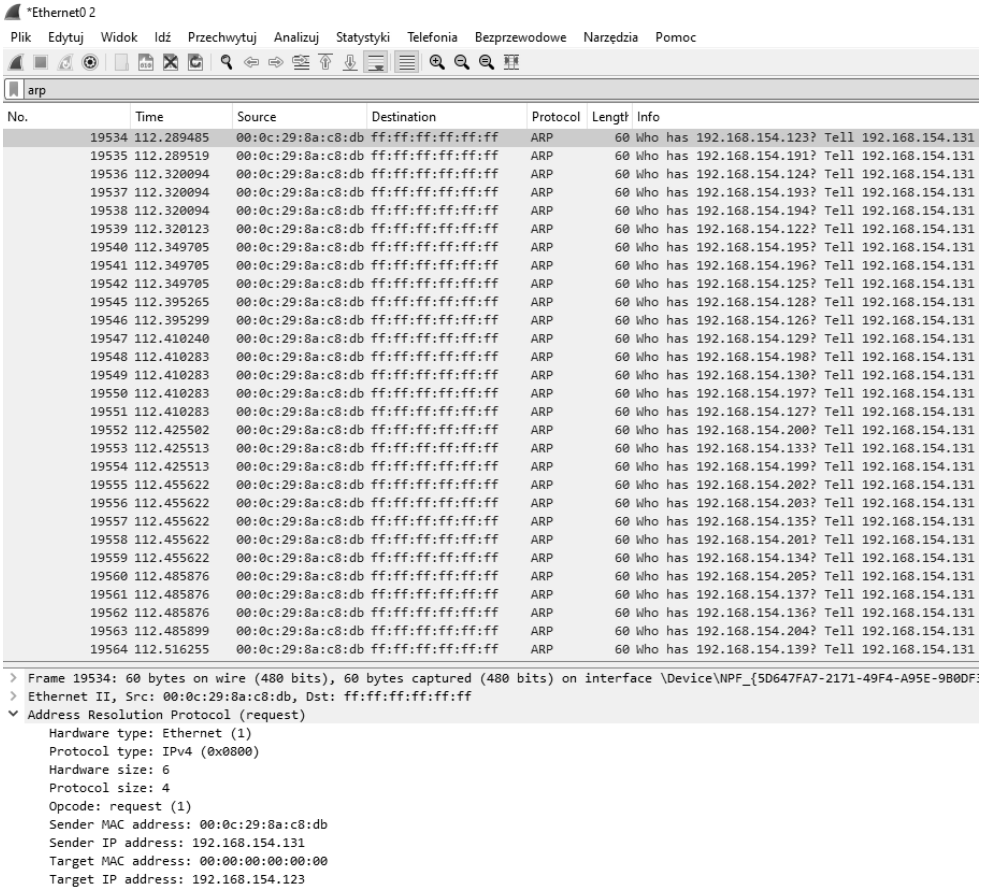
Jak rozpoznać tego typu ataki w sieci?

Duża liczba *ARP Request* może wskazywać na etap rozpoznania lub aktywne sondowanie sieci przez narzędzie atakującego, jednak nie jest jeszcze istotą ataku ARP spoofing. Właściwe zatruwanie cache'u ARP rozpoznaje się przede wszystkim po fałszywych *ARP Reply* albo powtarzających się *gratuitous ARP*, w których jeden adres MAC zostaje ogłoszony jako właściciel wielu adresów IP. Masowe *ARP Request* są zwykle objawem etapu sondowania lub działania funkcji wykrywania hostów, natomiast właściwe zatruwanie cache'u ARP realizuje się przede wszystkim przez spreparowane *ARP Reply* albo powtarzające się ogłoszenia mapowania IP/MAC.

W pierwszej kolejności użyj filtru `arp`. Na rysunku 4.9 możesz zauważyć wiele ramek ARP. Są to masowo wysyłane pakiety *ARP Request*. Świadczą o tym zapisy takie jak np. `Who has 192.168.154.xxx? Tell 192.168.154.131`. Źródłem tych pakietów jest stacja z tym samym adresem MAC. Pakiety kierowane są pod adres MAC rozgłoszeniowy `ff:ff:ff:ff:ff:ff`. Na razie w sumie nic odkrywczego nie piszę, ale jest pewien szczegół, który powinien Cię zawsze zainteresować. Tak naprawdę informacja `Tell 192.168.154.131` oznacza, że jest to źródłowy adres, do którego każda ze stacji ma odpowiedzieć. I gdyby nie to, że w zaledwie kilkanaście sekund mamy już na liście prawie 20 000 takich pakietów, można by uznać to za poprawną komunikację. Ale liczba tych zapytań jest bardzo duża i w zasadzie mamy do czynienia z zalewaniem. Tak zachowuje się typowy program do sondowania sieci.

Żeby sprawdzić, czy masz do czynienia z sondowaniem, musisz jeszcze zajrzeć do odpowiedzi (*opcode 2*), gdyż to one wpisują fałszywy MAC do tablicy ofiary. Spójrz więc na rysunek 4.10, gdzie znajdują się pakiety odfiltrowane filtrem `arp.opcode == 2`.

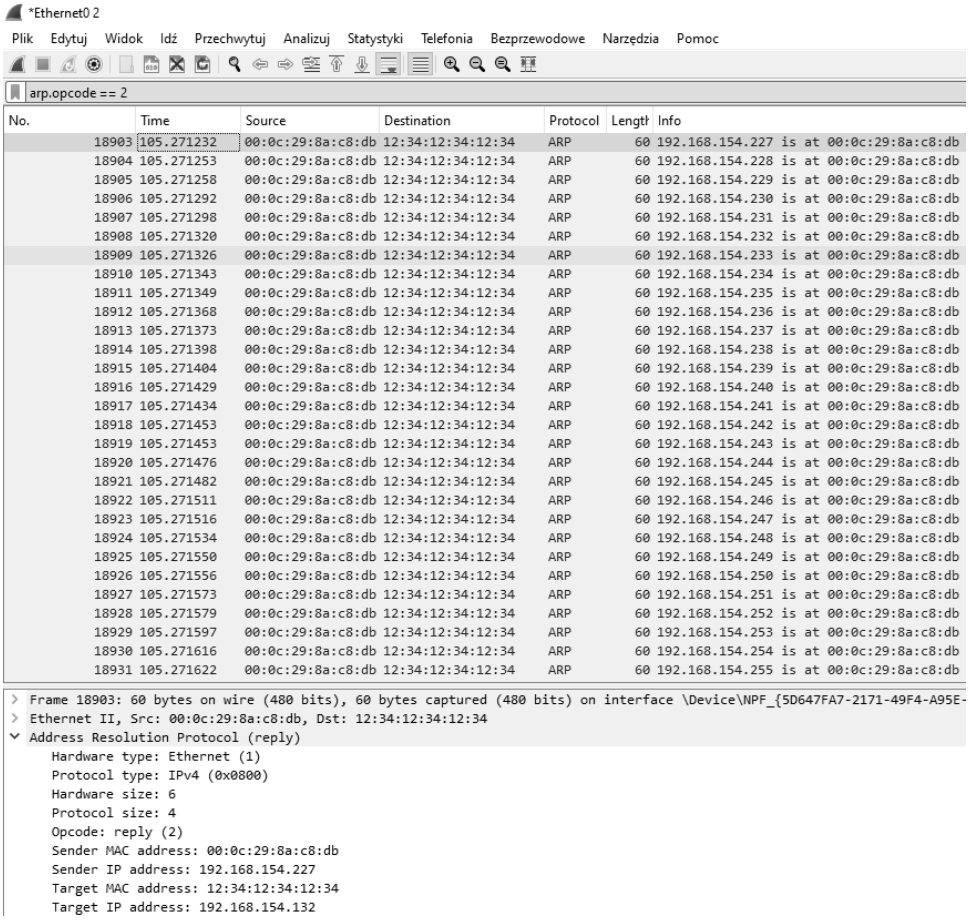
Rysunek przedstawia masowe *ARP Reply*. Powtarza się w nich, to, że stacja `KALI` z adresem MAC `00:0c:29:8a:c8:db` rozgłasza, że wiele adresów w podsieci jest przypisanych do adresu MAC stacji `KALI`. Jest to typowy efekt ataków ARP. Jeśli zastosujesz dodatkowo filtr `arp.src.proto_ipv4 == 192.168.154.131`, pojawią się pakiety, w których pole *Sender Protocol Address* ma wartość `192.168.154.131`. Pamiętaj jednak, że w ataku ARP spoofing nie musi to być rzeczywisty adres IP stacji atakującej, lecz często adres, pod który się ona podszywa, np. adres bramy.



