Fresh Malware Hunts for Crypto Wallet and Credentials

fortinet.com/blog/threat-research/fresh-malware-hunts-for-crypto-wallet-and-credentials

July 19, 2021

FortiGuard Labs Threat Research Report

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

The FortiGuard Labs team was recently monitoring a new phishing campaign that uses the classic strategy of attaching a malicious Microsoft Word document to an unsolicited email that recipients were then asked to open. However, after I performed some deep research on this phishing campaign, I realized that a fresh malware was being delivered by the Word document designed to steal crypto wallet information and credentials from the victims' infected devices. This malware doesn't seem to belong to any known malware family, so we named it "dmechant", which is a constant string compiled in the malware sample.

In this analysis I reveal my findings on this new malware, including how it is launched by the Word document, how the executable deploys itself on the victim's device, what kind of sensitive information it searches for, and how stolen data is sent to the attacker via SMTP protocol.

The Email Captured by FortiGuard Labs and the Word Document

The spam email looks like an urgent order reminder from a purchase manager. It asks the recipient to review the materials in the attached Word document and then reply to the email as soon as possible.

Figure 1.1 – The captured phishing email

As with Word documents used in other phishing campaigns, this one includes a malicious Macro. Once opened, Word displays a security warning bar to inform the user of the risk of opening it.

Figure 1.2 – The content of the attached Word document

Interestingly, the content of this document is written in Spanish. Our best guesses are that this campaign is targeting multiple regions and the wrong document was attached, or this is being done deliberately to disguise the warning. In either case, I translated it into English using Google translate:

Document created in an earlier version of Microsoft Office Word

To see this content, click on "enable editing" from the yellow bar and then click "Enable content"

The malicious Macro is executed once the button Enable Content is clicked. It has a VBA function called Document_Open(). This is a built-in function of the Macro and is called automatically when the document is opened. It then calls other functions to extract a JS (JavaScript) code into a file (rtbdxsdcb.js) under the %temp% folder. After that, it executes this “JS” file to finish the Macro’s work.

Figure 1.3 is a screenshot of Macro VBA code displaying the JS code to be extracted and the process of writing into local file rtbdxsdcb.js, as well as it being executed.

As you can see, it calls the Shell function to execute the command to run the JS file:

cmd.exe /start %Temp%\rtbdxsdcb.js

Figure 1.3 – JS code, JS file and executing the JS file

Going through the JS code we find that its code is highly obfuscated. According to my analysis, its major task is to request the URL:

hxxps[:]//cdn[.]discordapp[.]com/attachments/789415918744764447/857131714521202688/blessed[.]exe

This downloads an executable file into %Temp% and saves it as erbxcb.exe, which is the payload file of this malware. Later, it creates a WScript.Shell object to start up the payload executable file on the infected device.

Stage I – Starting the Payload Executable File

The payload file, erbxcb.exe, is disguised as a PDF document to confuse the victim. Figure 2.1 shows what this file looks like.

Figure 2.1 – The downloaded executable file – erbxcb.exe

Once it starts, erbxcb.exe loads a block of compressed data from its PE file and decompresses it into a local file named %Temp%\arwtfgxjpx80. (This is an encrypted PE file that I will discuss shortly.) It continues to dynamically extract a piece of code into the Stack memory and get it executed.

The Extracted Code Mainly Performs:

1. It copies erbxcb.exe into a newly-created folder (%AppData%\bplg) and renames “aoqn.exe”. As you can see in Figure 2.2, it was about to call a function (call 12F675, whose arguments are the source file and the target file) to copy this executable file into a new location.

Figure 2.2 – File copy function

2. It adds the copied file into the auto-run group in the system registry. To do this, it calls several APIs, like RegOpenKeyExW() and RegSetValueExW(). Figure 2.3 is a screenshot of the system registry with the added auto-run item, making this malware able to run at Windows startup.

Figure 2.3 – Display of the added auto-run item in system registry

3. It then loads the decompressed file %Temp%\ arwtfgxjpx80, mentioned earlier, into memory and calls a function to decrypt it. As a result, it is able to extract an executable PE file in memory.

The dynamic code then creates a duplicate of the current process (erbxcb.exe) in a suspended state (by calling the API CreateProcessW() with a CREATE_SUSPENDED flag). In this way it is able to deploy the decrypted PE file into the newly-created second process. It then transports each section of the decrypted PE file into the second process to override its existing code. By calling the API SetThreadContext() it is able to set the entry point of the second process to the deployed PE file, and calling ResumeThread() lets the deployed PE file run inside the second process.

The last thing the dynamic code does in the first process is to call the API ExitProcess() to exit the program.

Stage II – Real Malware Runs Inside the Second Process

According to my analysis, the decrypted PE file running inside the second process is written in MS Visual Basic 5.0-6.0 language without any packer. I dumped it into a local file, and Figure 3.1 shows the detection result in an analysis tool—Exeinfo PE.

Figure 3.1 – Decrypted PE file detection result

When it starts, it first gathers basic information about the victim’s device, such as current System Time, Windows Version, User Name, Computer Name, and so on. This information will be sent to the attacker along with other stolen data.

It then sets %AppData%\Microsoft\Windows\Templates as its home folder, where it puts some of its temporary files, such as extracted files, collected browser profiles, and collected crypto wallet data.

Collecting Crypto Wallet Data

This malware focuses on collecting profiles of crypto wallets (if installed) from the infected device. It has ten predefined crypto wallet software instances in the malware with a dynamically combined default profile folder path, as shown at the bottom of Figure 3.2.

