OriginBotnet Spreads via Malicious Word Document

fortinet.com/blog/threat-research/originbotnet-spreads-via-malicious-word-document

September 11, 2023

Affected platforms: Windows
Impacted parties: Any organization
Impact: Remote attackers steal credentials, sensitive information, and cryptocurrency
Severity level: Critical

In August, FortiGuard Labs obtained a Word document containing a malicious URL designed to entice victims to download a malware loader. This loader employs a binary padding evasion strategy that adds null bytes to increase the file's size to 400 MB. The payloads of this loader include OriginBotnet for keylogging and password recovery, RedLine Clipper for cryptocurrency theft, and AgentTesla for harvesting sensitive information. Figure 1 illustrates the comprehensive attack flow.

In this blog, we examine the various stages of how the file is deployed and delve into the specifics of the malware it delivers.

Figure 1: Attack flow

Document Analysis

A phishing email delivers the Word document as an attachment, presenting a deliberately blurred image and a counterfeit reCAPTCHA (Figure 2) to lure the recipient into clicking on it. Clicking activates an embedded malicious link in the file “\word_rels\document.xml.rels,” as shown in Figure 3.

Figure 2: Word document

Figure 3: Malicious URL

Loader Analysis

The initial loader was acquired from https://bankslip[.]info/document/scancop20233108[.]exe. This file, written in .NET, deciphers the “Main_Project” resource data in “HealthInstitutionSimulation.Properties.Resources.resources.” It uses an XOR operation with the string “WdxDFWxcf09WXfVVjLwKKcccwnawf” and then 'Activator.CreateInstance()' to execute the decoded information. The decoding procedure is shown in Figure 4.

Figure 4: Decoding resource data in “scancop20233108.exe”

The second stage uses the “Main Project.dll” with the entry point illustrated in Figure 5. In this stage, the code initiates a “Sleep()” function within “Delation()” and establishes persistence through the “Moschop()” function.

Figure 5: Entry point of “Main Project.dll”

It then loads Base64-encoded strings and uses the AES-CBC algorithm for decryption, retrieving a PowerShell command, as shown in Figure 6. To ensure persistence, it duplicates the EXE file into the directory “%AppData%\Microsoft\Windows\Start Menu\Programs\Startup” under the filename “audacity.exe.exe” to ensure that the file runs automatically even if the victim restarts their device.

Figure 6: PowerShell command for persistence in “Main Project.dll”

Following that, it employs the command “GetType(‘I.L’).GetMethod(‘U’)” to invoke a method from the DLL that was decrypted from the resources labeled “DataPresent.” This is passed to the third-stage payload, decrypted from the data within the resources labeled “Moss,” using the AES-ECB algorithm, as shown in Figure 7.

Figure 7: Load decrypted payload in “Main Project.dll”

The third stage uses “scancopper4647979413.exe,” which is another .NET executable file. It utilizes the “Activator.CreateInstance()” method to generate an instance decoded from the resources, “rumdisintegration.dat,” effectively triggering the execution of the fourth-stage file, “cargomind.dll.” It then uses the “CreateInstance()” method with two parameters: the object type for instantiation and an array of arguments to be transmitted to the created object.

Figure 8: The entry point of “scancopper4647979413.exe”
The fourth stage is represented by a DLL file, “cargomind.dll.” Its entry point is shown in Figure 9. It comprises three Base64-encoded strings intended for subsequent operations. The “Deserialize()” function, as shown in Figure 10, is responsible for decoding these strings, parsing the key-value pairs for each option, and ultimately returning a dictionary.

Figure 9: The entry point of “cargomind.dll”

Figure 10: Function for parsing data
Figure 11 displays the result obtained from “list2.” It reveals the existence of three tasks, each comprising six distinct options.