RYSUNEK 4.9. Ramki ARP przechwycone w trakcie ataku ARP spoof

Pakiety *ICMP Redirect* nie są typowym ani wymaganym wskaźnikiem ataku ARP spoofing/MiTM. Mechanizm *Redirect* jest związany z decyzjami routingu i jest generowany przez routery w określonych sytuacjach. W analizie ARP MiTM należy koncentrować się przede wszystkim na niespójnych mapowaniach IP/MAC, masowych *ARP Reply*, częstych *gratuitous ARP* oraz na zmianach adresu MAC przypisanego do bramy lub hostów końcowych.

Możesz zastosować filtr `icmp && (ip.src == 192.168.154.132 || ip.dst == 192.168.154.132)`, gdzie podany adres IP to adres stacji atakowanej. Filtr ten może ujawnić dodatkowe komunikaty ICMP związane z zachowaniem routingu w badanym środowisku, ale ich obecność jeszcze nie potwierdza przekazywania ruchu przez host atakujący. Nie jest to jednak ostateczny dowód, bo pakiety *ICMP Redirect* to osobny mechanizm *ICMP* (*kod 5*) wysyłany przez routery.

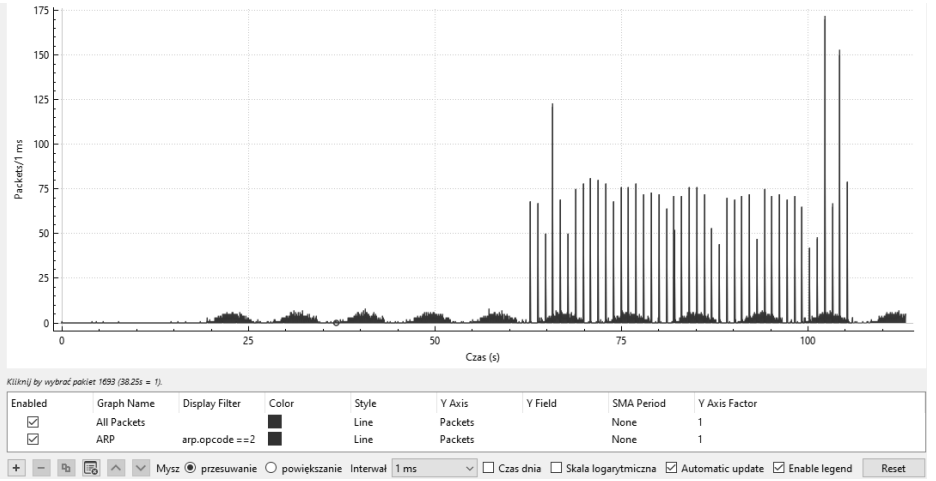


RYSUNEK 4.10. Ramki odpowiedzi ARP

Reasumując: typowy atak ARP najpierw charakteryzuje się nagłym wzrostem liczby *ARP Reply* (*opcode 2*), które są generowane przez jeden adres MAC. Dodatkowo wiele adresów IP zostaje powiązanych z tym samym adresem MAC, co jest jednym z najbardziej charakterystycznych objawów zatruwania tablic ARP.

Jeśli chcesz, możesz od razu zobaczyć to na wykresie. W programie Wireshark wybierz menu *Statystyki*, a potem *I/O graphs*. Następnie ustaw interwał na `1ms`, a w filtrze podaj `arp.opcode==2`. Rysunek 4.11 pokazuje gwałtowny skok pakietów ARP. To pokazuje jasno, że coś jest nie tak.

Do wstępnego wychwycenia skanu `SYN` można użyć filtru `tcp.flags.syn==1 && tcp.flags.ack==0`, a następnie analizować częstotliwość ramek, zakres portów docelowych, liczbę hostów docelowych oraz brak dokończenia połączeń. Jeśli w danym laboratorium występują dodatkowe cechy, takie jak określony rozmiar okna lub układ opcji TCP, można je traktować jako pomocnicze wskaźniki, a nie jako warunek konieczny.



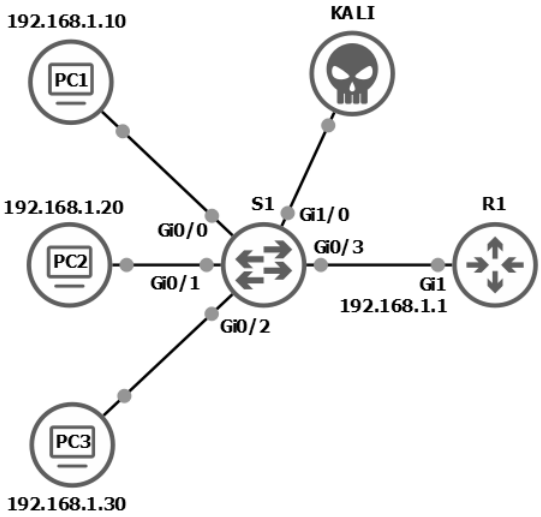
RYSUNEK 4.11. Graf ramek ARP

MAC flooding (przepełnienie tablicy CAM)

Atak typu *MAC flooding* należy do klasycznych zagrożeń bezpieczeństwa w warstwie drugiej modelu OSI i jest ściśle związany z zasadą działania sieci Ethernet oraz przełączników mostkujących. Szczerze mówiąc, w wielu sieciach, z którymi się spotkałem, ten atak niestety wciąż może się powieść. Ethernet jako technologia warstwy łącza danych jest definiowany przez rodzinę standardów IEEE 802.3, natomiast logika mostkowania, filtrowania i przekazywania ramek między portami przełącznika wynika z mechanizmów opisanych obecnie w rodzinie standardów IEEE 802.1, w praktyce odnoszonych do mostów i sieci mostkowanych. Przełącznik warstwy drugiej podejmuje decyzję o przekazaniu ramki na podstawie adresu MAC odbiorcy oraz własnej tablicy skojarzeń adresów z portami. Jeżeli przełącznik zna port przypisany do danego adresu docelowego, przekazuje ramkę wyłącznie tam, gdzie powinien. Jeżeli jednak adres docelowy nie jest znany, ramka może zostać rozesłana na porty należące do tego samego segmentu logicznego. Właśnie tę cechę próbuje wykorzystać atak *MAC flooding*.

