New Agent Tesla Variant Being Spread by Crafted Excel Document | FortiGuard Labs

By Xiaopeng Zhang

Published: 2023-09-05 · Archived: 2026-04-05 15:29:58 UTC

Affected platforms: Microsoft Windows
Impacted parties: Windows Users
Impact: Collects sensitive information from a victim's computer
Severity level: Critical

Our FortiGuard Labs captured a phishing campaign that spreads a new Agent Tesla variant. This well-known malware family uses a .Net-based Remote Access Trojan (RAT) and data stealer to gain initial access. It is often used for Malware-as-a-Service (MaaS).

I performed an in-depth analysis of this campaign, from the initial phishing email to the actions of Agent Tesla installed on the victim's machine to the collecting of sensitive information from the affected device. In this analysis, you will learn about the contents of this attack, such as how the phishing email starts the campaign, how the CVE-2017-11882/CVE-2018-0802 vulnerability (and not the VBS macro) is exploited to download and execute the Agent Tesla file on the victim's device, as well as how Agent Tesla collects the sensitive data from the victim's device, such as the credentials, key loggings, and screenshots of the victim's screen.

Despite fixes for CVE-2017-11882/CVE-2018-0802 being released by Microsoft in November, 2017 and January, 2018, this vulnerability remains popular amongst threat actors, suggesting there are still unpatched devices in the wild, even after over five years. We are observing and mitigating 3000 attacks per day, at the IPS level. The number of observed vulnerable devices is around 1300 per day.

Phishing Email

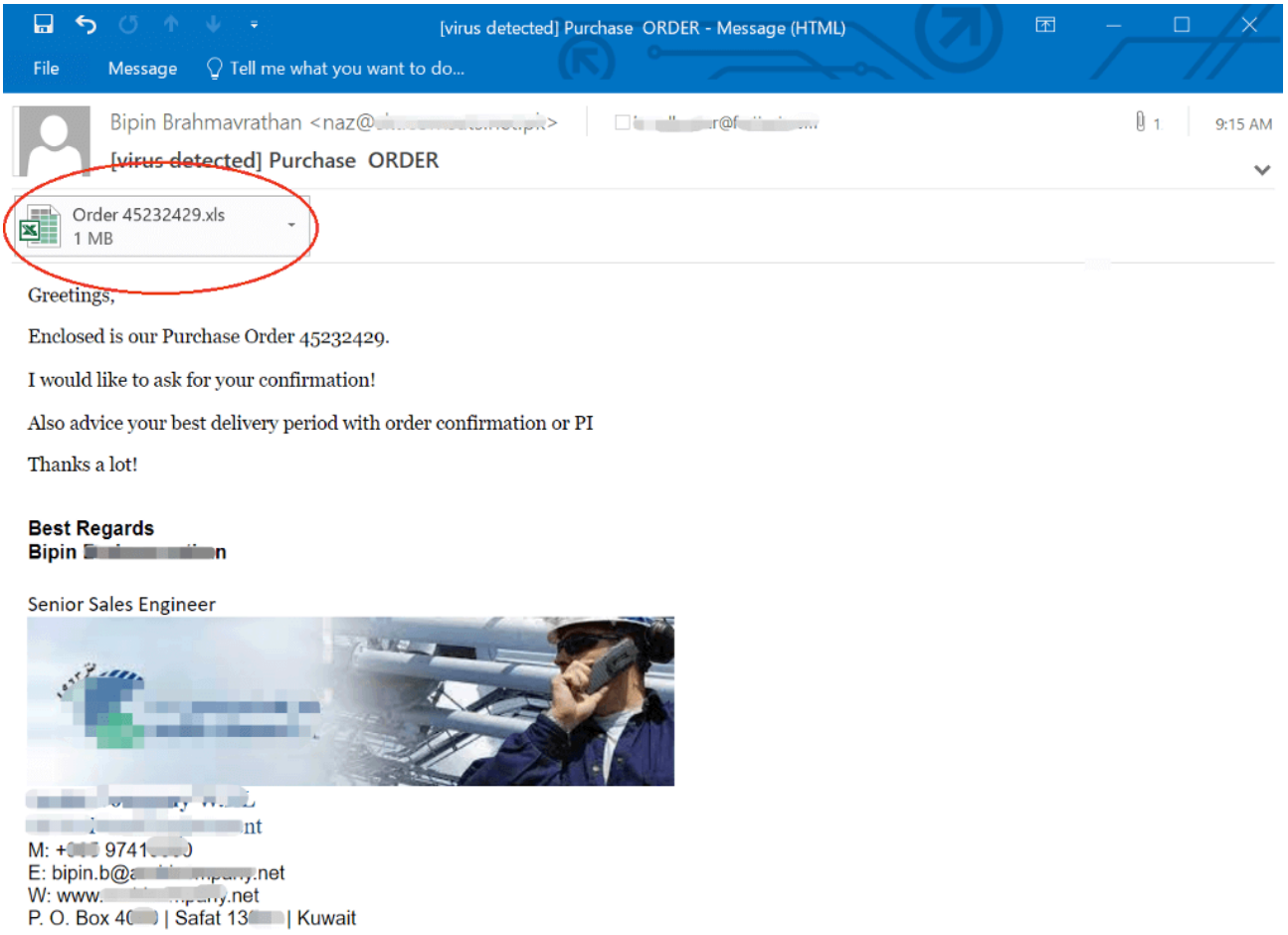


Figure 1.1: The captured phishing email

The phishing email is disguised as a Purchase Order notification, shown in Figure 1.1, that asks for the recipient to confirm an order from an industrial equipment supplier company. An Excel document is attached to this email called “Order 45232429.xls”.

CVE-2017-11882/CVE-2018-0802 Exploited by the Excel Document

The attached Excel document is in OLE format. It contains crafted equation data that exploits the CVE-2017-11882/CVE-2018-0802 vulnerability to execute a malicious shellcode.

