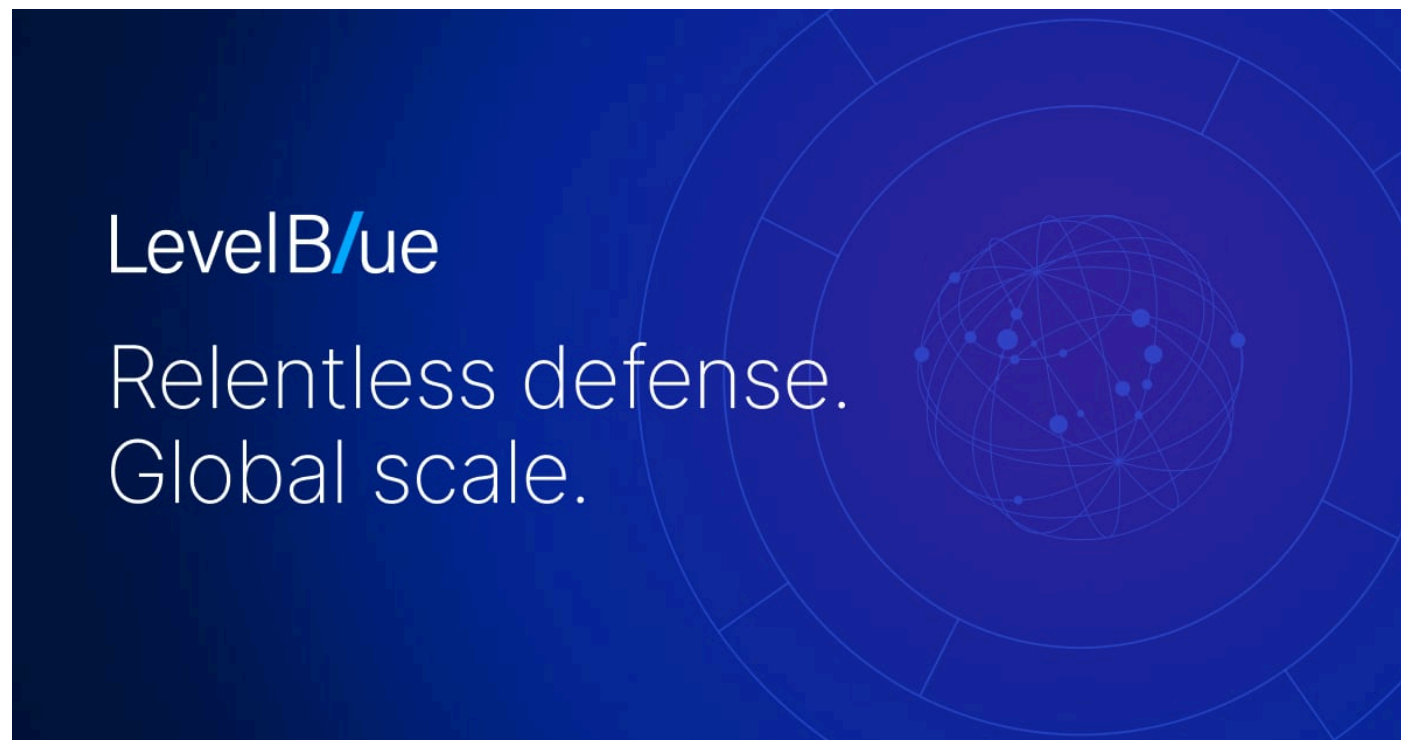


Unmasking a Multi-Stage Loader: Autolt Abuse Leading to Vidar Stealer Command-and-Control Communication

 levelblue.com/blogs/spiderlabs-blog/unmasking-a-multi-stage-loader-autoit-abuse-leading-to-vidar-stealer-command-and-control-communication

Mahadev Joshi

May 7, 2026



- [Home](#)
- [SpiderLabs Blog](#)

LevelBlue's Security Services issues Threat Analysis reports to inform on impacting threats. The Threat Analysis reports investigate these threats and provide practical recommendations for protecting against them.

In this Threat Analysis report, we investigate a multi-stage malware execution chain identified through proactive [threat hunting](#) activities within a client environment. LevelBlue's Endpoint telemetry and dynamic analysis revealed a sequence of processes demonstrating script masquerading, defense evasion, staged payload extraction, and command-and-control (C2) communication.

Key points

- The infection chain began with the execution of *MicrosoftToolkit.exe*, a commonly abused tool, which spawned a command shell to initiate the next stage of the attack.
- A disguised file (*Swingers.dot*) was renamed to a *.bat* script and executed, demonstrating file extension masquerading to bypass security controls.
- The attack included process discovery and termination attempts (*tasklist*, *findstr*), followed by payload extraction using *extract32.exe* and execution of an AutoIt loader (*Replies.scr*) with an external parameter.

- Outbound network connections from the Autolt component to infrastructure associated with the *Vidar Stealer* were observed, indicating deployment of an information-stealing payload.