Spójrz na rysunek 4.12. Znajduje się na nim topologia, która posłuży Ci do tego, aby samemu zasymulować tego typu atak, by później przeanalizować przechwycony ruch. W topologii do przełącznika `S1` podłączono trzy hosty użytkowników, stację atakującą `KALI` oraz router `R1` pełniący funkcję bramy domyślnej dla sieci `192.168.1.0/24`. Hosty `PC1`, `PC2` i `PC3` pracują w tej samej domenie rozgłoszeniowej, a ich ruch przechodzi przez centralny przełącznik.

Aby przeprowadzić symulację, a właściwie atak, przejdź do stacji `KALI`. Do wykonania ataku wykorzystasz narzędzie macof, wchodzące w skład pakietu dsniff. W oficjalnym opisie dystrybucji Kali Linux narzędzie to zostało scharakteryzowane wprost jako program służący do zalewania sieci lokalnej losowymi adresami MAC. Jego zadaniem nie jest więc prowadzenie zwykłej komunikacji, lecz wygenerowanie bardzo dużej liczby ramek



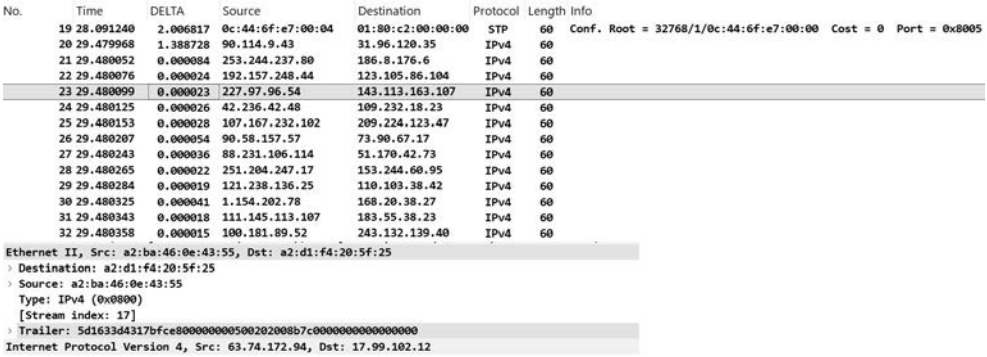
RYSUNEK 4.12. Topologia składająca się z trzech stacji roboczych, routera i stacji atakującej KALI

Ethernet z dynamicznie zmienianymi polami źródłowymi i docelowymi, tak aby przełącznik był zmuszony do ciągłego uczenia się nowych wpisów. Aby rozpocząć symulację, użyj polecenia `sudo macof -i eth0 -n 50000`, które uruchamia generowanie dużej liczby ramek przez interfejs `eth0`. Początek ataku ilustruje rysunek 4.13.

```
┌──(kali㉿kali)-[~]
└─$ sudo macof -i eth0 -n 50000
[sudo] password for kali:
6c:2c:4e:60:3b:c8 3:ee:c3:66:c6:59 0.0.0.0.58841 > 0.0.0.0.25838: S 227659444:227659444(0) win 512
bf:8f:c6:2:98:5a df:bb:f7:21:ee:85 0.0.0.0.33452 > 0.0.0.0.13056: S 2098391686:2098391686(0) win 512
23:9d:3b:7b:38:cf 46:a2:7e:46:db:ee 0.0.0.0.12839 > 0.0.0.0.29572: S 722459569:722459569(0) win 512
0:7c:44:7a:1d:8 34:ff:99:40:3:6a 0.0.0.0.51746 > 0.0.0.0.18364: S 497931901:497931901(0) win 512
b9:c4:6b:23:3c:66 6f:f3:24:46:99:80 0.0.0.0.35155 > 0.0.0.0.28374: S 1742216045:1742216045(0) win 512
d5:8:a0:75:90:b8 11:43:83:2:1b:ab 0.0.0.0.5271 > 0.0.0.0.55699: S 1582322774:1582322774(0) win 512
8c:10:a0:2e:56:d0 66:f8:e2:5a:c1:f 0.0.0.0.33104 > 0.0.0.0.1148: S 161194502:161194502(0) win 512
53:9d:30:50:bf:66 9:be:ff:18:c3:90 0.0.0.0.31024 > 0.0.0.0.28843: S 1614354698:1614354698(0) win 512
85:e4:a4:7:df:21 2d:4d:54:78:96:27 0.0.0.0.2152 > 0.0.0.0.57746: S 2075053612:2075053612(0) win 512
da:d1:f1:5:b8:de 14:2d:89:70:40:ca 0.0.0.0.9354 > 0.0.0.0.28617: S 1749314539:1749314539(0) win 512
96:c7:f1:6c:3e:9 4f:2d:bb:28:9f:85 0.0.0.0.12243 > 0.0.0.0.4913: S 1503954259:1503954259(0) win 512
```

RYSUNEK 4.13. Atak za pomocą narzędzia macof ze stacji KALI

Aby zebrać informacje, jakie są przesyłane podczas ataku, włącz przechwytywanie na łączu między stacją `KALI` a przełącznikiem `S1`. Na rysunku 4.14 znajduje się fragment przechwyconego ruchu. Możesz tam zauważyć bardzo dużą liczbę krótkich ramek Ethernet, pojawiających się w niezwykle małych odstępach czasu. Charakterystyczne jest to, że kolejne ramki zawierają stale zmieniające się adresy MAC nadawcy, a często także nietypowe lub losowe adresy IP w polach wyższych warstw. Taki obraz nie odpowiada normalnej komunikacji użytkowej i jeśli coś takiego zobaczysz w rzeczywistej sieci, od razu powinna zapalić Ci się czerwona lampka. W standardowym ruchu stacja robocza lub serwer używa zwykle jednego, stabilnego adresu MAC przypisanego do interfejsu, natomiast na rysunku źródłowe identyfikatory warstwy drugiej zmieniają się lawinowo. Dla analityka jest to podstawowy artefakt wskazujący na próbę przeciążenia procesu uczenia tablicy przełącznika. Istotne znaczenie ma również analiza czasu. Jeżeli w ułamkach sekundy pojawiają się setki kolejnych ramek, oznacza to, że przełącznik otrzymuje ogromną liczbę nowych wartości *eth.src* w bardzo krótkim oknie czasowym, a więc może bardzo szybko utracić zdolność do utrzymywania stabilnych skojarzeń adresów z portami.