Figure 11: The tasks in “cargomind.dll”
Let's explore the options within “list2[0]” in detail:

1. “u”: URL, which is specified as https://softwarez[.]online/javau[.]exe.
2. “k”: Action, with “d” indicating a download action, as shown in Figure 12.
3. “df”: File directory, where “ad” designates the ApplicationData folder (%appdata%), with the associated function being “ConstructPath(),” as shown in Figure 13.
4. “sf”: Subfolder, denoted as “Java.”
5. “fn”: File name, identified as “javau.exe.”
6. “e”: Execution status, where “y” signifies “yes” and triggers the execution of the downloaded file using “Process.Start.”

Figure 12: Function for option “k”

Figure 13: Function for constructing file path
For the remaining two tasks in “list2,” the action is set to “b.” Consequently, it invokes the “ExecuteBinder()” function to decode data specified in the “r_k” option, as shown in Figure 14. The targeted files in this context are “newcrisp.dat” and “backyard.dat,” both sourced from the resources section of the prior stage, “scancopper46477979413.exe,” as shown in Figure 15.

Figure 14: Function for decoding payload

Figure 15: Resources data in “scancopper46477979413.exe”

Malware Analysis – RedLine Clipper

The initial malware originates from the URL https://softwarez[.]online/javau[.]exe. It is a .NET executable file that has been packed using SmartAssembly. Upon deciphering the resource data, we uncovered the ultimate payload, “RedLine Clipper,” as shown in Figure 16.

Figure 16: Decoded data in “javau.exe”
RedLine Clipper (SHA256: 4617631b4497eddcbd97538f6712e06fabdb53af3181d6c1801247338bffaad3), also known as ClipBanker, specializes in stealing cryptocurrencies by manipulating the user's system clipboard activities to substitute the destination wallet address with one belonging to the attacker. The compromised version (Figure 17) supports cryptocurrencies, including Bitcoin, Ethereum, Dogecoin, Litecoin, Dashcoin, and Monero. It continually monitors the clipboard for a copied coin wallet address, which is typically lengthy and complex, making manual entry impractical. When a wallet address is detected on the clipboard, RedLine Clipper covertly alters it to match the attacker's wallet address.

Ordinarily, cryptocurrency wallet addresses adhere to specific formats, but due to their complexity, users often copy and paste them during transactions. Consequently, if the wallet address is tampered with at this stage, users intending to send funds to a particular wallet may inadvertently deposit them into the attacker's wallet instead.

To carry out this operation, RedLine Clipper utilizes the “OnClipboardChangeEventHandler” to regularly monitor clipboard changes and verify if the copied string conforms to the regular expression depicted in Figure 18. It's worth noting that the attacker targets all six supported cryptocurrencies in this scheme.

Figure 17: Redline Clipper Cracked

Figure 18: Run() function for RedLine Clipper

Malware Analysis – Agent Tesla

The second file, an Agent Tesla variant, is stored as “COPPER.exe” (SHA256: c241e3b5d389b227484a8baec303e6c3e262d7f7bf7909e36e312dea9fb82798). This malware can log keystrokes, access the host's clipboard, and conduct disk scans to uncover credentials and other valuable data. Further, it can transmit gathered information to its Command and Control (C2) server through various communication channels, including HTTP(S), SMTP, FTP, or even dispatching it to a designated Telegram channel.

To ensure its persistence, the malware replicates itself to the location “%AppData%\EbJgI\EbJgI.exe” and establishes itself as an auto-run entry within the system registry, as shown in Figure 20. Additionally, it compiles a list of specific software installed on the victim's device, including web browsers, email clients, FTP clients, and more, as shown in Figure 21.

Figure 19: File copy in Agent Tesla

Figure 20: Registry setting in Agent Tesla

Figure 21: Partial list of targeted software

This specific version of Agent Tesla employs SMTP as its C2 connection protocol. You can see the details of the traffic session in Figure 22.