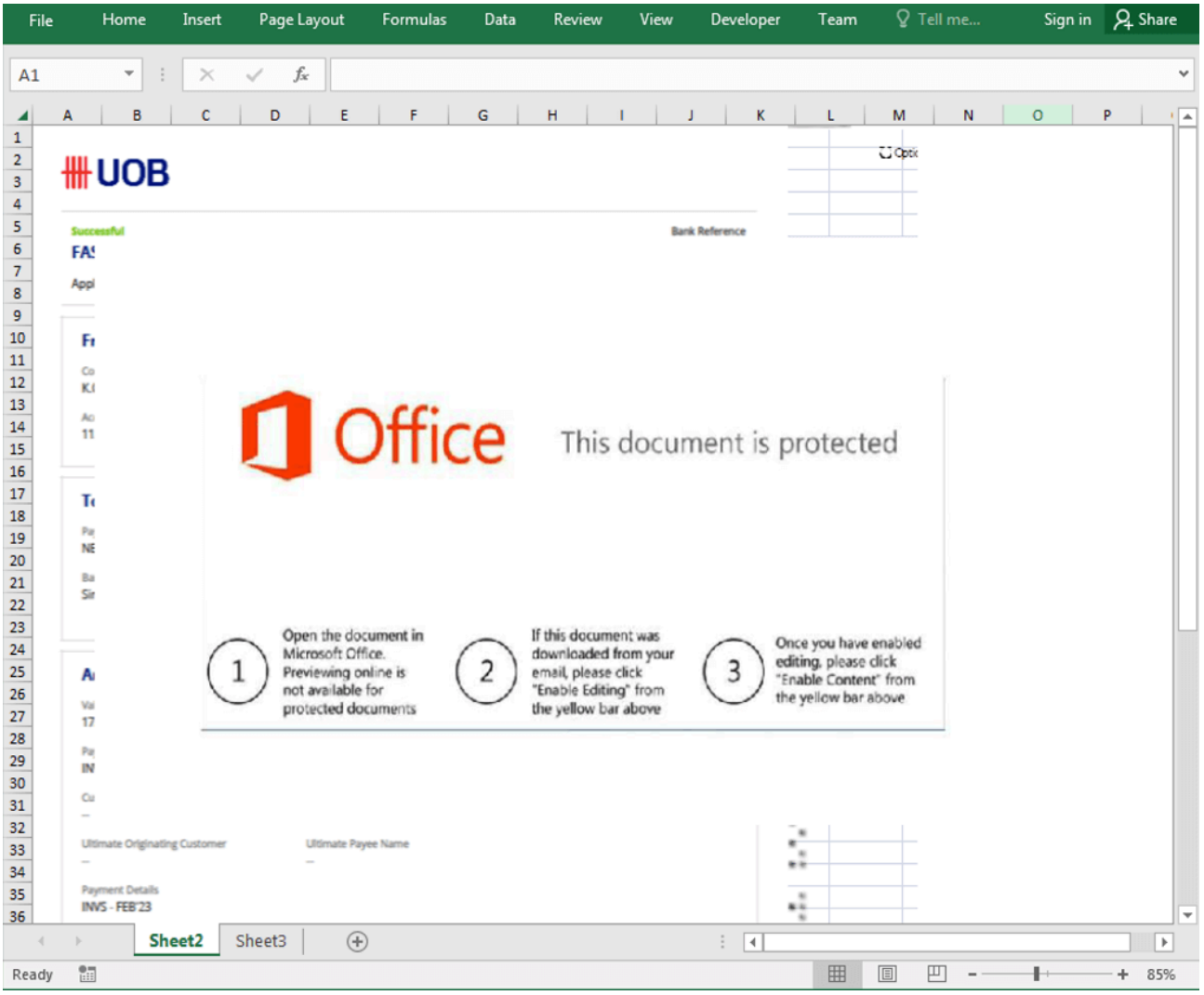


Figure 2.1: The content of the Excel file

Opening the attached Excel document displays a deceptive message to the user (Figure 2.1). Meanwhile, the shellcode inside the crafted equation data is secretly executed.

CVE-2017-11882/CVE-2018-0802 is an RCE (remote code execution) vulnerability that results in memory corruption inside the EQNEDT32.EXE process when parsing the crafted equation data when exploited. This can lead to arbitrary code execution.

Figure 2.2 shows the Excel document parsed in an OLE compound reader, where the equation data is inside the stream “\x01Ole10NativE” under the storage folder “MBD0057E612”.

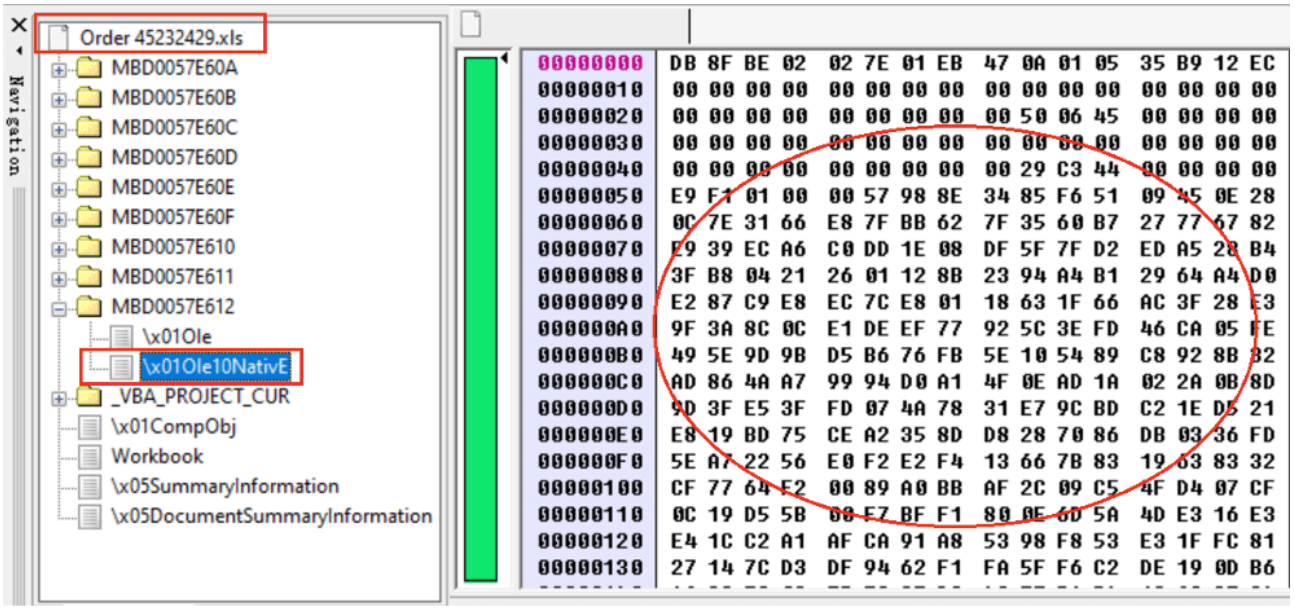


Figure 2.2: The equation content inside the OLE Excel document

Once the crafted Excel document is opened, the malicious equation data is parsed automatically by an MS Office process called “EQNEDT32.EXE.” This triggers the CVE-2017-11882/CVE-2018-0802 vulnerability, and the malicious shellcode inside the equation data is executed in the background.

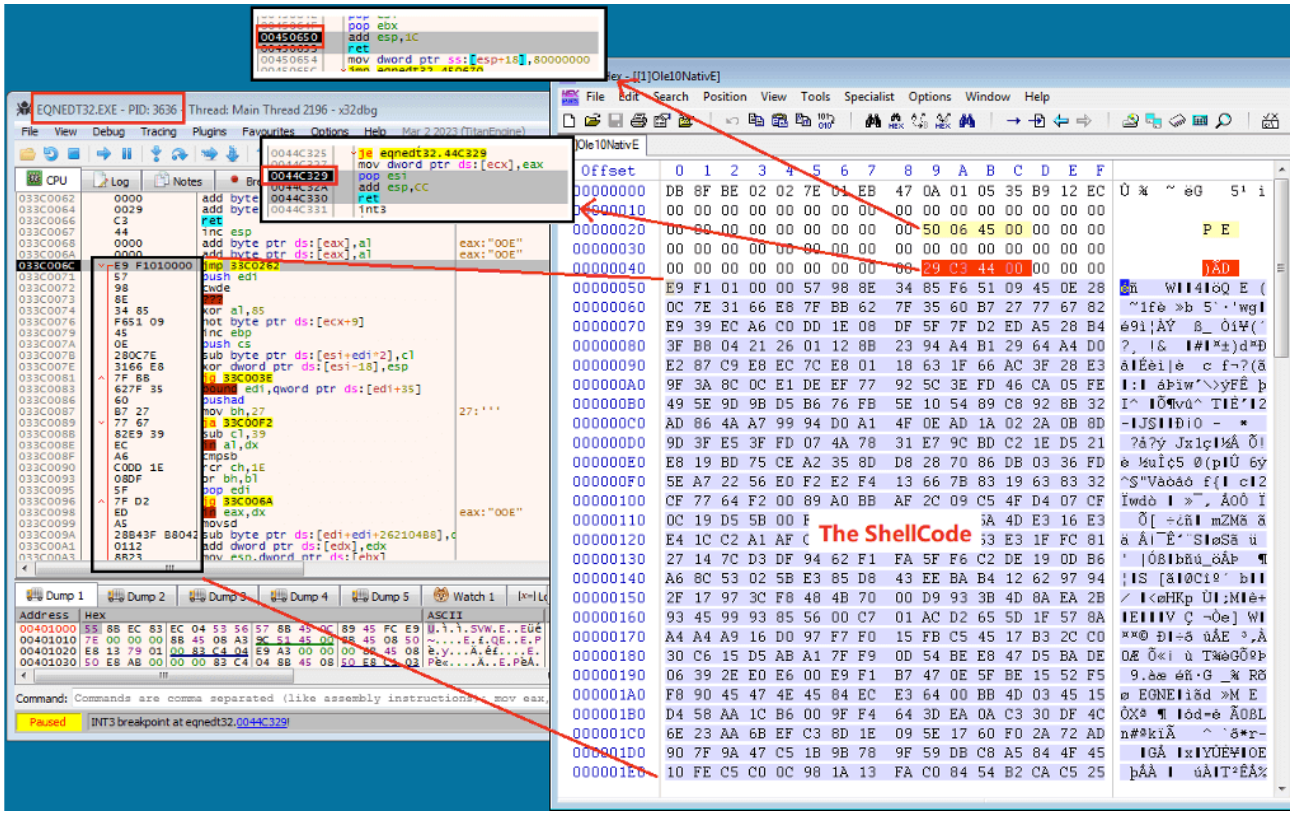


Figure 2.3: The ShellCode about to execute in the vulnerable EQNEDT32.EXE process

In Figure 2.3, we can see that the crafted equation data overrides the stack of the EQNEDT32.EXE and makes it jump twice (through the fixed addresses of 0x450650 and 0x44C329) to the shellcode at 0x33C006C (in the

stack).