No.	Time	DELTA	Source	Destination	Protocol	Length	Info
19	28.091240	2.006817	0c:44:6f:e7:00:04	01:80:c2:00:00:00	STP	60	Conf. Root = 32768/1/0c:44:6f:e7:00:00 Cost = 0 Port = 0x8005
20	29.479968	1.388728	90.114.9.43	31.96.120.35	IPv4	60	
21	29.480052	0.000084	253.244.237.80	186.8.176.6	IPv4	60	
22	29.480076	0.000024	192.157.248.44	123.105.86.104	IPv4	60	
23	29.480099	0.000023	227.97.96.54	143.113.163.107	IPv4	60	
24	29.480125	0.000026	42.236.42.48	109.232.18.23	IPv4	60	
25	29.480153	0.000028	107.167.232.102	209.224.123.47	IPv4	60	
26	29.480207	0.000054	90.58.157.57	73.90.67.17	IPv4	60	
27	29.480243	0.000036	88.231.106.114	51.170.42.73	IPv4	60	
28	29.480265	0.000022	251.204.247.17	153.244.60.95	IPv4	60	
29	29.480284	0.000019	121.238.136.25	110.103.38.42	IPv4	60	
30	29.480325	0.000041	1.154.202.78	168.20.38.27	IPv4	60	
31	29.480343	0.000018	111.145.113.107	183.55.38.23	IPv4	60	
32	29.480358	0.000015	100.181.89.52	243.132.139.40	IPv4	60	

```
Ethernet II, Src: a2:ba:46:0e:43:55, Dst: a2:d1:f4:20:5f:25
› Destination: a2:d1:f4:20:5f:25
› Source: a2:ba:46:0e:43:55
  Type: IPv4 (0x0800)
  [Stream index: 17]
› Trailer: 5d1633d4317bfce8000000005002020 08b7c000000000000000
Internet Protocol Version 4, Src: 63.74.172.94, Dst: 17.99.102.12
```

RYSUNEK 4.14. Ruch przechwycony na łączu KALI–S1

Następny przechwyt, znajdujący się na rysunku 4.15, wykonano na łączu między hostem PC1 a przełącznikiem S1 i właśnie ten punkt obserwacji ma największą wartość poznawczą w przypadku omówienia tego ataku. W normalnie działającej sieci przełączanej host końcowy nie powinien widzieć obcego ruchu unicast, który nie jest do niego adresowany. Tymczasem w przechwyceniu po stronie PC1 pojawiają się ramki IPv4 z adresami źródłowymi i docelowymi niezwiązanymi z komunikacją tej stacji. Z analitycznego punktu widzenia jest to kluczowa przesłanka wskazująca, że przełącznik przestał poprawnie filtrować część ruchu unicast na podstawie znanych wpisów tablicy adresów. Cisco wyjaśnia, że przy nieznanym adresie docelowym ramka jest rozsyłana na wszystkie porty w danym VLAN-ie z wyjątkiem portu wejściowego, a jednym z powodów takiego stanu może być przepełnienie tablicy przekazywania. W praktyce oznacza to, że część ramek zaczyna być traktowana jako unknown unicast i trafia również na porty, które w poprawnie funkcjonującej sieci nie powinny ich otrzymywać. To właśnie tłumaczy obecność obcego ruchu na łączu PC1–S1.

RYSUNEK 4.15. Ruch przechwycony na łączu PC1–S1

No.	Time	DELTA	Source	Destination	Protocol	Length	Info
23	25.501009	0.002649	227.97.96.54	143.113.163.107	IPv4	60	
24	25.507191	0.006182	227.97.96.54	143.113.163.107	IPv4	60	
25	25.509673	0.002482	42.236.42.48	109.232.18.23	IPv4	60	
26	25.514368	0.004695	107.167.232.102	209.224.123.47	IPv4	60	
27	25.516426	0.002058	107.167.232.102	209.224.123.47	IPv4	60	
28	25.520197	0.003771	90.58.157.57	73.90.67.17	IPv4	60	
29	25.521710	0.001513	90.58.157.57	73.90.67.17	IPv4	60	
30	25.525199	0.003489	88.231.106.114	51.170.42.73	IPv4	60	
31	25.526833	0.001634	251.204.247.17	153.244.60.95	IPv4	60	
32	25.530790	0.003957	121.238.136.25	110.103.38.42	IPv4	60	
33	25.532806	0.002016	121.238.136.25	110.103.38.42	IPv4	60	
34	25.536395	0.003589	1.154.202.78	168.20.38.27	IPv4	60	
35	25.538427	0.002032	1.154.202.78	168.20.38.27	IPv4	60	
36	25.542459	0.004032	111.145.113.107	183.55.38.23	IPv4	60	
37	25.544414	0.001955	100.181.89.52	243.132.139.40	IPv4	60	
38	25.548748	0.004334	242.65.213.15	67.154.61.71	IPv4	60	
39	25.550917	0.002169	63.74.172.94	17.99.102.12	IPv4	60	

```
› Frame 39: Packet, 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface -, id 0
⌄ Ethernet II, Src: a2:ba:46:0e:43:55, Dst: a2:d1:f4:20:5f:25
  › Destination: a2:d1:f4:20:5f:25
  › Source: a2:ba:46:0e:43:55
    Type: IPv4 (0x0800)
    [Stream index: 17]
  › Trailer: 5d1633d4317bfce8000000005002020 08b7c000000000000000
› Internet Protocol Version 4, Src: 63.74.172.94, Dst: 17.99.102.12
```

Zapewne zauważysz, że w środowisku laboratoryjnym efekt ataku pojawi się bardzo szybko. Wynika to z samej natury ataku. Przełącznik uczy się adresu źródłowego MAC

z każdej odebranej ramki, dlatego masowe generowanie ramek z losowymi adresami prowadzi do gwałtownego wzrostu liczby potencjalnych wpisów. Jeżeli generator pracuje z dużą intensywnością, to już po krótkim czasie mogą być widoczne objawy utraty selektywnego przełączania. W realnych wdrożeniach wpływ na skalę i tempo zjawiska ma pojemność tablic sprzętowych, architektura przełącznika, mechanizmy ochronne producenta oraz charakterystyka ruchu w danym VLAN-ie, jednak mechanizm analityczny pozostaje ten sam: nagły wzrost liczby źródłowych adresów MAC po stronie atakującego i pojawienie się obcego ruchu unicast po stronie hosta nieuczestniczącego w wymianie danych.

Na koniec chciałbym jeszcze zwrócić uwagę na to, że z perspektywy obronnej najważniejsze jest ograniczenie możliwości niekontrolowanego uczenia się wielu adresów MAC na portach dostępowych oraz utrudnienie rozlewania nieznanego ruchu unicast. W praktyce stosuje się funkcje typu port security, odpowiednią segmentację sieci VLAN, ograniczanie liczby adresów MAC dopuszczanych na porcie, a także mechanizmy wykrywania anomalii w ruchu warstwy drugiej. Istotne znaczenie ma również właściwe projektowanie sieci, ponieważ skutki przepełnienia tablic przełączania są szczególnie niebezpieczne tam, gdzie wiele hostów współdzieli ten sam rozległy segment VLAN-u.