Introduction

Autolt is a legitimate Windows scripting language designed to automate tasks such as system administration and application interaction. While it is widely used for benign automation, threat actors frequently abuse Autolt to create compiled scripts that can execute malicious code, load encrypted payloads, and evade traditional security detection. Because Autolt executables resemble normal applications, they are often used as stealthy loaders in multi-stage attacks.

In this incident, an Autolt-compiled script (*Replies.scr*) was used as a loader component within the attack chain. The script processed an external payload file and subsequently initiated network communication.

Further analysis confirmed communication with infrastructure associated with *Vidar*, a well-known information-stealing malware capable of harvesting credentials, browser data, cryptocurrency wallets, and system information.

Vidar is a prominent information-stealing malware that first appeared in late 2018 and was developed using the source code of the *Arkei stealer*.

The observed behavior indicates that the Autolt component functioned as a delivery mechanism for the Vidar stealer.



Figure 1. Vidar infection chain.

Attack Flow Breakdown

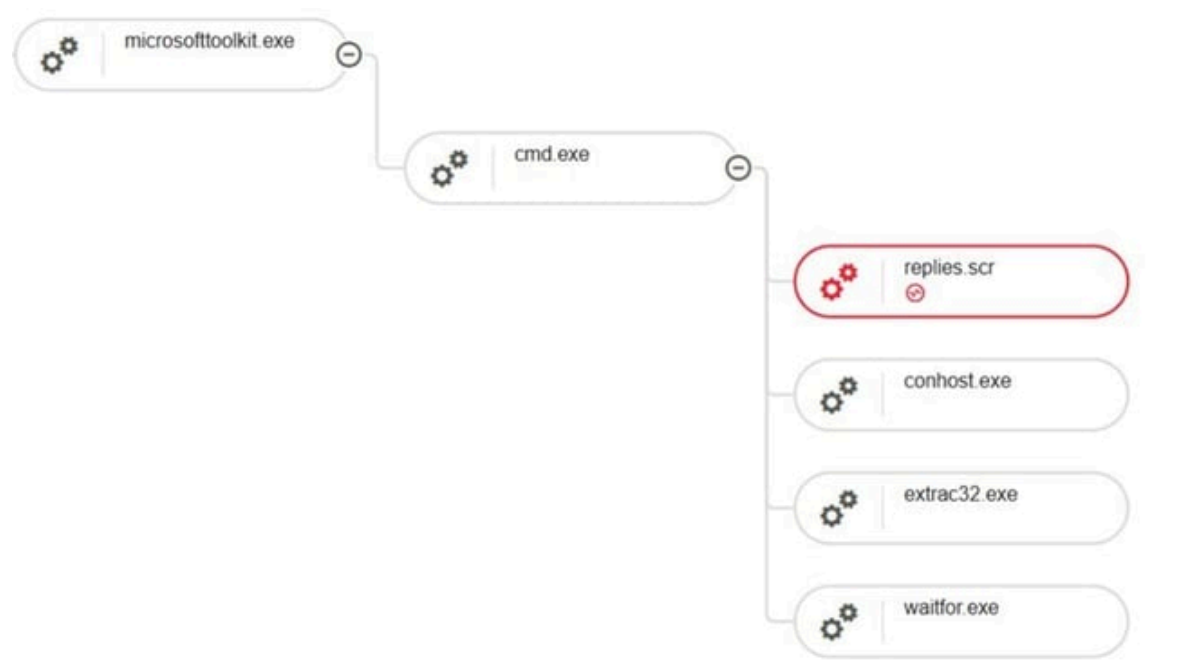


Figure 2. Attack tree observed in LevelBlue's EDR.



Figure 3. Another instance of Replies.scr making a C2 connection.

Attack Overview

The attack chain begins with the execution of Microsoft Toolkit, a commonly abused hack tool. This initiates a sequence of commands that rename and execute a disguised script file.

The script performs environment checks and attempts to identify security-related processes before extracting additional payload components. An AutoIt-compiled executable is then executed with an external payload file, indicating a staged loader architecture.

Subsequent activity reveals outbound communication to infrastructure associated with Vidar, confirming the deployment of an information-stealing payload.

Attack Type: Loader / Dropper infection chain

Technical analysis

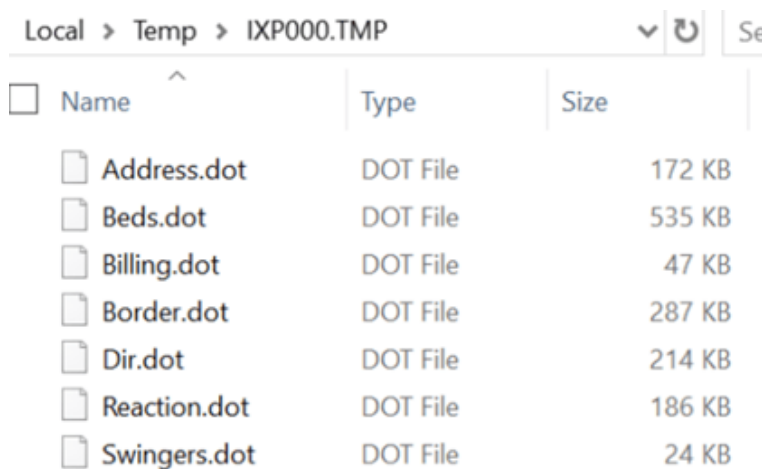
LevelBlue's EDR telemetry and dynamic analysis revealed the following sequence:

- Execution of MicrosoftToolkit.exe (hack tool)
- Spawn of cmd.exe
- Renaming and execution of swingers.dot as swingers.dot.bat
- Process discovery (tasklist, findstr)
- Payload extraction via extract32.exe
- Execution of replies.scr with parameter D
- Execution delay/Synchronization via waitfor.exe
- Outbound C2 communication from a secondary instance of replies.scr consistent with Vidar infrastructure

Execution and Initial Staging

The use of a hack tool as the initial execution vector suggests user-driven execution, likely under the assumption of legitimate software activation. This technique reduces the need for traditional delivery mechanisms such as phishing or exploitation.

The malware uses *.dot* files as a payload container format, likely hiding encoded or segmented data inside multiple files.



The screenshot shows a Windows File Explorer window with the address bar set to 'Local > Temp > IXP000.TMP'. The window displays a list of files with columns for Name, Type, and Size. There are seven files listed, all of which are '.dot' files. The files are: Address.dot (172 KB), Beds.dot (535 KB), Billing.dot (47 KB), Border.dot (287 KB), Dir.dot (214 KB), Reaction.dot (186 KB), and Swingers.dot (24 KB). Each file has a small icon to its left representing a document.

Name	Type	Size
Address.dot	DOT File	172 KB
Beds.dot	DOT File	535 KB
Billing.dot	DOT File	47 KB
Border.dot	DOT File	287 KB
Dir.dot	DOT File	214 KB
Reaction.dot	DOT File	186 KB
Swingers.dot	DOT File	24 KB

Figure 4. Snippet of files created by Microsofttool.exe.

Masquerading and Script Execution

Renaming *.dot* file to *.bat* file enabled the execution of embedded commands while evading simplistic file-type controls. This reflects a reliance on user-space scripting rather than exploit-based delivery.

0000005D52EEE150	000001B4EF564830	L"C:\\WINDOWS\\SYSTEM32\\cmd.exe"
0000005D52EEE158	0000000000000000	
0000005D52EEE160	0000000000000000	
0000005D52EEE168	0000000000000000	
0000005D52EEE170	000001B4EF565580	L"cmd /c move Swingers.dot Swingers.dot.bat && Swingers.dot.bat"
0000005D52EEE178	0000866400000230	
0000005D52FFF180	0000000000008664	

Figure 5. File-based staging and renaming.

Defense Evasion

The invocation of *tasklist.exe* and *findstr.exe* indicates active enumeration and potential disruption of security tooling. This behavior is consistent with attempts to weaken endpoint visibility prior to payload deployment.

Payload Staging

The subsequent execution of *replies.scr* (Autolt-compiled binary) with an external argument (D) demonstrates a staged loader architecture, where the Autolt binary functions as a loader rather than the final payload.

Autolt provides a flexible and low-friction mechanism for executing malicious logic while blending in with legitimate administrative tooling.

Payload Reconstruction

Data is progressively:

- Extracted from *.dot* files
- Filtered via *findstr*
- Written into *Replies.scr*
- Executed with argument D

This indicates a builder-style loader mechanism.

This matches typical loader behavior: the initial *.scr* file (*Replies.scr*) interacts with an encrypted payload (D) and then reads it into memory for decryption and execution.

1:35:57.5863915 AM	Replies.scr	51... CreateFile	C:\Users\████\AppData\Local\Temp\IXP000.TMP\584145\ID	SUCCESS
1:35:57.5864147 AM	Replies.scr	51... QueryBasicInformat...	C:\Users\████\AppData\Local\Temp\IXP000.TMP\584145\ID	SUCCESS
1:35:57.5864296 AM	Replies.scr	51... CloseFile	C:\Users\████\AppData\Local\Temp\IXP000.TMP\584145\ID	SUCCESS

Figure 6. *Replies.scr* interacting with file D.

1:35:57.5891797 AM	Replies.scr	51... ReadFile	C:\Users\████\AppData\Local\Temp\IXP000.TMP\584145\ID	SUCCESS
1:35:57.5892331 AM	Replies.scr	51... ReadFile	C:\Users\████\AppData\Local\Temp\IXP000.TMP\584145\ID	SUCCESS
1:35:57.5894782 AM	Replies.scr	51... ReadFile	C:\Users\████\AppData\Local\Temp\IXP000.TMP\584145\ID	SUCCESS
1:35:57.5896715 AM	Replies.scr	51... ReadFile	C:\Users\████\AppData\Local\Temp\IXP000.TMP\584145\ID	SUCCESS

Figure 7. *Replies.scr* reading from file D.



Figure 8. Binary file D loaded by Replies.scr.

Dedicated to hunting and eradicating the world's most challenging threats.

[SpiderLabs](#)

Vidar Analysis

As soon as we load and run the extracted executable payload into x64dbg, we get the following message.