After self-decryption, we observe that the shellcode's main job is downloading and executing an additional malware file from the URL "hxxp://23[.]95.128.195/3355/chromium.exe". To do this, it calls several APIs, such as URLDownloadToFileW(), to download the malware to a local folder, and ShellExecuteW() to run the malware on the victim's device. In Figure 2.4, we can see that the shellcode is about to call the API URLDownloadToFileW() to download it into a local file and rename it as "dasHost.exe" under the "%TEMP%" folder.

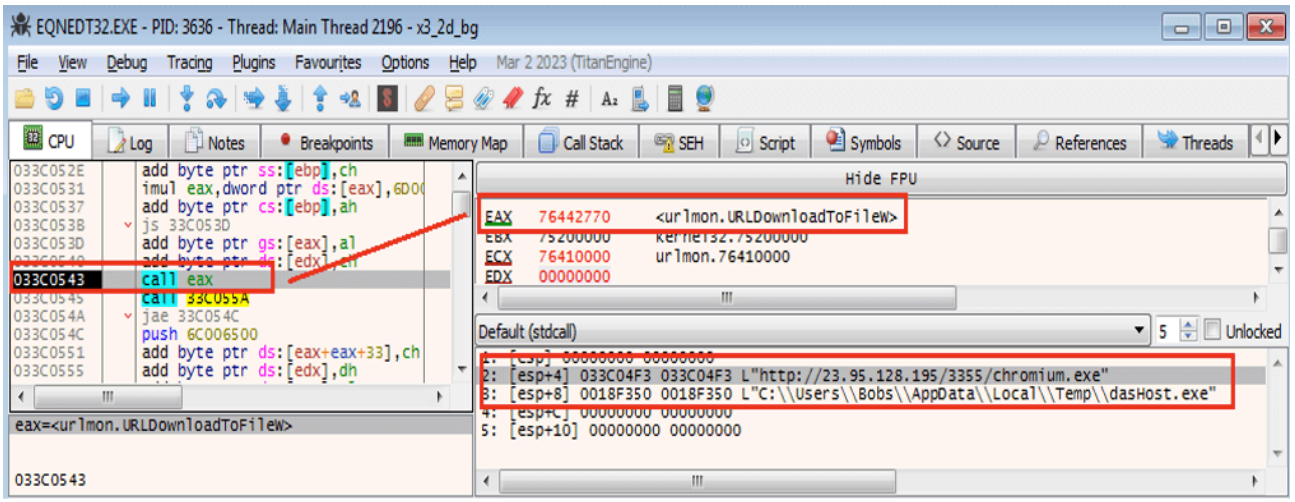


Figure 2.4: Calling the API to download the malware

A Look into the Downloaded File

The downloaded file ("dasHost.exe") is a .Net program protected by two packers, IntelliLock and .NET Reactor.

Figure 3.1 displays the EntryPoint function of the downloaded file inside dnSpy, where the assembly name of the file is "Nvgqn7x". As you may have noticed, all the names of the namespaces, classes, methods, and variables are thoroughly obfuscated.

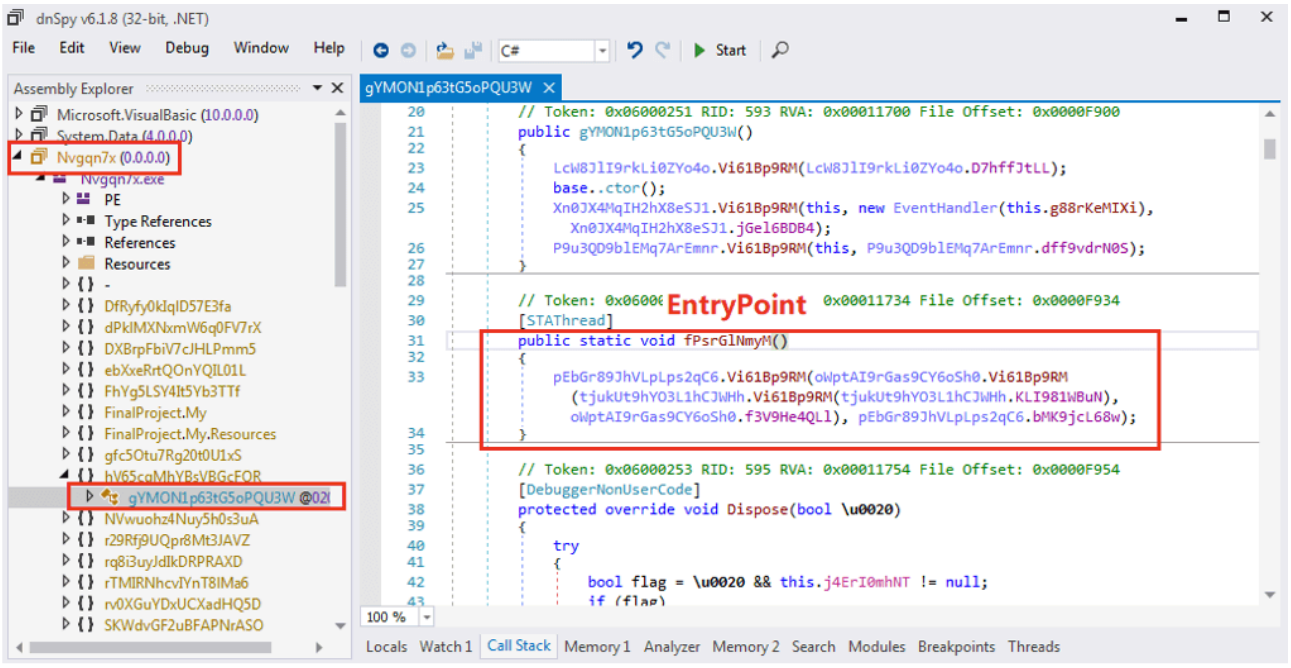


Figure 3.1: The EntryPoint function of the obfuscated downloaded file

There are resource files inside the downloaded file's .Net Resources section. The downloaded file ("dasHost.exe") extracts two fileless execution modules from the .Net Resources section. One is the payload module of Agent Tesla, and the other is a Loader module to the payload file of Agent Tesla.