Jeśli podczas analizy ruchu sieciowego w rzeczywistej sieci coś budzi Twoje obawy, wówczas możesz posłużyć się dodatkowymi filtrami, które same w sobie nie wykryją ataku, ale pozwolą Ci upewnić się o jego wystąpieniu.

Najbardziej podstawowy filtr, od którego warto zacząć, to po prostu `eth`. Wprawdzie nie wykrywa ataku, ale porządkuje analizę i pozwala skupić się na warstwie drugiej. W przechwycie z portu atakującego taki filtr szybko ujawnia nienaturalnie dużą liczbę krótkich ramek Ethernet pojawiających się w bardzo małych odstępach czasu.

Jeżeli chcesz pokazać ruch z jednego konkretnego hosta atakującego, bardzo przydatne jest filtrowanie po fizycznym adresie interfejsu tego hosta, o ile w danym scenariuszu chociaż część ramek rzeczywiście wychodzi z jego prawdziwego adresu MAC. W praktyce przy macof to nie zawsze będzie najważniejszy trop, bo narzędzie właśnie zmienia pola źródłowe, ale w niektórych przechwyceniach i przy wstępnej orientacji może pomóc filtr `eth.addr == aa:bb:cc:dd:ee:ff` albo precyzyjniej `eth.src == aa:bb:cc:dd:ee:ff`.

Bardzo użyteczny jest filtr odrzucający ruch rozgłoszeniowy i multicastingowy. Dzięki niemu łatwiej zobaczysz nietypowy unicast, który pojawia się na porcie ofiary. Możesz użyć np. `eth.dst != ff:ff:ff:ff:ff:ff`, co pozwala odsiać broadcasty. A jeżeli chcesz jeszcze mocniej zawęzić obraz do ramek unicast, możesz użyć filtru `!(eth.dst[0] & 1)`. Filtr sprawdza najmłodszy bit pierwszego oktetu adresu docelowego MAC. Jeżeli jest wyzerowany, adres jest unicastowy. Taki filtr może być bardzo pomocny, gdyż na porcie pozwala pokazać rzecz najważniejszą: nie chodzi o to, że host widzi broadcast, bo to jest normalne, lecz o to, że zaczyna widzieć cudzy ruch unicast, który normalnie nie powinien do niego trafiać.

Tak więc bardzo dobry filtr do opisu skutków ataku na porcie hosta może wyglądać tak: `!(eth.dst[0] & 1) && !(eth.addr == MAC_PC1)`, gdzie *MAC_PC1* podstawiasz jako rzeczywisty adres MAC komputera `PC1`. Filtr po prostu pokaże ramki unicast, które nie są ani wysyłane przez stację, ani do niej adresowane. Jeżeli po zastosowaniu takiego filtru nadal widzisz ruch, to dla analityka jest to bardzo mocna przesłanka, że port hosta otrzymuje cudzy unicast, co może wskazywać na zaburzenie pracy przełącznika po ataku *MAC flooding*.

Przydatny może być również filtr na krótkie ramki. Będzie pomocny, bo generatory typu macof często produkują masowo niewielkie ramki. Zatem filtr może wyglądać tak: `frame.len == 60` albo szerzej: `frame.len <= 64`. Pamiętaj jednak, że krótkie ramki same w sobie nie są dowodem ataku, ale w połączeniu z ogromną liczbą zmieniających się adresów źródłowych MAC oraz bardzo małymi odstępami czasu mogą być użytecznym wskaźnikiem. Nie zapomnij o obserwacji kolumny *DELTA* (albo wykresów I/O). Po zastosowaniu filtrów takich jak `eth` albo `eth.type == 0x0800` powinieneś zwracać uwagę na gwałtowne serie ramek pojawiających się w odstępach liczonych w tysięcznych częściach sekundy. To jest w praktyce ważniejsze niż pojedynczy pakiet.

Chciałbym zwrócić uwagę jeszcze na to, że *MAC flooding* nie ma jednego uniwersalnego filtru wykrywającego atak w sposób bezbłędny. Wykrycie opiera się na korelacji kilku symptomów: bardzo dużej liczby ramek w krótkim czasie, szybkich zmian adresów *eth.src*, nietypowej liczby krótkich ramek oraz pojawienia się obcego ruchu unicast na portach, które normalnie nie powinny go odbierać. Zresztą, jak zapewne już zauważyłeś, powtarzam to przy okazji większości analiz.

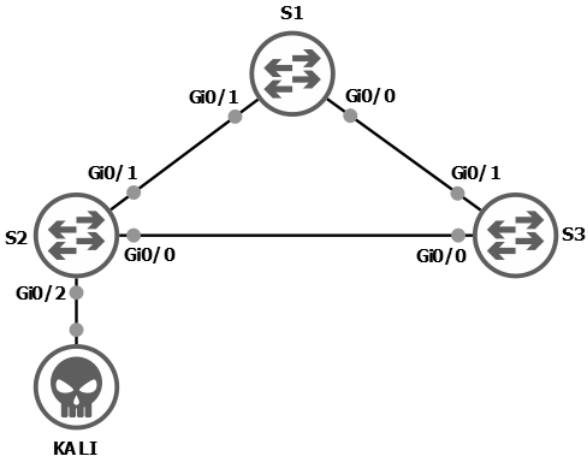
Wykrywanie STP manipulation

Już wiesz, że *Spanning Tree Protocol* należy do podstawowych mechanizmów ochrony topologii przełączanej przed pętlami w warstwie drugiej. Jego zadaniem jest wybór jednego przełącznika pełniącego funkcję root bridge, a następnie wyznaczenie takiej struktury aktywnych połączeń, aby przełączana domena pozostała spójna i jednocześnie wolna od pętli.

W praktyce analitycznej oznacza to, że każda nieautoryzowana zmiana roota lub nagła zmiana parametrów BPDU powinna być traktowana jako zdarzenie potencjalnie niebezpieczne. Mechanizm działania STP i semantyka BPDU wynikają ze standardów IEEE dla mostkowania i drzewa rozpinającego, przede wszystkim z rodziny IEEE 802.1, historycznie kojarzonej z IEEE 802.1D. W odróżnieniu od wielu protokołów warstwy trzeciej i wyższych, STP nie jest klasycznie opisany przez RFC IETF, lecz przez standardy IEEE. Dokumenty IEEE wskazują, że mosty wymieniają BPDU zawierające informacje identyfikujące root bridge, koszt ścieżki do roota oraz identyfikator mostu nadającego wiadomość, a właśnie te pola są dla analityka najważniejsze.

Zacznijmy więc od utworzenia topologii przedstawionej na rysunku 4.16, abyś mógł najpierw przeprowadzić atak, a potem przechwycić ruch i nauczyć się rozpoznawać jego oznaki. W badanym środowisku trzy przełączniki, oznaczone jako `S1`, `S2` i `S3`, tworzą trójkąt połączeń warstwy drugiej, natomiast do przełącznika `S2` został dołączony host `KALI`. Taki układ jest bardzo dobry do analizy bezpieczeństwa STP, bo pozwala nie tylko wywołać zmianę w logice drzewa rozpinającego, ale również zaobserwować jej skutki na innym łączu niż to, którym host atakujący jest bezpośrednio wpięty do sieci. Z punktu widzenia analityka ma to duże znaczenie, ponieważ w rzeczywistym środowisku przechwyt bardzo często wykonuje się właśnie z miejsca pośredniego, a nie bezpośrednio na porcie napastnika. Przełączniki nie wymagają konfiguracji. Wystarczy, że podłączysz stację `KALI` i rozpoczniesz atak.