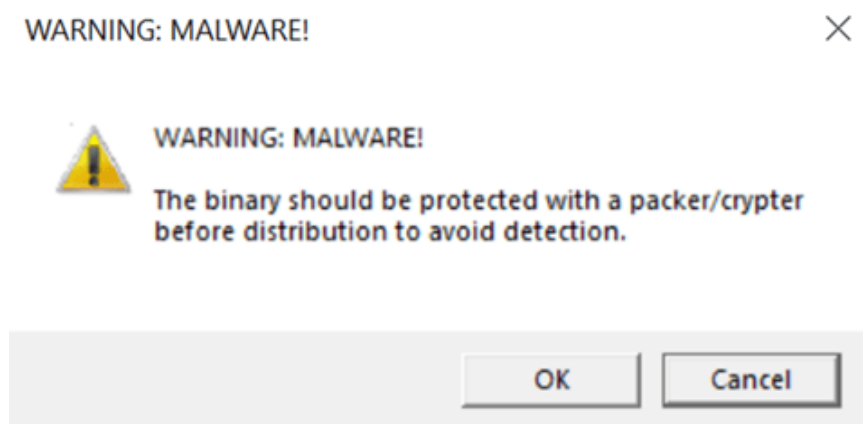


Figure 9. Snippet from x64dbg runtime.

We observed that this executable has several anti-analysis methods. The malware employs anti-debugging techniques using *ZwQueryInformationProcess* to query process debug-related attributes, such as *ProcessDebugPort* and *ProcessDebugFlags*. This allows the malware to detect the presence of a debugger and alter execution accordingly.

In addition to looking for a debugger, the malware queries *ProcessInstrumentationCallback* (0x25) via *ZwQueryInformationProcess* to detect the presence of instrumentation callbacks often used by EDRs or monitoring tools. If detected, the malware may alter execution or evade analysis.

C2 Communication

The malware invoked *InternetConnectA* with a URL pointing to *https[:]//telegram[.]me/sre22qe* and *https[:]//steamcommunity[.]com/profiles/76561198777118079*. This indicates the use of *WinINet*-based communication for remote interaction.

Additionally, the presence of *Sleep* referenced in a preserved register suggests potential anti-analysis or execution delay behavior prior to network communication.

00007FFF99D7412F	CC	int3	
00007FFF99D74130 <win	48:895C24 08	mov qword ptr ss:[rsp+8],rbx	[rsp+8]: "https://telegram.me/sre22
00007FFF99D74135	48:896C24 10	mov qword ptr ss:[rsp+10],rbp	[rsp+10]: "https://telegram.me/sre22
00007FFF99D7413A	48:897424 18	mov qword ptr ss:[rsp+18],rsi	
00007FFF99D7413F	57	push rdi	
00007FFF99D74140	41:54	push r12	
00007FFF99D74142	41:55	push r13	

Figure 10. Call to *InternetConnectA*.

qword ptr ss:[rsp+8]=[0000001E197EDF70 &"https://telegram.me/sre22qe"]=00007FF75A648A30 "https://telegram.me/sre22qe"
rbx=<kernel32.Sleep>
.text:00007FFF99D74130 wininet.dll:\$124130 #123530 <InternetConnectA>

Figure 11. *Sleep* API pointer.

qword ptr ss:[rsp+8]=[000000A8335AD900 &"https://steamcommunity.com/profiles/76561198777118079"]=00007FF684498030 "https://steamcommunity.com/profiles/76561198777118079"
rbx=<kernel32.Sleep>
.text:00007FFF99D74130 wininet.dll:\$124130 #123530 <InternetConnectA>

Figure 12. Call to *InternetConnectA*

qword ptr ss:[rsp+8]=[000000A8335AD900 &"https://steamcommunity.com/profiles/76561198777118079"]=00007FF684498030 "https://steamcommunity.com/profiles/76561198777118079"
rbx=<kernel32.Sleep>
.text:00007FFF99D74130 wininet.dll:\$124130 #123530 <InternetConnectA>

Figure 13. *Sleep* API pointer.

The malware constructs an HTTP GET request using *HttpOpenRequestA*, targeting the path */sre22qe* on the host *telegram[.]me*. This results in a full request to *https[:]//telegram[.]me/sre22qe*. The use of a GET request suggests that the malware is likely attempting to retrieve configuration data or staging information from a remote resource prior to executing further actions such as data exfiltration.

It also establishes an HTTP connection to *steamcommunity[.]com* using the WinINet API. It constructs a GET request targeting the following resource:

https[:]//steamcommunity[.]com/profiles/76561198777118079.

This behavior is implemented via *InternetConnectA* followed by *HttpOpenRequestA*, where the HTTP method ("GET") and object path ("/profiles/76561198777118079") are explicitly defined in memory.