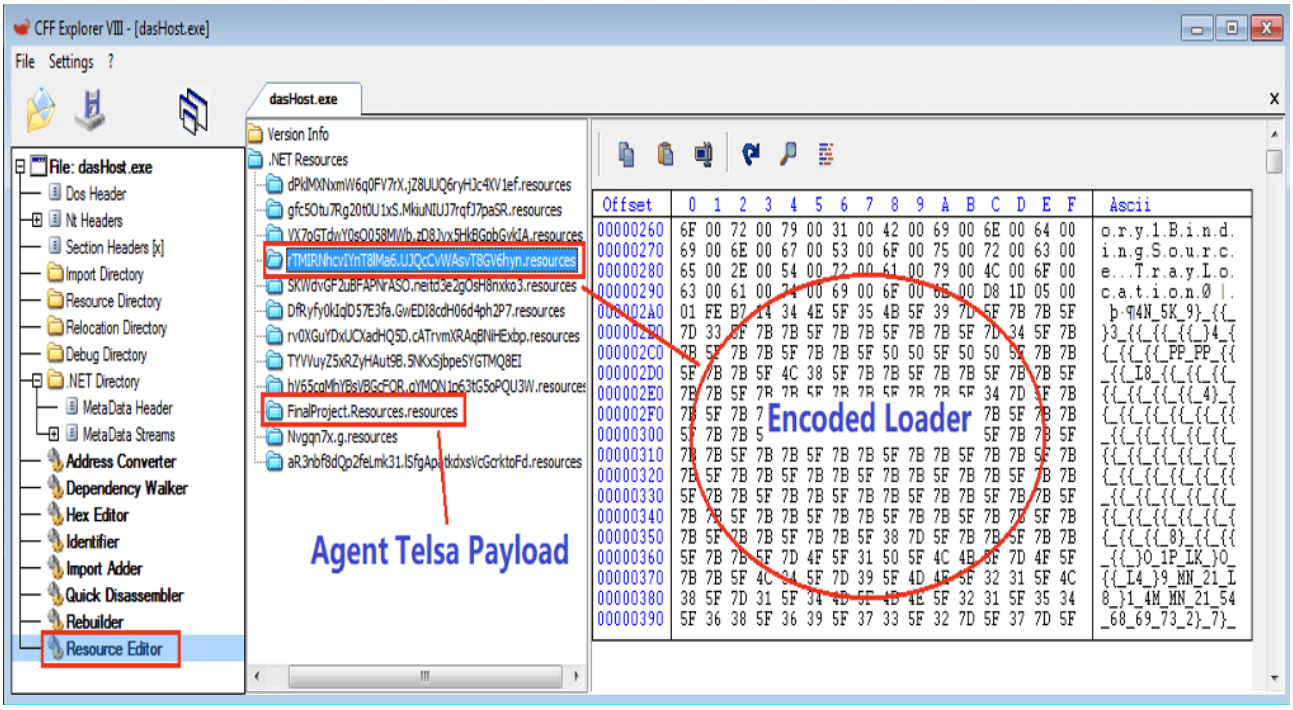


Figure 3.2: The .Net Resources section of the downloaded file

Figure 3.2 shows all the resources in the .Net Resources section. According to my analysis, the resource "rTMIRNhcvIYnT8lMa6.UJQcCvWAsvT8GV6hyn.resources" is the encoded Loader module, whose assembly name is "Cassa." The resource "FinalProject.Resources" is the encrypted and compressed Agent Tesla payload

module, whose assembly name is “NyZELH bX.” It gets decrypted, decompressed, and loaded as a module in the “DeleteMC()” function of the Loader module, as shown in Figure 3.3.

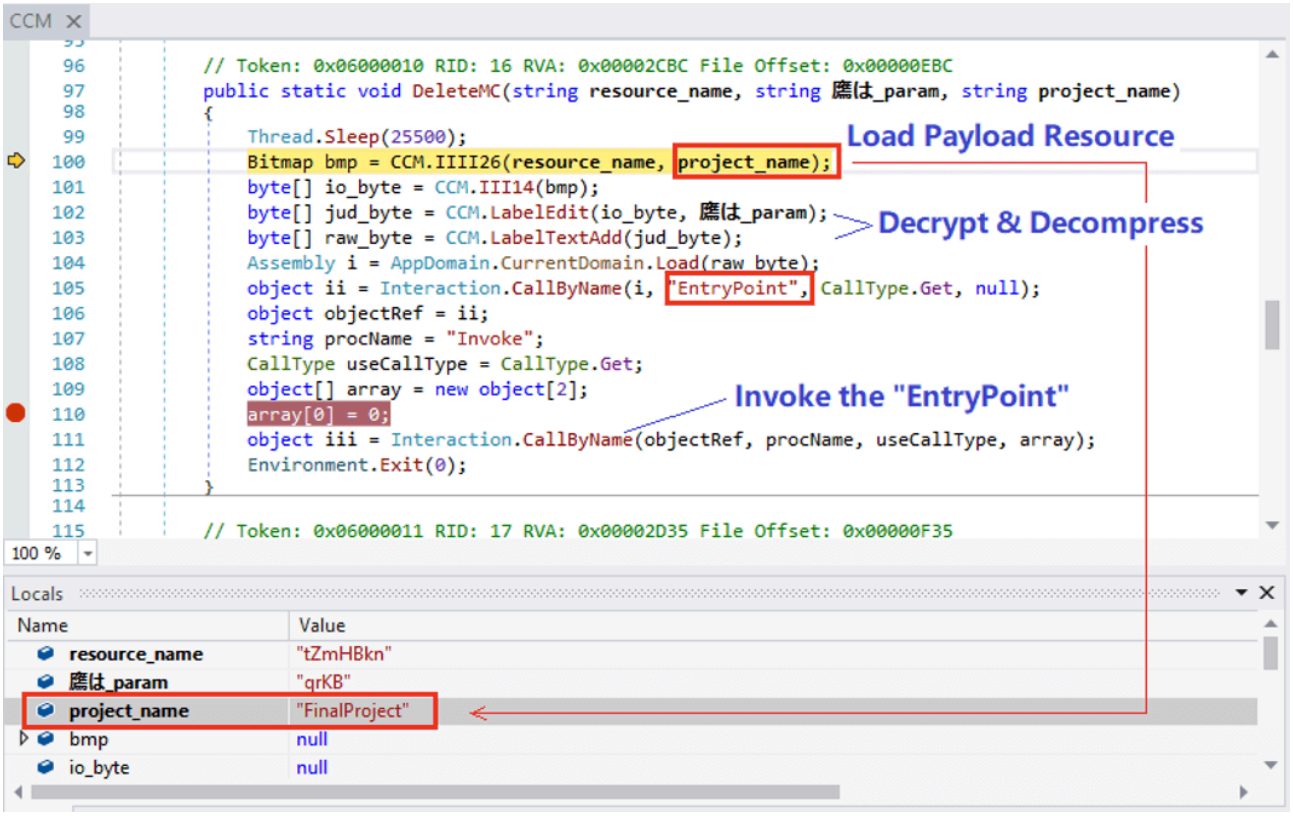


Figure 3.3: Loader “Cassa”‘s DeleteMC() function

As you may have noticed, the resource is disguised as a Bitmap resource and is mixed up with the payload. Bitmap.GetPixel() and Color.FromArgb() are the two APIs being called to read the payload from the resource. It then goes through decryption and gzip decompression to restore the payload file, which is loaded as an executable module by calling the AppDomain.CurrentDomain.Load() method. Finally, the payload file’s “EntryPoint” function is invoked from the Loader module (“Cassa”).

Agent Tesla Payload Module & Process Hollowing

The payload module is a .Net program and is fully obfuscated. Fortunately, I managed to have it de-obfuscated using several analysis tools.

As with most malware, the developers run the malware’s core module in a separate process. This is a common protection strategy to increase the malware's chance of survival on the victim's device.

The primary function (other than persistence) of the payload is to perform the process hollowing and then place another decrypted executable file—sourced from a separate resource (called “7gQsJ0ugxz.resources”) within the payload file—onto the hollowed process and execute it. In this analysis, I call this decrypted executable file the core module of Agent Tesla.

Figure 4.1: APIs to perform Process Hollowing

Figure 4.1 contains the key APIs the payload module invokes to perform the ProcessHollowing. It invokes CreateProcess() to create a suspended process of “dasHost.exe.” Next, it allocates memory in this process via the API VirtualAllocEx() for the core module. Then WriteProcessMemory() is invoked numerous times to copy the core module, saved in the array variable byte_1, onto the new process. It finally calls the APIs SetThreadContext() and ResumeThread() to restore the new process from a suspended state to execute the core module of Agent Tesla.