RYSUNEK 4.16. Topologia z trzema przełącznikami



W pierwszej kolejności sprawdźmy, który przełącznik pełni funkcję root bridge przed rozpoczęciem ataku. W tym celu na przełączniku `S2` wykonaj polecenie `show spanning-tree`. W przedstawionym przykładzie wynik wskazuje jednoznacznie, że to właśnie `S2` jest rootem dla `VLAN 1`. Widać to po zgodności pola `Root ID` z własnym `Bridge ID` przełącznika oraz po komunikacie, że bieżący most jest rootem. Zauważ również, że interfejs `Gi0/2` prowadzący do stacji `KALI` jest w trybie `Desg`.

W stabilnej domenie STP wszystkie przełączniki powinny propagować w BPDU informację o tym samym root bridge. Jeżeli więc root zostaje nagle zmieniony bez planowanej rekonfiguracji, analityk powinien potraktować to jako zdarzenie podejrzane. Mechanizm działania STP oraz zawartość BPDU są definiowane przez standardy IEEE z rodziny 802.1, historycznie kojarzone z IEEE 802.1D. To właśnie te standardy opisują wybór root bridge, obliczanie kosztów ścieżek i role portów w drzewie rozpinającym.

```
S2#show spann
VLAN0001
  Spanning tree enabled protocol ieee
  Root ID    Priority    32769
             Address     0c7b.930c.0000
             This bridge is the root
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
  Bridge ID  Priority    32769  (priority 32768 sys-id-ext 1)
             Address     0c7b.930c.0000
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time  15  sec
Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
Gi0/0               Desg FWD 4         128.1    P2p
Gi0/1               Desg FWD 4         128.2    P2p
Gi0/2               Desg FWD 4         128.3    P2p
```

Gdy stan początkowy został już potwierdzony, można przejść do symulacji ataku. Celem włamywacza w takim przypadku jest zrobić wszystko, aby jego stacja została uznana za `Root Bridge`.

W laboratorium wykorzystasz narzędzie yersinia, uruchomione poleceniem `sudo yersinia -I` (rysunek 4.17). W trybie interaktywnym naciśnij `g`, aby wybrać protokół (wybierz protokół STP). Jeśli chcesz, możesz zmienić parametry ramek wysyłanych podczas ataku.

Opcje te zmienisz po naciśnięciu litery `e` (*edit*). Następnie na dole zmień to, co chcesz zmienić. Teraz kliknij literę `x`, aby otworzyć menu ataków (*Execute attack*), i w menu wybierz opcję `4` (*Claiming Root Role*). Sens tego działania polega na wysyłaniu spreparowanych ramek BPDU, które mają sprawić, że legalny przełącznik uzna host atakujący za bardziej korzystnego kandydata na root bridge. Z perspektywy administracyjnej oznacza to próbę przejęcia kontroli nad logiką drzewa rozpinającego, natomiast z perspektywy analityka najważniejsze jest to, że taka operacja powinna pozostawić ślady zarówno w stanie przełączników, jak i w przechwyconym ruchu STP. Zanim rozpoczniesz atak, włącz przechwytywanie ruchu na łączu `S1`–`S3`.

RYSUNEK 4.17. Panel służący do wyboru ataku

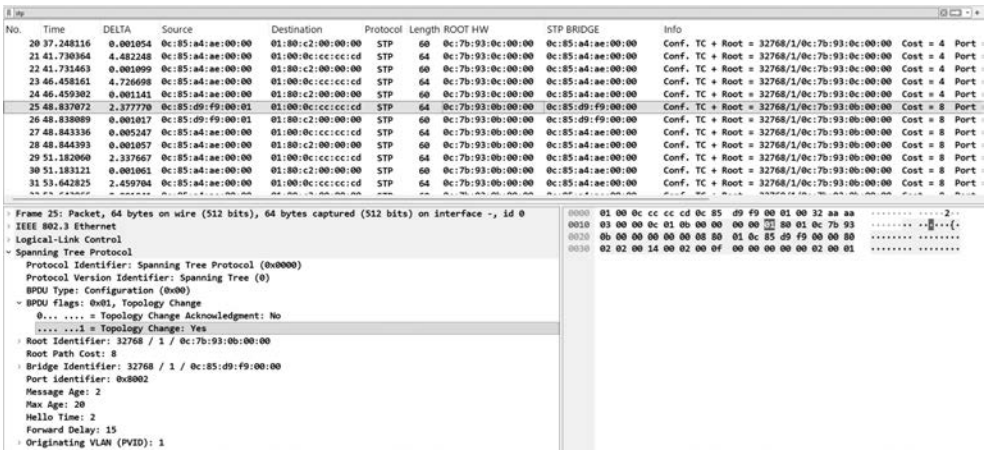
Po uruchomieniu ataku sprawdźmy najpierw ponownie stan przełącznika `S2` za pomocą tego samego polecenia `show spanning-tree`. Zauważ, że poniższy listing przedstawia teraz inny wynik. `S2` nie jest już rootem. Pole `Root ID` wskazuje nowy identyfikator mostu, a port prowadzący do hosta `KALI` staje się root portem. Oznacza to, że przełącznik `S2` zaakceptował informację o nowym root bridge i zaczął traktować ścieżkę prowadzącą przez host końcowy jako najlepszą drogę do roota. W normalnej sytuacji host użytkownika nie powinien mieć możliwości przejęcia roli root bridge w domenie przełączanej. Jeżeli więc port, do którego podłączony jest zwykły host, nagle staje się root portem przełącznika infrastrukturalnego, to mamy do czynienia z objawem klasycznej manipulacji STP.

```
S2#sh spann
VLAN0001
  Spanning tree enabled protocol ieee
  Root ID    Priority    32769
             Address     0c7b.930b.0000
             Cost        4
             Port        3 (GigabitEthernet0/2)
```

```
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
  Bridge ID  Priority    32769  (priority 32768 sys-id-ext 1)
             Address     0c7b.930c.0000
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time  15  sec
Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
Gi0/0               Desg FWD 4         128.1    P2p
Gi0/1               Desg FWD 4         128.2    P2p
Gi0/2               Root FWD 4         128.3    P2p
Gi0/3               Desg FWD 4         128.4    P2p
```

Na tym etapie można by uznać, że atak został potwierdzony, ale z punktu widzenia analityki ruchu sieciowego najciekawsze dopiero przed nami. Sprawdźmy teraz przechwyt wykonany nie na porcie KALI ani nawet nie na łączu S2–KALI, lecz na łączu pomiędzy S1 i S3. Taki wybór miejsca obserwacji jest celowy, ponieważ pozwala odpowiedzieć na ważne pytanie: czy analityk, który nie widzi bezpośrednio ruchu napastnika, nadal jest w stanie wykryć, że w domenie STP dzieje się coś nieprawidłowego? Odpowiedź brzmi: tak, ale wykrywa on przede wszystkim skutek ataku, a nie sam moment jego wstrzyknięcia.