Figure 22: C2 connection of Agent Tesla

Malware Analysis – OriginBotnet

The third file, OriginBotnet, is stored as “david.exe” (SHA256: be915d601276635bf4e77ce6b84feeec254a900c0d0c229b0d00f2c0bca1bec7). It is named after its namespace, as seen in Figure 23. OriginBotnet has a range of capabilities, including collecting sensitive data, establishing communications with its C2 server, and downloading additional files from the server to execute keylogging or password recovery functions on compromised devices.

Figure 23: Entry point of OriginBotnet

Initially, OriginBotnet scans running processes to determine if it is already active within the environment.

Figure 24: Checking process

It then initializes its settings and gathers essential information about the victim’s device, such as the installed AntiVirus Product, CPU, GPU, country, OS name, and username, as shown in Figure 25. Once the system information has been collected, the malware connects with the C2 server at https://nitrosoftwares[.]shop/gate.

Figure 25: Settings for OriginBotnet

Figure 26 shows the function responsible for transmitting messages. The communication is conducted via a POST request using a parameter named “p.” The POST data is subjected to TripleDES encryption (in ECB mode, with PKCS7 padding) and subsequently encoded in Base64 format. The encryption key for TripleDES is stored within the “x-key” field of the HTTP Header. Additionally, the Content-Type and User-Agent values are hard-coded as “application/x-www-form-urlencoded” and “Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:99.0) Gecko/20100101 Firefox/99.0,” respectively. Figures 27 and 28 provide insights into the traffic capture and decrypted message.

Figure 26: Function for sending a message to the C2 server

Figure 27: C2 connection of OriginBotnet

Figure 28: Decrypted message

After receiving an “OK” signal from the C2 server, OriginBotnet enters a waiting state and proceeds to parse incoming C2 commands. The process for handling these commands is outlined in Figure 28. The available commands include “downloadexecute,” “uninstall,” “update,” and “load.”

Figure 29: Function for handling C2 command
If the victim receives either the “downloadexecute” or “update” command, the malware proceeds to parse additional parameters, including the URL. It then directly downloads supplementary files from the specified URL and executes them. It selects the appropriate execution method depending on the file’s extension (.exe, .msi, or .java). This may involve using “Process.Start” or invoking commands such as “msiexec.exe /I” or “java.exe -jar,” as shown in Figure 30.

When receiving an “uninstall” command, OriginBotnet invokes “MoveFile” to relocate the file to a temporary folder.

Figure 30: Function for downloading and execution
The final command, “load,” retrieves plugins from the C2 server. The POST session and the decoded data for this specific request are displayed in Figure 31. In this context, two plugins are available for OriginBotnet: Keylogger and PasswordRecovery. The plugin DLL file is transmitted as a Base64 encoded string within the “bytes” parameter. The processing function for this operation is shown in Figure 32.

Figure 31: Message and decoded data of requesting a plugin

Figure 32: Function for processing plugin
The Keylogger plugin (SHA256: c204f07873fafdfd48f37e7e659e3be1e4202c8f62db8c00866c8af40a9a82c5) is designed to covertly record and log each keystroke executed on a computer as well as monitor user activities. It employs techniques such as “SetWindowsHookEx” for capturing keyboard input events and “GetForegroundWindow” to determine the active window the user is working in. It also keeps tabs on clipboard text content through “SetClipboardViewer.” The stolen text file uses a format similar to Agent Tesla's, as shown in Figure 35.