After that, the payload module exits by calling Environment.Exit() in the Loader module’s DeleteMC() (refer to Figure 3.3).

Persistence

To persist in collecting the victim’s sensitive data, even if the affected system is restarted or the Agent Tesla process is killed, it does the following two things.

1. TaskScheduler

It executes a command to create a task in the system TaskScheuler inside the payload module. The command in my analysis environment is "C:\Windows\System32\schtasks.exe" /Create /TN "Updates\kCqKCO" /XML

"C:\Users\Bobs\AppData\Local\Temp\tmp68E9.tmp," where "Updates\kCqKCO" is the task name and "/XML" specifies it’s been created from an XML file that is provided by following a parameter (i.e. tmp68E9.tmp). Figure 5.1 shows the details of the XML content, where the file “C:\Users\Bobs\AppData\Roaming\kCqKCO.exe” is a duplication of the downloaded “dasHost.exe.” The task is set to start at the victim’s logon.

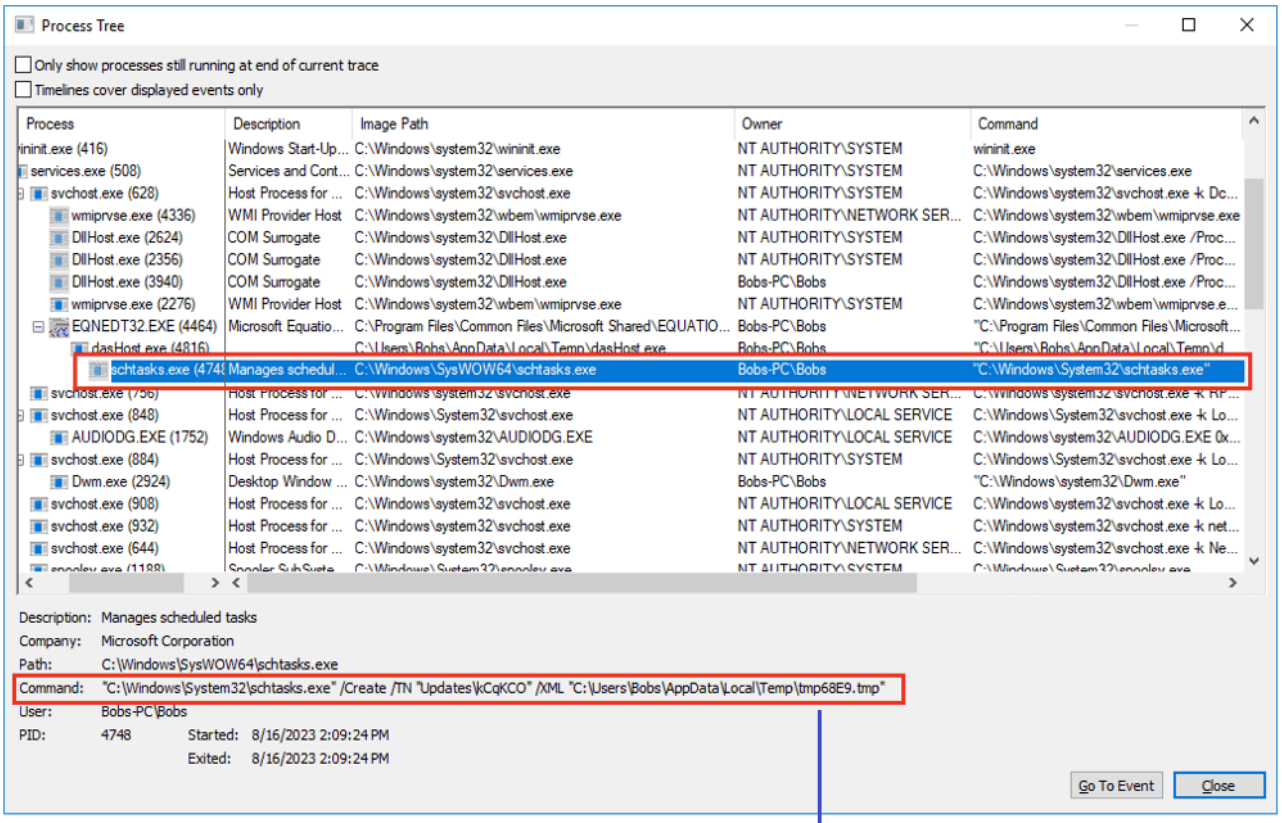


Figure 5.1: Creating a task inside the system TaskScheduler

2. Auto-run in the system registry

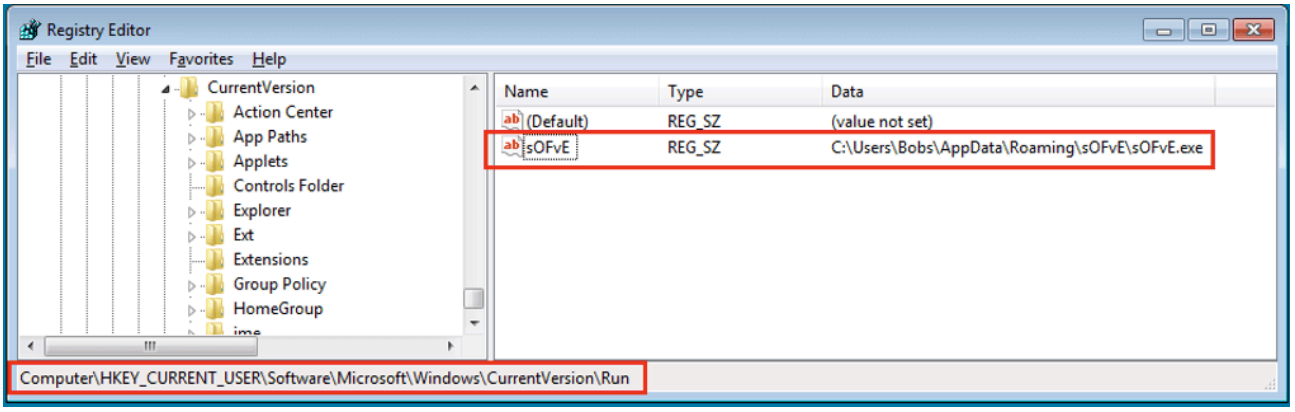


Figure 5.2: Auto-run item in the system registry

The core module adds an auto-run item in the system registry“C:\Users\Bobs\AppData\Roaming\sOFvE\sOFvE.exe” (shown in Figure 5.2). It is another duplication of “dasHost.exe” that is launched automatically at system startup.

Stealing Sensitive Information from the Victim

The Agent Tesla core module collects sensitive information from the victim’s device. This information includes the saved credentials of some software, the victim’s keylogging information, and screenshots of the victim’s device.