Rysunek 4.18 przedstawia przechwycone dane. W pierwszej kolejności warto zawęzić analizę do ramek STP, np. filtrem `stp`. Warto też wyświetlić w osobnych kolumnach pola związane z BPDU, przede wszystkim *stp.root.hw*, *stp.bridge.hw*, *stp.root.path_cost* oraz *stp.flags.tc*. Dzięki temu jako analityk nie będziesz musiał za każdym razem rozwijać całego drzewa dekodowania pakietu. Od razu zobaczysz najważniejsze wartości na liście ramek. W przechwycie z łącza S1–S3 widać, że ramki STP są nadal wysyłane na właściwy adres grupowy `01:80:c2:00:00:00`, a więc z formalnego punktu widzenia format komunikacji kontrolnej jest poprawny. Problem ujawnia się dopiero po analizie zawartości BPDU.



RYSUNEK 4.18. Ruch STP przechwycony na łączu S1–S3

Zwróć uwagę na pole *Root Identifier*. To właśnie ono ma tutaj znaczenie kluczowe. Jeżeli przed atakiem domena STP wskazywała jednego roota, a po ataku w przechwycie na innym łączu zaczyna pojawiać się inny *Root Identifier*, to znaczy, że zmiana została już przyjęta i jest dalej propagowana przez legalne przełączniki. W omawianym laboratorium jest to dokładnie to, co obserwujemy. Przełączniki nie ogłaszają już poprzedniego, legalnego roota,

lecz nowy identyfikator. Zauważ, że nie widać tu jeszcze yersinii ani bezpośrednio hosta KALI wysyłającego sfałszowane BPDU, ale widać coś równie ważnego, czyli skuteczną zmianę stanu drzewa rozpinającego.

Dalej zwróć uwagę na pole *Bridge Identifier*. Mówi ono, który przełącznik faktycznie wysłał daną BPDU. Jeżeli więc w przechwycie pojawia się sytuacja, w której *Bridge Identifier* należy do legalnego przełącznika, ale *Root Identifier* wskazuje już inny, nowy most, oznacza to, że przełącznik infrastrukturalny sam rozsyła dalej informację o nowym root bridge. To właśnie jest najważniejszy objaw pośredni. Innymi słowy: legalna infrastruktura została zmuszona do uznania nowego roota i zaczęła sama propagować tę zmianę. Z punktu widzenia wykrywania włamań jest to bardzo cenna obserwacja, bo pozwala wykazać incydent nawet wtedy, gdy nie mamy przechwytu z punktu wejścia ataku. Warto również przeanalizować pole *Root Path Cost*. Jeżeli po zmianie roota kolejne przełączniki zaczynają reklamować nowy koszt dojścia do roota, a wartość ta jest logicznie zgodna z topologią, oznacza to, że zaszła rzeczywista rekonwergencja STP. W naszym przypadku widać właśnie taki efekt. Przełącznik znajdujący się bliżej nowego roota raportuje niższy koszt, natomiast urządzenia położone dalej propagują już odpowiednio większą wartość. Dla analityka jest to potwierdzenie, że nie ma do czynienia z pojedynczą anomalią jednej ramki, lecz z realną zmianą w drzewie rozpinającym.

Dodatkową wskazówką może być flaga *Topology Change*. Sama jej obecność nie oznacza jeszcze włamania, ponieważ może pojawiać się również podczas legalnych zmian w topologii. Jeżeli jednak pojawia się równocześnie ze zmianą *Root Identifier*, zmianą ról portów oraz utratą pozycji root bridge przez dotychczasowy przełącznik nadrzędny, to staje się bardzo istotnym elementem korelacji. Analityk nie powinien więc traktować jednej wartości w izolacji, lecz zestawiać razem kilka obserwacji, czyli zmianę roota, zmianę kosztów, zmianę ról portów oraz obecność komunikatów o zmianie topologii.

W tym miejscu chciałbym bardzo wyraźnie podkreślić jedną rzecz. Z przechwytu wykonanego na łączu S1–S3 nie da się zwykle bezpośrednio udowodnić, że to konkretnie host KALI był źródłem pierwotnych sfałszowanych BPDU. Do tego potrzebny byłby przechwyt bliżej portu S2–KALI albo zapis z mechanizmów ochronnych samego przełącznika. Z tego miejsca obserwacji można jednak z dużą pewnością wykazać, że domena STP zmieniła roota w sposób niezgodny z wcześniejszym stanem sieci i że zmiana ta została rozpropagowana przez legalną infrastrukturę. To rozróżnienie jest bardzo ważne. Często w praktyce nie zawsze będziesz identyfikować od razu źródło ataku, ale możesz bardzo skutecznie wykryć jego skutek.

Jeszcze słowo na temat bezpieczeństwa. Aby ograniczyć ryzyko takich incydentów, stosuje się mechanizmy ochronne zapobiegające akceptowaniu BPDU z portów dostępowych oraz wymuszające przewidywalną pozycję legalnych przełączników w drzewie STP. Z punktu widzenia bezpieczeństwa nie chodzi tu tylko o stabilność sieci, ale również o ochronę przed przejęciem wpływu na ścieżki ruchu w warstwie drugiej. Nawet bez wchodzenia w szczegóły konfiguracji warto pamiętać, że dobrze zaprojektowana infrastruktura nie powinna pozwalać zwykłemu hostowi na zmianę roota całej domeny przełączanej.

Podsumowując: w wykrywaniu manipulacji STP kluczowe jest porównanie stanu przed zdarzeniem i po zdarzeniu. Najpierw należy ustalić legalnego roota, następnie sprawdzić, czy po ataku jego rola została przejęta przez inne urządzenie, a na końcu potwierdzić w ewentualnym przechwycie, że nowy *Root Identifier* jest propagowany przez legalne przełączniki.

Użyteczne filtry

Na zakończenie warto poznać jeszcze kilka dodatkowych filtrów, które mogą istotnie ułatwić pracę analityka. Nie są one samodzielnym dowodem ataku, ale w praktyce często pomagają szybciej zawęzić obszar analizy i potwierdzić podejrzenia wynikające z obserwacji ruchu.

Filtr `arp.isgratuitous == 1` pozwala odszukać pakiety *gratuitous ARP*, czyli takie, w których host sam ogłasza mapowanie własnego adresu IP na adres MAC. W normalnej sieci takie ramki mogą pojawiać się legalnie, np. podczas przełączeń awaryjnych lub odświeżania informacji sąsiadów, jednak ich nagłe i częste występowanie bywa bardzo cenną wskazówką przy analizie ARP spoofing. Wireshark udostępnia dla tego zjawiska osobne pole filtru.