00007FFF99D99530 <wininet.HttpOpenRequestA>	48:88C4	mov rax,rsi	HttpOpenRequest
00007FFF99D99535	48:8958 08	mov qword ptr ds:[rax+8],rbx	rbx:"HTTP/1.1
00007FFF99D99537	48:8968 10	mov qword ptr ds:[rax+10],rbp	
00007FFF99D99538	48:8970 18	mov qword ptr ds:[rax+18],rsi	rsi:"GET"
00007FFF99D9953F	57	push rdi	rdi:"/profiles/765611987771180;
00007FFF99D99540	48:81EC 80000000	sub rsp,80	
00007FFF99D99547	48:88F2	mov rsi,rdx	rsi:"GET"
00007FFF99D9954A	49:88F8	mov rdi,r8	rdi:"/profiles/765611987771180;
00007FFF99D9954D	33D2	xor edx,edx	
00007FFF99D9954F	48:88E9	mov rbp,rcx	
00007FFF99D99552	48:8D48 98	lea rcx,qword ptr ds:[rax+68]	
00007FFF99D99556	49:88D9	mov rbx,r9	rbx:"HTTP/1.1
00007FFF99D99559	44:8D42 58	lea r8d,qword ptr ds:[rdx+58]	
00007FFF99D9955D	E8 D612F7FF	call <JMP.6memset>	
00007FFF99D99562	48:8D4C24 50	lea rcx,qword ptr ss:[rsp+50]	
00007FFF99D99567	E8 80AEEFF	call wininet.7FFF99C8448C	

Figure 14. Calls *HttpOpenRequestA* to construct an HTTP request.

The use of a legitimate platform such as Steam suggests potential abuse of public web content for C2 or configuration retrieval.

The malware initiates an HTTP request using *HttpSendRequestA*. At the time of the call, no request headers or payloads are present, and the optional data buffer is NULL with a length of zero. This suggests that the request is used as a beacon or polling mechanism rather than immediate data exfiltration.

The image shows a snippet of assembly code on the left and a C structure definition on the right. The assembly code includes instructions for moving data from the stack to registers and calling `HttpSendRequestA`. The structure definition lists parameters for the `HttpSendRequest` function, including `dwOptionalLength`, `hRequest`, `lpzHeader`, `dwHeadersLength`, and `lpOptional`.

```

mov dword ptr ss:[rsp+20],eax
mov rcx,rsi
xor edx,edx
xor r8d,r8d
mov r9,qword ptr ss:[rsp+1048]
call qword ptr ds:[<&HttpSendRequestA>]

DWORD dwOptionalLength
HINTERNET hRequest
LPCTSTR lpzHeader
DWORD dwHeadersLength
LPVOID lpOptional = r9:EntryPo
HttpSendRequest

```

Figure 15. Call to *HttpSendRequestA*

The destination endpoint was identified earlier via *InternetConnectA* / *HttpOpenRequestA*, indicating a structured C2 communication channel.

The malware also performs DNS resolution for *gz[.]technicalprorj[.]xyz* via public DNS (8.8.8[.]8), indicating dynamic infrastructure usage prior to establishing outbound communication.

Observed network activity aligns with known patterns associated with Vidar, confirming the final objective of the infection chain. This includes likely exfiltration of credentials, browser data, and other sensitive information.

Post-Execution Cleanup and Artifact Removal

The malware (*MicrosoftToolkit.exe*) was observed terminating its own processes and removing files dropped during execution. This behavior is consistent with indicator removal and operational cleanup.

This capability significantly reduces forensic artifacts and may hinder retrospective investigation.

Post-Execution Cleanup and Anti-Analysis Behavior

During dynamic analysis, the malware (*MicrosoftToolkit.exe*) was observed performing post-execution cleanup activities, including:

- Deletion of dropped or staged files from the disk
- Removal of execution artifacts
- Termination of its own processes

This behavior indicates an attempt to minimize forensic artifacts and reduce the detection footprint on the infected system.

The malware is walking a linked list of *.dot* payload files, resetting their attributes, deleting them from disk, and freeing all associated memory structures to erase traces.

The image shows a snippet of assembly code on the left and a C structure definition on the right. The assembly code includes instructions for moving data from registers to the stack, calling `SetFileAttributesA`, `DeleteFileA`, and `LocalFree`. The structure definition lists parameters for the `SetFileAttributes` and `DeleteFile` functions, including `lpFileName` and `dwFileAttributes`.

```

00007FF6023F4409 . 48:8B08 mov rcx,qword ptr ds:[rbx]
00007FF6023F440C . BA:80000000 mov edx,80
00007FF6023F4411 . 48:FF15 684E0000 call qword ptr ds:[<&SetFileAttributesA>]
00007FF6023F4418 . 0F1F4400 00 nop dword ptr ds:[rax+rax],eax
00007FF6023F441D . 48:8B08 mov rcx,qword ptr ds:[rbx]
00007FF6023F4420 . 48:FF15 214E0000 call qword ptr ds:[<&DeleteFileA>]
00007FF6023F4427 . 0F1F4400 00 nop dword ptr ds:[rax+rax],eax
00007FF6023F442C . > 48:8B0F mov rcx,qword ptr ds:[rdi]
00007FF6023F442F . 48:8B58 08 mov rbx,qword ptr ds:[rbx+8]
00007FF6023F4433 . 48:FF15 5E4E0000 call qword ptr ds:[<&LocalFree>]
00007FF6023F443A . 0F1F4400 00 nop dword ptr ds:[rax+rax],eax
00007FF6023F443F . 48:8BCF mov rcx,rdi
00007FF6023F4442 . 48:FF15 4F4E0000 call qword ptr ds:[<&LocalFree>]

LPCTSTR lpFileName
DWORD dwFileAttributes = FILE_ATTRIBUTE_NORMAL
SetFileAttributes

LPCTSTR lpFileName
DeleteFile

HLOCAL hMem
rbx:&"C:\\Users\\REM\\AppData\\Local\\Temp\\IXP000.TMP\\
LocalFree

HLOCAL hMem
LocalFree

```

Figure 16. Deletion routine for the files written on disk.