Stealing credentials

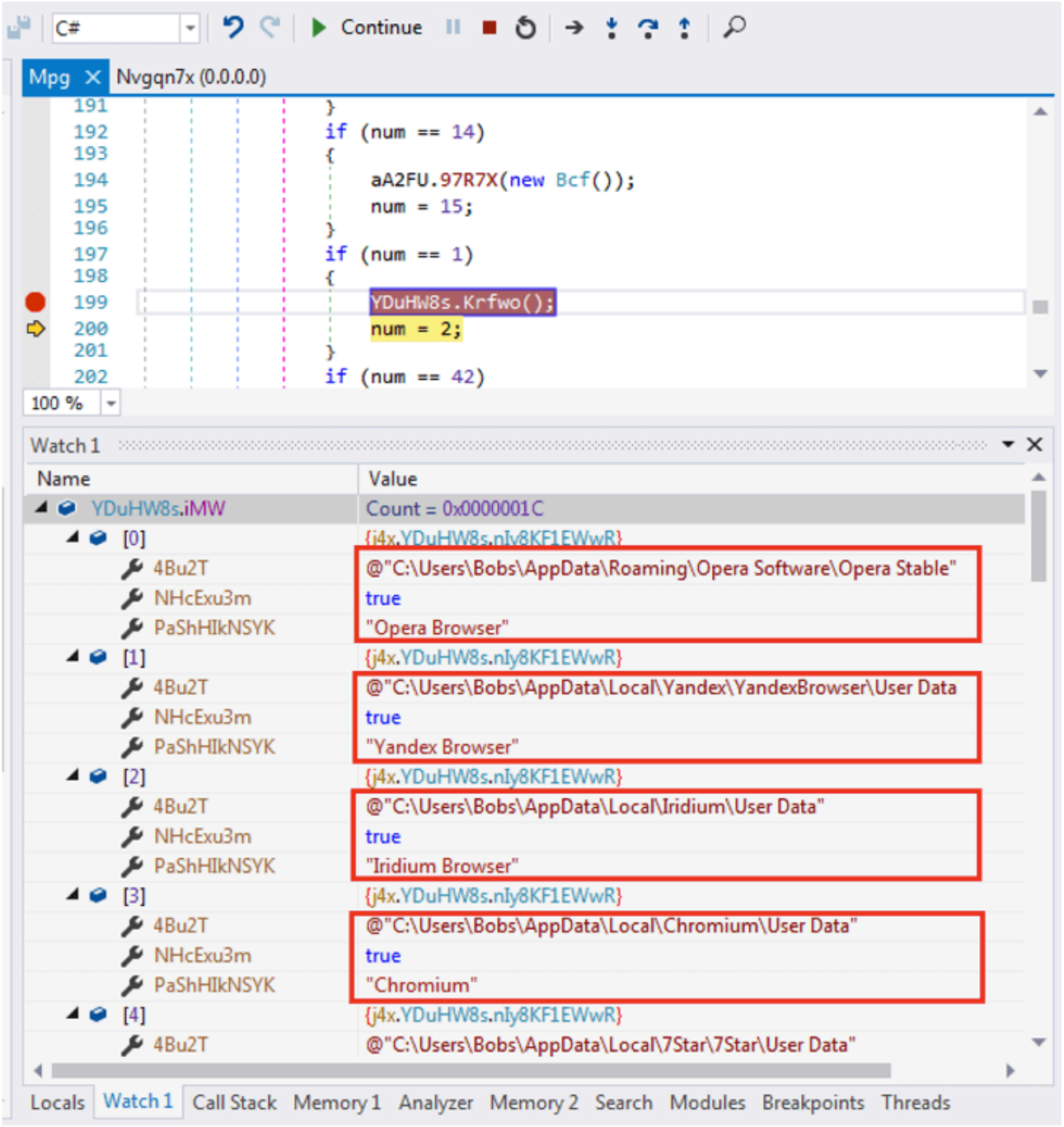


Figure 6.1: Web browser information from which Agent Tesla steals credentials

It steals saved credentials from specified software installed on the victim's device, including web browsers, email clients, FTP clients and more.

Based on their features, the affected software can be categorized as below:

Web Browsers:

"Opera Browser", "Yandex Browser", "Iridium Browser", "Chromium", "7Star", "Torch Browser", "Cool Novo", "Kometa", "Amigo", "Brave", "CentBrowser", "Chedot", "Orbitum", "Sputnik", "Comodo Dragon", "Vivaldi", "Citrio", "360 Browser", "Uran", "Liebao Browser", "Elements Browser", "Epic Privacy", "Coccoc", "Sleipnir 6",

"QIP Surf", "Coowon", "Chrome", "Flock Browser", "QQ Browser", "IE/Edge", "Safari", "UC Browser", "Falkon Browser".

Email clients:

"Outlook", "ClawsMail", "IncrediMail", "FoxMail", "eM Client", "Opera Mail", "PocoMail", "Windows Mail App", "Mailbird", "The Bat!", "Becky!", "Eudora".

FTP clients:

"Flash FXP", "WS_FTP", "FTPGetter", "SmartFTP", "FTP Navigator", "FileZilla", "CoreFTP", "FtpCommander", "WinSCP".

VPN clients:

"NordVPN", "Private Internet Access", "OpenVPN",

IM client:

"Discord", "Trillian", "Psi/Psi+".

Others:

"Mysql Workbench", "\Microsoft\Credentials\", "Internet Download Manager", "JDownloader".

Keylogging

Agent Tesla calls the API SetWindowsHookEx() to set a keyboard hook to monitor low-level input events.

```
Block_4:
try
{
    IL_86:
    string moduleName = Process.GetCurrentProcess().MainModule.ModuleName;
    IntPtr moduleHandle = Y7ALd2ht.GetModuleHandle(moduleName);
    this.qkJOzU8 = Y7ALd2ht.SetWindowsHookEx(13, this.EiqpViCm9, moduleHandle, 0);
    if (this.qkJOzU8 != IntPtr.Zero)
    {
        return;
    }
}
catch
{
}
```

WH_KEYBOARD_LL
13

Hook Procedure

Figure 6.2: Set hook procedure to log keystrokes

In Figure 6.2, the callback hook procedure “this.EiqpViCm9()” is called whenever the victim is typing on their device. Agent Tesla records the program title, time, and victim’s keyboard input contents into a local file “%Temp%/log.tmp” from time to time.

It also has a method called by a Timer every 20 minutes to check the “log.tmp” file and submit its content via SMTP.

Recording screenshots

In the core module, Agent Tesla sets another Timer with a 20-minute interval to call another Timer function. This Timer function checks for any activity on the device and determines whether to record the screenshot and submit it. To do this, it calls the API GetLastInputInfo() to retrieve the time of the last input event received by the system and then compare it with the current time.

The following pseudo-code snippet illustrates how Agent Tesla captures a screenshot.

bitmap = new **Bitmap**(bounds.Width, Screen.PrimaryScreen.Bounds.Height);
encoderParameters = new EncoderParameters(1);
encoder = 25Tl1j7fiou.KIOsJnSv3(**ImageFormat.Jpeg**);
EncoderParameter encoderParameter = new **EncoderParameter**(Encoder.Quality, (long)Convert.ToInt32(60L));
encoderParameters.Param[0] = encoderParameter;
graphics = Graphics.**FromImage**(bitmap);
graphics.**CopyFromScreen**(new Point(0, 0), new Point(0, 0), blockRegionSize);
bitmap.**Save**(memoryStream, encoder, encoderParameters);

The “memoryStream” variable saves the screenshot in jpeg format.

Submitting Sensitive Data via SMTP

Agent Tesla provides multiple ways to submit the stolen data, such as using the HTTP POST method or as the body of an email over SMTP. This variant chooses to submit the data collected from the victim’s device over email SMTP protocol. The SMTP server address and port hardcoded in the variant are "mail.daymon.cc" and 587.

Figure 7.1 shows the malware about to call the smtpClient.Send() function to submit credentials data. The email subject starts with the keyword “PW_” followed by the User name/Computer name for credentials data.