This crypto wallet software includes: Zcash, Armory, Bytecoin, Jaxx Liberty, Exodus, Ethereum, Electrum, Atomic, Guarda, and Coinomi.

Figure 3.2 –The full path list of crypto wallets

Once a crypto wallet is detected, it copies the entire folder containing the wallet's profile data (including sub-folders and files—like the path listed in Figure 3.2) to its home folder (at "%AppData%\Microsoft\Windows\Templates"). It then compresses the copied profile folders into a ZIP archive—CryptoWallets.zip—which is eventually sent to the attacker as an email's attachment.

As far as we have been able to determined, the attacker could obtain your bitcoin private key, bitcoin address, crypto wallet address and other significant information.

Collect Saved Credentials by Executing an Extracted EXE file

It also extracts an EXE file ("sdedffggdg.exe") from the Resource section, which is written by .Net Framework. This malware collects credentials from predefined browsers and saves the collected information into a newly-created credentials.txt file under its home folder.

Figure 3.3 is an example of a credentials.txt file, after it executes the extracted .net EXE file("sdedffggdg.exe"), where it shows the credentials collected from the Google Chrome browser from my testing device.

Figure 3.3 – Collected credentials saved in Chrome

"sdedffggdg.exe" is able to collect saved credentials from the following predefined browsers (if installed): "Chrome","Opera","Yandex","360 Browser","Comodo Dragon","CoolNovo","SRWare Iron","Torch Browser","Brave Browser","Iridium Browser","7Star", "Amigo","CentBrowser","Chedot","CocCoc","Elements Browser","Epic Privacy Browser","Kometa","Orbitum","Sputnik","uCozMedia","Vivaldi","Sleipnir 6", "Citrio","Coowon","Liebao Browser","QIP Surf" and "Edge Chromium".

Collect Other Saved Credentials

Other than the browsers identified above, this malware also collects saved credentials from other installed software clients, including Outlook, CoreFTP, FileZilla, NordVPN, FoxMail, Thunderbird, and similar browsers such as Firefox, Waterfox, K-Meleon, Cyberfox, and BlackHawk.

Figure 3.4 – Screenshot of collected credentials in a debugger

In the memory, shown in Figure 3.4, you can see an example of the credentials just after they are collected from the application FileZilla on my research machine. Each group of credentials consists of URL, Username, Password, and Application. Each group of the credentials is split by "==========".

Stage III – Sending Collected Data to the Attacker via SMTP

This malware has a specific function used to generate and send an email that includes the stolen sensitive information using the created CDO.Message object.

Figure 4.1 shows a snippet of this function code (I only included the key code).

Figure 4.1 – Snippet of code generating the exfiltration email

As you can see, it calls rtcCreateObject2() to create the CDO.Message object, which then sets the email's From, To, Subject, TextBody, and AddAttachment fields.

Subject starts with a keyword and follows with the victim's Computer Name and User Name. The keyword string should be one of the "CryptoWallets" for the crypto wallet profile, "Passwords" for credentials, or "Keylogger" for key logger data (this function was not enabled in this version).

After an email is created, the CDO.Message's Send() function is called to send the Email out to the attacker's email address.

Below are two examples of sending these emails via SMTP protocol.

Figure 4.2 sends an email with the crypto wallets profiles in the attachment CryptoWallets.zip to the attacker. The attachment is encoded in base64.

Figure 4.2 – SMTP packet example of sending collected Crypto Wallet profile

Figure 4.3 shows sending stolen credentials, where the text body contains basic information, stolen credentials, and its attachment includes collected browser profiles that contain saved credentials. The attachment is also encoded in base64.

Figure 4.3 – Packet example of sending stolen credentials data

Conclusion

In this post I first detailed the phishing email captured by our FortiGuard Labs and its attached Word document. Next, I demonstrated how a JS file is extracted and executed by the Macro to download the malware's payload executable file.

I also introduced how the malware adds itself into the auto-run group and runs in the second process. I also demonstrated the main tasks of the malware, how and what kind of sensitive information it collects from the infected device, including crypto wallet profile information and the credentials of browsers and clients, and how that collected data is sent to the attacker using the SMTP protocol.

I mentioned many processes in this blog. To help you better comprehend the relationship of the relevant processes initiated by the opening of the Word document, I have made a screenshot of the Process Tree, shown in Figure 5.1. The involved processes are WINWORD.EXE, cmd.exe, WScript.exe, erbxcb.exe, and sdedffggdg.exe.

Figure 5.1 – Process Tree led by opening the Word document

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 downloading URL has been rated as "**Malicious Websites**" by the FortiGuard Web Filtering service.

The Word document and downloaded executable file are detected as "**VBS/Agent.4885!tr**" and "**W32/Kryptik.J!tr**" and are blocked by the FortiGuard AntiVirus service.

FortiMail users are protected by FortiGuard AntiVirus, which detects the original Word document as a malicious attachment in the phishing email, and further protected with the CDR service, which can be used to neutralize the threat of any macros within Office documents.

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

IOCs

URLs:

hxxps[:]//cdn[.]discordapp[.]com/attachments/789415918744764447/857131714521202688/blessed.exe

Sample SHA-256:

[Ncc June Purchase -Contract 0262320216486488574.doc]

105D9496D4F80AE5EF3C7642F55117B65A10398AFE5FF9C30D706FA9873CFD6A

[erbxcb.exe or aoqn.exe]

9C7023EFB920442DFD73D96D303A1B2CFB53A727F2B6B55DE8B4D25BC90FE95E

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