After finishing cleaning up its dropped .dot files, the malware runs an internal routine, then switches its working directory to the staging folder to continue execution using relative paths.

```

00007FF6023F44CF | . E8 40440000 | call <microsof toolkit.sub_7FF6023F8914> | LPCTSTR lpPathNam
00007FF6023F44D4 | > 48:8D0D 65530000 | lea rcx,qword ptr ds:[7FF6023F9840] | SetCurrentDirectory
00007FF6023F44DB | . 48:FF15 664E0000 | call qword ptr ds:[<&SetCurrentDirectory>]

```

Figure 17. Transition in execution flow.

Afterwards, it iterates through a directory using *FindNextFileA* and deletes each file it finds, cleaning up other files dropped onto the disk.

```

00007FF6023F2839 | . 0F1F4400 00 | nop dword ptr ds:[rax+rax],eax | LPCTSTR lpFileNam
00007FF6023F283E | . 48:8D4D 60 | lea rcx,qword ptr ss:[rbp+60] | DeleteFile.
00007FF6023F2842 | . 48:FF15 FF690000 | call qword ptr ds:[<&DeleteFileA>]
00007FF6023F2849 | . 0F1F4400 00 | nop dword ptr ds:[rax+rax],eax | LPWIN32_FIND_DATA lpFindFileD:
00007FF6023F284E | > 48:8D5424 20 | lea rdx,qword ptr ss:[rsp+20] | HANDLE hFindFil
00007FF6023F2853 | . 48:8BCE | mov rcx,rsi | FindNextFile
00007FF6023F2856 | . 48:FF15 A36B0000 | call qword ptr ds:[<&FindNextFileA>]

```

Figure 18. File-enumeration and deletion.

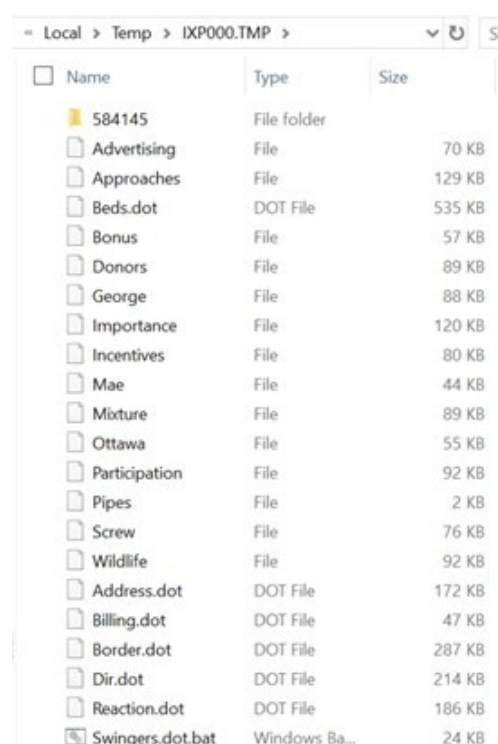


Figure 19. Snippet of all the files/folder dropped on disk and later deleted.

The malware performs final cleanup and then terminates itself using *RtlExitUserProcess*, indicating the end of a single execution cycle after completing its file-based payload operations.

```

00007FF830A3F060 | 48:83EC 28 | sub rsp,28 | FatalExit
00007FF830A3F064 | FF15 E6700600 | call qword ptr ds:[<&RtlExitUserProcess>]

```

Figure 20. Termination of the current process.

Such cleanup routines are commonly used by malware for:

- Defense evasion, by removing indicators of compromise (IOCs)
- Operational security (OPSEC), reducing the likelihood of detection by security tools
- Anti-analysis, limiting the ability of analysts to recover payloads

Observed behavior is consistent with anti-analysis or OPSEC techniques.

Threat Assessment

The observed activity reflects a commodity stealer infection chain with moderate-to-high sophistication in execution flow and evasion. While individual techniques are not novel, their combined use demonstrates:

- Effective blending with legitimate tools
- Reduced on-disk footprint
- Rapid transition from execution to data exfiltration

Vidar is known to target:

- Browser-stored credentials
- Session cookies
- Cryptocurrency wallets
- System and user data
- The presence of C2 communication indicates a high risk of data compromise.

Conclusion

This intrusion highlights the continued effectiveness of script-based, multi-stage loaders in delivering commodity information stealers such as Vidar. A sophisticated multi-stage loader infection leveraging Windows-native tools and file-masquerading techniques. The attacker avoids dropping a single identifiable malware binary and instead reconstructs and executes payloads dynamically through staged file manipulation.