Figure 33: API for starting the hook of the keyboard

Figure 34: Get foreground window

Figure 35: Log format for copied text
The PasswordRecovery plugin (SHA256: 56ced4e1abca685a871b77fab998766cbddfb3edf719311316082b6e05986d67) retrieves and organizes the credentials of various browser and software accounts. It records these results and reports them via HTTP POST requests. Its primary function is shown in Figure 36. The plugin is designed to target the following browsers and software applications:

- Chromium Browsers: Opera, Yandex, Iridium, Chromium, 7Star, Torch, Cool Novo, Kometa, Amigo, Brave, CentBrowser, Chedot, Orbitum, Sputnik, Comodo Dragon, Vivaldi, Citrio, 360 Browser, Uran, Liebao, Elements, Epic Privacy, Coccoc, Sleipnir 6, QIP Surf, Coowon, Chrome, and Edge Chromium
- Other Browsers: Firefox, SeaMonkey, Thunderbird, BlackHawk, CyberFox, K-Meleon, IceCat, PaleMoon, IceDragon, Waterfox, Postbox, Flock, IE, UC, Safari for Windows, QQ Browser, and Falkon Browser
- Email & FTP Clients: Outlook, Windows Mail App, The Bat!, Becky!, IncrediMail, Eudora, ClawsMail, FoxMail, Opera Mail, PocoMail, eM Client, Mailbird, FileZilla, WinSCP, CoreFTP, Flash FXP, FTP Navigator, SmartFTP, WS_FTP, FtpCommander, FTPGetter
- Others: DynDns, OpenVPN, NordVpn, Private Internet Access, Discord, Paltalk, Pidgin, Trillian, Psi/Psi+, MySQL Workbench, Internet Downloader Manager, JDownloader 2.0, \Microsoft\Credentials\, RealVNC, TightVNC

Figure 36: The main function for PasswordRecovery

Conclusion

This cyberattack campaign uncovered by FortiGuard Labs involved a complex chain of events. It began with a malicious Word document distributed via phishing emails, leading victims to download a loader that executed a series of malware payloads. These payloads included RedLine Clipper, Agent Tesla, and OriginBotnet. The attack demonstrated sophisticated techniques to evade detection and maintain persistence on compromised systems. We also provided a comprehensive breakdown of each attack stage, shedding light on the intricacies of the deployed malware and the tactics employed.

Fortinet Protections

The malware described in this report are detected and blocked by FortiGuard Antivirus as:

MSOffice/Agent.DA32!tr.dldr
MSIL/Agent.8DF3!tr
MSIL/Agent.DGH!tr
MSIL/Agent.F!tr.spy
MSIL/Agent.CSS!tr.spy
MSIL/Kryptik.AHUA!tr
MSIL/Kryptik.PSV!tr
MSIL/Injector.WGW!tr
MSIL/Injector.WHL!tr
MSIL/ClipBanker.PK!tr
MSIL/Keylogger.ELM!tr
MSIL/OriginBotnet.G!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 URLs are rated as “Malicious Websites” by the FortiGuard Web Filtering service.

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:

bankslip[.]info
softwarez[.]online
nitrosoftwares[.]shop

Files:

c9e72e2865517e8838dbad0ce41561b2bd75c399b7599c1711350f9408189b9b
56ced4e1abca685a871b77fab998766cbddfb3edf719311316082b6e05986d67
c204f07873fafdfd48f37e7e659e3be1e4202c8f62db8c00866c8af40a9a82c5
21ad235118c371e2850c539040b6dcdd88196c021245440155fe80aacf6ccc7e
4617631b4497eddcbd97538f6712e06fabdb53af3181d6c1801247338bffaad3
be915d601276635bf4e77ce6b84feeec254a900c0d0c229b0d00f2c0bca1bec7
c241e3b5d389b227484a8baec303e6c3e262d7f7bf7909e36e312dea9fb82798
dfd2b218387910b4aab6e5ee431acab864b255832eddd0fc7780db9d5844520a
f36464557efef14b7ee4cebadcc0e45af46f5c06b67c5351da15391b03a19c4c
b15055e75ae0eeb4585f9323ef041fa25ed9b6bf2896b6ea45d871d49a1c72b8
49c969a5461b2919fd9a7dc7f76dd84101b2acc429b341f8eeee248998e9da32
65e47578274d16be1be0f50767bad0af16930df43556dd23d7ad5e4adc2bcbe3