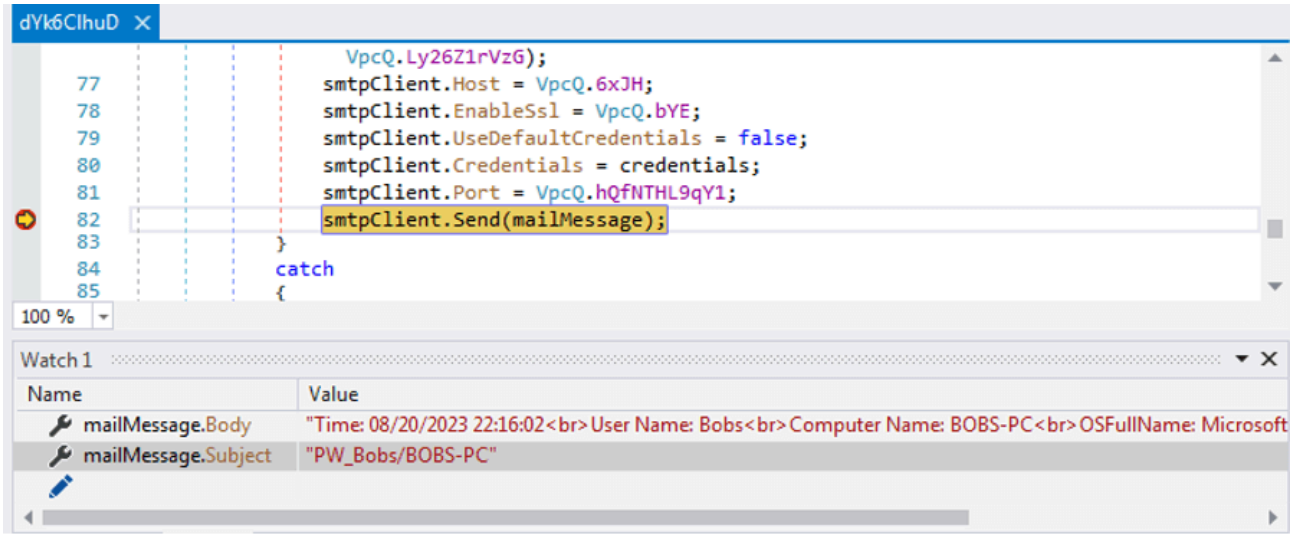


Figure 7.1: Submitting stolen credentials in an email

The email body is formatted in HTML. It is shown in Figure 7.2 when parsing the email body as HTML in a browser.

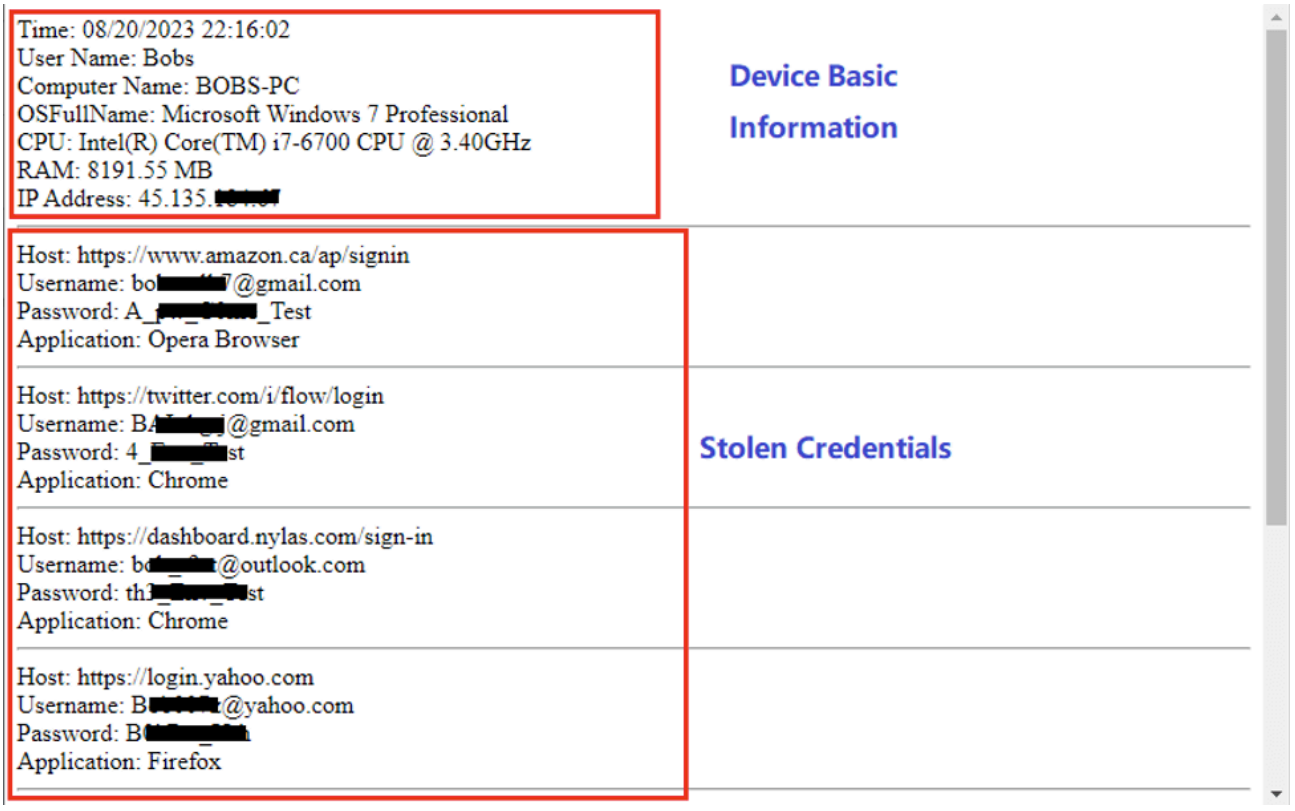


Figure 7.2: Example of stolen credentials

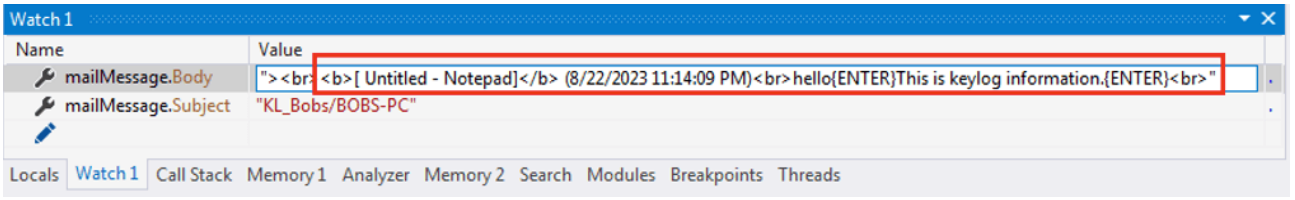


Figure 7.3: Example of the information that the keylogger collected

The email's subject is "KL_{User name/Computer name}," where KL is short for keylogger, and the email body is the collected keylogging data. As displayed in Figure 7.3, the email body includes the records of my keystrokes typed in a Notepad titled "Untitled - Notepad."

The captured screenshot is kept in a variable and added as an email attachment when submitted to the attacker. Figure 7.4 shows it about to add screenshot data to the email as an attachment. The email subject format for ScreenShots is "SC_{User name/Computer name}," and the email body is just the basic information about the victim's device.