The use of AV enumeration, LOLBin chaining, and script-based execution strongly indicates a preparatory stage for deeper compromise.

The incorporation of defense evasion and artifact cleanup techniques underscores the need for behavioral detection and proactive threat hunting to identify such activity before data exfiltration occurs.

The observed self-cleanup behavior further complicates forensic analysis and may hinder traditional incident response efforts.

Remediation and Mitigation

- Affected systems should be immediately isolated from the network to prevent further data exfiltration and lateral movement.
- Given the capability to download additional malware, full system reimaging is strongly recommended to ensure complete eradication.
- As Vidar primarily targets credential theft, all potentially compromised credentials including browser, email, VPN, and administrative accounts must be reset, and active sessions invalidated.

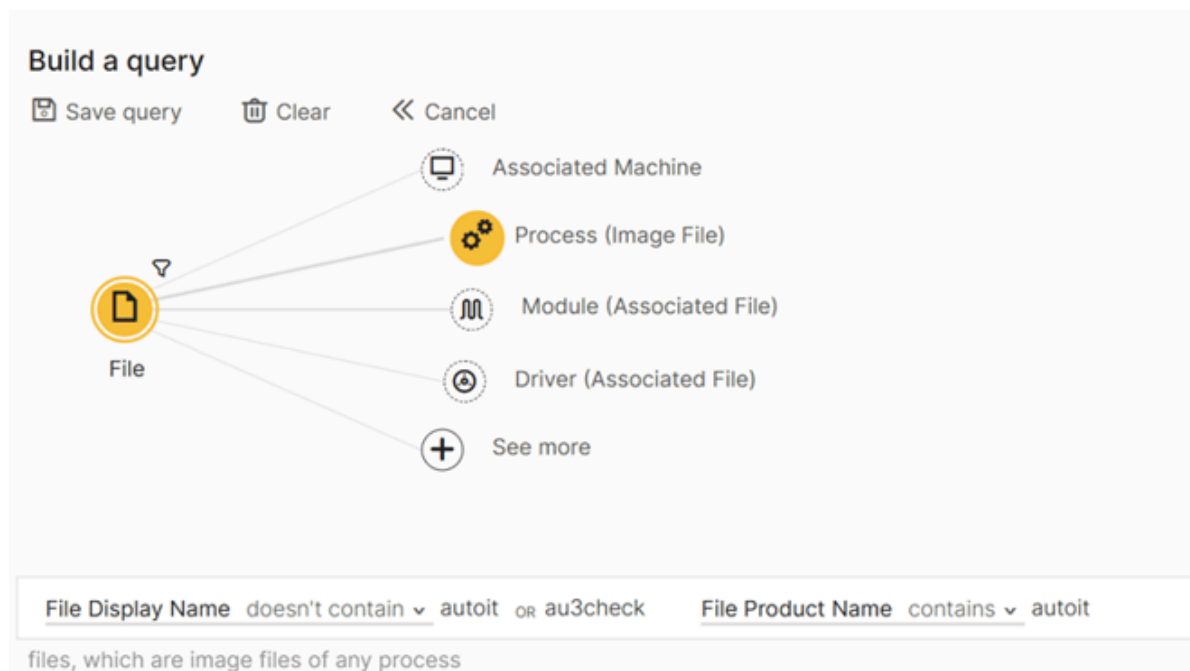
- Multi-factor authentication (MFA) should be enforced across critical services to reduce the risk of unauthorized access.
- Network defenses should be strengthened by monitoring DNS and outbound traffic for suspicious connections, including communication with newly registered or uncommon domains.
- To prevent recurrence, organizations should restrict the execution of unauthorized or untrusted tools.

Hunting Query

Note that the query is already run by LevelBlue's Global SOC in the environments of MDR Complete customers to ensure that possible infections are detected and notified as soon as possible.

1. To detect the execution of renamed Autoit execution, we recommend running the following hunting query in LevelBlue's Defense Platform.

File Element -> add the filters "File Display Name doesn't contain autoit or au3check" and "File Product Name contains autoit"



We provided the following hunting query to obtain the same result:

```
https://[yourenvironment]/#/s/search?queryString=0<-
File"elementDisplayName:!autoit%7Cau3check,productName:@autoit"-
>Process.imageFile&sorting=creationTime&sortingDirection=-1
```

2. To detect communication to domains and IPs associated with this infection, we recommend running the following hunting query in LevelBlue's Defense Platform:

Domain name Element -> add the filters "Domain Display Name contains gz.technicalprorj[.]xyz or telegram[.]me" and select the "DNS query resolved Domain to IP (Source Domain)" Element.