Filtr `arp.isprobe == 1` jest przydatny wtedy, gdy chcesz oddzielić zwykłe zapytania ARP od ramek związanych z wykrywaniem konfliktu adresów IPv4. Taki filtr może pomóc odróżnić normalne zachowanie hosta uruchamiającego mechanizm sprawdzania adresu od właściwego etapu zatruwania tablic ARP. Jest to szczególnie użyteczne w środowiskach laboratoryjnych, w których wiele maszyn jest często uruchamianych i wyłączanych. Pole to znajduje się w referencji filtrów Wiresharka dla ARP.

Filtr `arp.duplicate-address-detected` może być bardzo pomocny podczas szukania niespójności w mapowaniach IPv4/MAC. Wireshark potrafi oznaczyć sytuacje sugerujące wcześniejsze użycie tego samego adresu IP i właśnie ten filtr pozwala takie pakiety szybko wyłapać. Nie zastępuje on pełnej analizy incydentu, ale dobrze nadaje się do wstępnego wychwycenia konfliktów, które mogą mieć związek z błędną konfiguracją albo aktywnością atakującego. To pole jest opisane w referencji filtrów ARP Wiresharka.

Filtr `arp.dst.proto_ipv4 == 192.168.1.1` pozwala zawęzić widok do ramek ARP, których celem jest konkretny adres IPv4. W praktyce jest to bardzo wygodne, gdy chcesz sprawdzić, czy wiele stacji nagle zaczęło pytać o adres bramy domyślnej albo wybranego hosta. Taki filtr dobrze uzupełnia analizę ARP, bo zamiast patrzyć wyłącznie na nadawcę, możesz od razu skupić się na ofierze lub na adresie, pod który ktoś się podszywa. Pole *arp.dst.proto_ipv4* jest dostępne w oficjalnej referencji Wiresharka.

Filtr `tcp.flags.reset == 1` może być bardzo pomocny przy analizie skanowania TCP. W rozdziale omówiłem już filtrowanie pakietów `SYN`, jednak równie wartościowe bywa spojrzenie od drugiej strony, czyli na odpowiedzi resetujące. W przypadku intensywnego skanowania portów zamkniętych analityk może zobaczyć wyraźne serie ramek `RST` lub `RST/ACK`, które dobrze pokazują, jak wiele prób dotarło do nieaktywnych usług. Pole *tcp.flags.reset* znajduje się w oficjalnej referencji filtrów TCP Wiresharka.

Filtr `icmp.type == 3 && icmp.code == 3` pozwala od razu wyświetlić komunikaty *ICMP Destination unreachable/Port unreachable*. Jest to bardzo użyteczne przy analizie skanowania UDP, ponieważ takie odpowiedzi często stanowią podstawę do wnioskowania, że port UDP jest zamknięty. Zamiast przeglądać cały ruch ICMP, możesz od razu ograniczyć widok do najważniejszego kodu diagnostycznego. Ten filtr dobrze porządkuje analizę wtedy, gdy przechwyt zawiera wiele różnych komunikatów ICMP. Opis składni filtrów znajduje się w dokumentacji Wiresharka.

Filtr `udp.dstport == 53` (albo analogicznie inny wybrany port UDP) jest przydatny wtedy, gdy chcesz sprawdzić, czy odpytywanie nie koncentrowało się na konkretnych usługach. W praktyce analitycznej pozwala to szybko odróżnić szeroki skan całego zakresu

portów od bardziej ukierunkowanego działania, np. tylko wobec DNS, SNMP czy NTP. Taki filtr nie wykrywa skanu, ale świetnie wspiera analizę intencji intruza i zakresu jego rozpoznania. Referencja pól filtrów TCP i UDP jest dostępna w dokumentacji Wiresharka.

Filtr `stp.flags.tc == 1` pozwala wyłapać ramki BPDU, w których ustawiono znacznik *Topology Change*. W normalnej pracy sieci jego obecność nie musi oznaczać ataku, ale jeśli pojawia się nagle i jednocześnie z innymi oznakami rekonwergencji, może być bardzo ważnym elementem korelacji.

Filtr `stp.root.hw == xx:xx:xx:xx:xx:xx` jest bardzo praktyczny wtedy, gdy chcesz sprawdzić, które ramki reklamują konkretnego roota. Dzięki temu można szybko zobaczyć moment, w którym legalna infrastruktura zaczyna propagować już nowy identyfikator root bridge.

Skorowidz

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z

Ż

WIRESHARK

Analiza ruchu sieciowego i wykrywanie włamań

By wiedzieć, co w sieci piszczy

Świadomość tego, co i w jaki sposób jest przekazywane za pomocą sieci, to istotna sprawa. Szczególnie w kontekście potencjalnych ataków i włamań. Jednym z programów pozwalających przechwycić, przeanalizować w czasie rzeczywistym i szczegółowo przebadać pakiety danych przesyłanych w sieciach komputerowych jest Wireshark. Umożliwia on między innymi diagnozowanie problemów sieciowych, wykrywanie ataków i optymalizowanie wydajności sieci.

Jeśli chcesz wiedzieć o nim więcej, poznać pełny pakiet możliwości Wiresharka i nauczyć się z nich korzystać, najnowsza książka Adama Józefioka z pewnością Ci w tym pomoże. Za jej sprawą między innymi opanujesz podstawowe i zaawansowane funkcje programu. Dowiesz się, jak z udziałem Wiresharka przechwytywać i analizować ruch w sieci, wykrywać włamania do niej, a także monitorować jej bezpieczeństwo. Dzięki lekturze poradnika i przetestowaniu opisanych w nim narzędzi poznasz również statystyczne i analityczne funkcje tego przydatnego oprogramowania.

Helion

helion.pl

HELION S.A.
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!

ISBN 978-83-289-2639-4

9 788328 926394

Cena: 149,00 zł

Adam Józefiok

Ukończył studia doktoranckie na Politechnice Śląskiej w Gliwicach, na Wydziale Automatyki, Elektroniki i Informatyki. Specjalizuje się w tematyce sieci komputerowych (przełączanie, routing, bezpieczeństwo i projektowanie). Jest autorem publikacji polskich i zagranicznych z tej dziedziny. Brał udział w konferencjach naukowych (krajowych i międzynarodowych) dotyczących sieci komputerowych. Jest pracownikiem naukowym Katedry Sieci i Systemów Komputerowych Politechniki Śląskiej. Na co dzień administruje również bezpieczeństwem urządzeń sieciowych, konfiguruje między innymi urządzenia sieciowe (Cisco, HP), utrzymuje sieci WAN. Przez siedem lat kierował komórką realizacji wsparcia usług IT. Ma wieloletnie doświadczenie w pracy jako administrator sieci i projektant sieci komputerowych. Przez lata projektował serwerownie i tworzył projekty okablowania. Opracowywał procedury i dokumentacje projektowe. Na koncie ma wiele zrealizowanych projektów w zakresie zakupu sprzętu IT wraz z prowadzeniem procedur przetargowych, wdrożeniem, wykonaniem dokumentacji i testami. Posiada certyfikaty: CCNA Security, CCNP Routing and Switching, Cisco CCDP, CCNAV, jak również certyfikat instruktorski Cisco CCAI, certyfikaty ITIL i PRINCE2. Jego pasją jest pisanie książek, praca ze studentami i szeroko rozumiana dydaktyka.