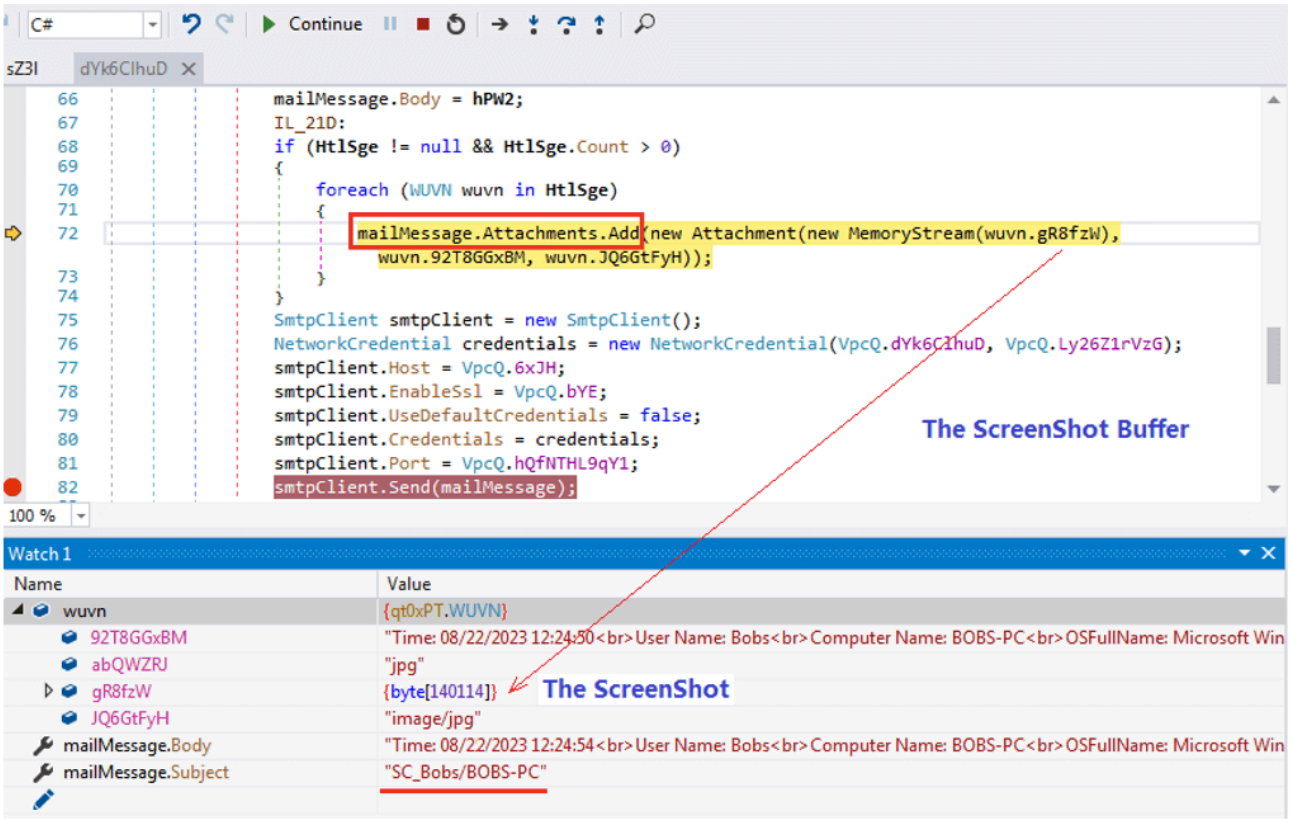
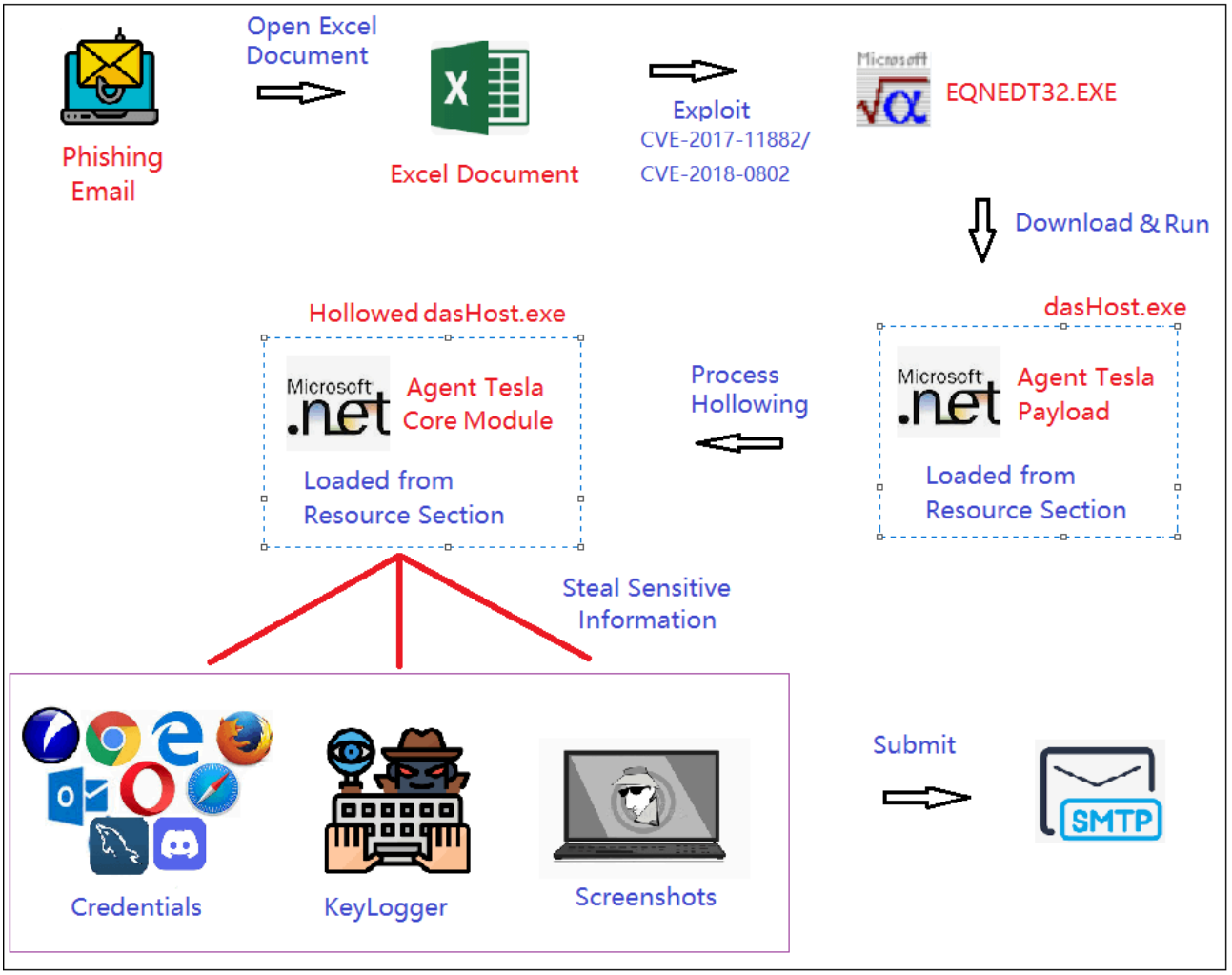


Figure 7.4: Example of submitting victim's screenshot

Summary

The following flowchart roughly describes the outline of the malicious campaign.



This analysis shows that a malicious Excel document attached to a phishing email exploits an aging security vulnerability to execute a shellcode that downloads Agent Tesla. It keeps relevant modules encrypted and encoded in the Resource section to protect its core module from being analyzed.

I then explained how this variant establishes persistence on the victim's device. I also showed the kind of software and data Agent Tesla is able to steal from the infected device, including credentials, keylogging data, and active screenshots.

Lastly, I provided several examples of the kind of sensitive data this variant of Agent Tesla obtained from my analysis environment and how this stolen sensitive data is submitted to the attacker via emails over SMTP protocol.

Fortinet Protections

Fortinet customers are already protected from this campaign with FortiGuard's AntiSPAM, Web Filtering, and AntiVirus services as follows:

The downloading URLs are rated as "**Malicious Websites**" by the FortiGuard Web Filtering service.

FortiMail has recognized the phishing email as SPAM.

FortiGuard Antivirus service detects the attached Excel document and the downloaded file with AV signatures "**MSExcel/CVE_2017_11882.EQMA!exploit**" and "**MSIL/AgentTesla.BEDA!tr**".

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

The FortiGuard CDR (content disarm and reconstruction) service can disarm the malicious Equation data inside the Excel document.

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

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

IOCs

URLs:

Hxxp[:]//23[.]95.128.195/3355/chromium.exe

C2 Server List:

SMTP server @ mail.daymon.cc:587

Relevant Sample SHA-256:

[Order 45232429.xls]
FDC04DC72884F54A4E553B662F1F186697DAF14EF8A2DC367BC584D904C22638

[chromium.exe / dasHost.exe / downloaded file]
36B17C4534E34B6B22728DB194292B504CF492EF8AE91F9DDA7702820EFCFC3A

Source: https://www.fortinet.com/blog/threat-research/agent-tesla-variant-spread-by-crafted-excel-document