Build a query

Save query

Clear

Domain name

DNS query resolved Domain to IP (Source Domain)

DNS query resolved Domain to Domain (Source Domain)

DNS query resolved Domain to Domain (Target Domain)

DNS query resolved IP to Domain (Target Domain)

See more

Domain Display Name

contains

gz.technicalprorj.xyz

OR

telegram.me

domain names, which are source domains of any dnsqueryresolveddomaintoip

We provide the following hunting query to obtain the same result:

```
https://[yourenvironment]/#/s/search?queryString=0<-
DomainName"elementDisplayName:@gz.technicalprorj.xyz%7Ctelegram.me"-
>DnsQueryResolvedDomainToIp.sourceDomain
```

LevelBlue’s Security Recommendations

We recommend the following actions in LevelBlue’s Defense Platform:

- Enable **Application Control** to block the execution of malicious files.
- Enable **Anti-Ransomware** in your environment’s policies, set the Anti-Ransomware mode to Prevent, and enable Shadow Copy detection to ensure maximum protection against ransomware.
- Enable **Variant Payload Prevention** with prevent mode on LevelBlue’s Behavioral Execution Prevention tool.

LevelBlue is dedicated to teaming with defenders to end cyberattacks from endpoints to the enterprise to everywhere. Learn more about LevelBlue’s [XDR](#), check out our [Extended Detection and Response \(XDR\) Toolkit](#), or [schedule a demo](#) today to learn how your organization can benefit from an operation-centric approach to security.

IOCs

IOC	IOC type	Description
-----	----------	-------------

fc27479ff929d846e7c5c5d147479c81e483a2ec911bd1501a53aa646a29620d	SHA-256	MicrosoftToolkit.exe
d4fe9f48178cdf375a3be30d17f1dc016b5861dff8683f0bb35a0ba8d44f892f	SHA-256	swingers.dot.bat
978ad86c90d85b74947bb627ec24f8bcd26812b500e82f5af202160506ac29c6	SHA-256	Beds.dot
881619a47b62b52305d92640cc4d4845a279c23a5a749413785fc8fcb0fdf7fb	SHA-256	replies.scr
968ecf51c442ec0ff91f91689ac524e7e8e9eab0c1a2a65cf13e54cf95194efe	SHA-256	D (payload file)
149.154.167[.]99	IP Addresses	Vidar-associated C2 IP
telegram[.]me	Domain Name	C2 domain
gz[.]technicalprorj[.]xyz	Domain Name	Vidar-associated C2 domain

MITRE ATT&CK Mapping

Tactic	Techniques / Sub-Techniques	Summary
TA0001 - Initial Access	T1204.002 – User Execution: Malicious File	User downloaded and executed microsofttoolkit.exe (hacktool), serving as the initial entry point into the system
TA0002 - Execution	T1204.002 – User Execution: Malicious File	User executed microsofttoolkit.exe, initiating the infection under the assumption of legitimate software activation
TA0002 - Execution	T1059.003 – Command and Scripting Interpreter: Windows Command Shell	Batch/script-based execution used to stage further activity
TA0002 - Execution	T1027 – Obfuscated/Compressed Files	Payload staged via compressed or embedded format using extract32.exe

TA0002 - Execution	T1140 – Deobfuscate/Decode Files or Information	Extraction process used to unpack the next-stage payload
TA0002 - Execution	T1059 – Command and Scripting Interpreter	Autolt-based loader executed via script-like behavior
TA0002 - Execution	T1218 – Signed Binary Proxy Execution	.scr (Autolt compiled binary) used as a loader to execute malicious logic
TA0005 - Defense Evasion	T1036 – Masquerading	A .dot file was renamed to .bat to bypass basic file-type restrictions
TA0005 - Defense Evasion	T1562.001 – Disable or Modify Security Tools	taskkill.exe used to terminate security-related processes
TA0005 - Defense Evasion	T1059.003 – Command Shell	findstr.exe leveraged for filtering and identifying security processes
TA0005 - Defense Evasion	T1070.004 – Indicator Removal on Host: File Deletion	Malware deleted dropped files to remove evidence
TA0005 - Defense Evasion	T1489 – Service Stop	Processes terminated to reduce forensic artifacts and evade detection
TA0011 - Command and Control	T1071.001 – Application Layer Protocol: Web Protocols	Malware communicated with C2 over HTTP/HTTPS
TA0011 - Command and Control	T1573 – Encrypted Channel	Encrypted communication used to evade detection
TA0010 - Exfiltration	T1041 – Exfiltration Over C2 Channel	Stolen data (credentials, browser data) exfiltrated via C2 channel

Stay Informed

Sign up to receive the latest security news and trends straight to your inbox from LevelBlue.

WEBINAR

[Tracing a Multi-Vector Malware Campaign: From VBS to Open Infrastructure](#)

About the Author

Mahadev Joshi is a Security Analyst with the Global SOC team. He is passionate about cybersecurity and malware analysis, with a focus on understanding and countering advanced threats. He is eager to learn more and stay ahead of emerging threats. Mahadev has a Bachelor of Science in Information Technology.

ABOUT LEVELBLUE

LevelBlue secures what's next with intelligence-led security delivering visibility and speed to stop threats faster. As the world's largest and most analyst-recognized pure-play managed security services provider, our AI-powered managed services and cyber expertise across managed, advisory, and incident response services help clients operate with confidence. Learn more [about us](#).

[Emerging Threats](#) [Threat Intelligence](#)