

Vidar Stealer Unmasked: Code Signing Abuse, Go Loaders and File Inflation

 unit42.paloaltonetworks.com/vidar-stealer-xmrig-miner-campaign-analysis

Bharath Nannaka, Pranay Kumar Chhaparwal

July 7, 2026

Executive Summary

In April 2026, Unit 42 researchers identified a financially motivated campaign delivering Vidar stealer and the XMRig cryptocurrency miner to consumer and small- and medium-sized business victims worldwide.

Attackers lure victims via malvertising to pages for downloading files that impersonate cracked versions of copyright-protected software. Upon execution, the loader drops and runs both Vidar stealer and XMRig. Vidar stealer targets information like browser credentials, cookies and crypto wallets. XMRig mines Monero cryptocurrency.

We assess the operator of the campaign to be a Vidar stealer malware-as-a-service (MaaS) affiliate involved in operations targeting victims in the U.S. and European Union. This article provides a technical analysis of the campaign.

Palo Alto Networks customers are better protected from the threats discussed above through the following products and services:

- [Cortex XDR](#) and [XSIAM](#)
- [Advanced WildFire](#)
- [Advanced URL Filtering](#) and [Advanced DNS Security](#)
- [Prisma Browser](#)

If you think you might have been compromised or have an urgent matter, contact the [Unit 42 Incident Response team](#).

Related Unit 42 Topics [Vidar Stealer](#), [Malware](#), [Cryptocurrency](#), [XMRig](#)

Attack Timeline

Since its emergence, attackers have used Vidar stealer in multiple large-scale campaigns. We identified a notable spike in activity from mid-late April 2026, primarily targeting organizations in the U.S. and the EU.

Figure 1 illustrates the timeline of the April 2026 campaign, which shows the number of Vidar stealer samples discovered each day.

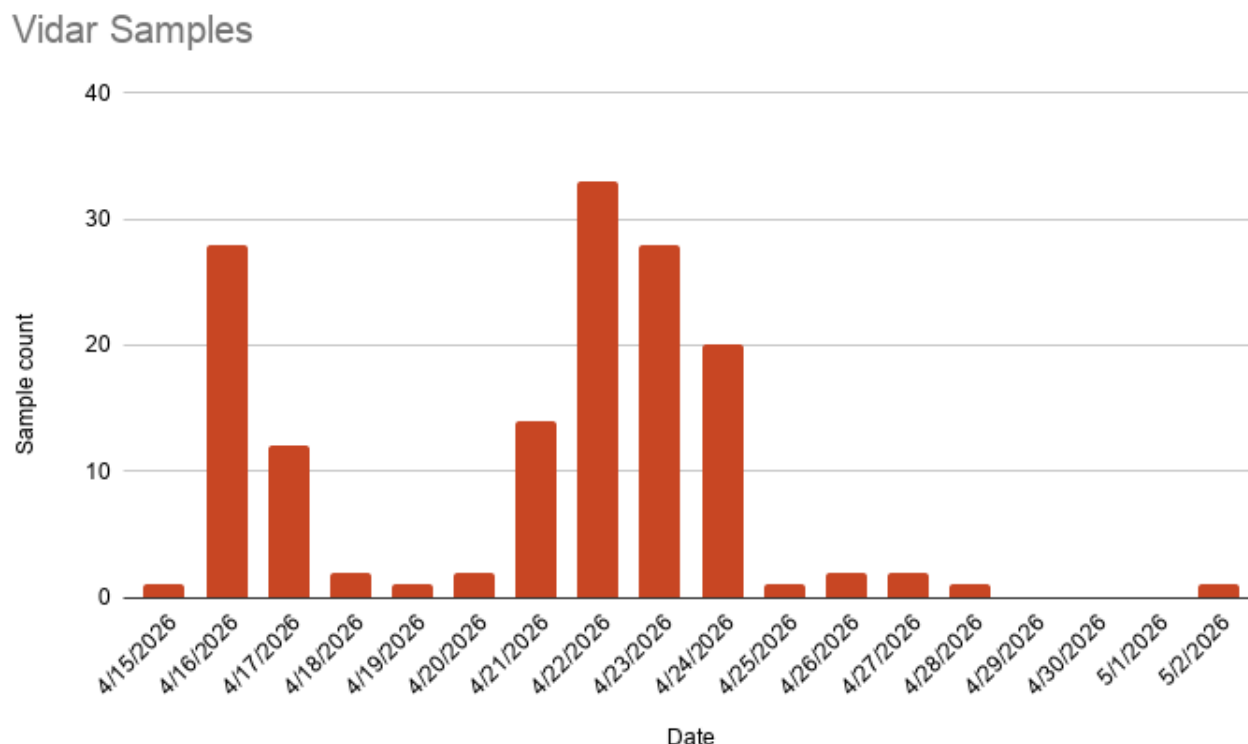


Figure 1. April 2026 campaign.

Our investigation of this activity led to the discovery of loader binaries distributing both Vidar stealer and XMRIg.

Initial Access: Malvertising via Fake Software Cracks

This group behind this campaign distributes loader binaries through [malvertising](#), targeting victims who search for pirated or cracked versions of copyright-protected software. The filenames used in this campaign mimic cracked versions of popular copyright-protected programs as well as generic installers.

The campaign delivers the malware in password-protected archives with a .bin extension in the filenames. This appears to be a deliberate choice to bypass email gateway scanning and to prevent automated sandbox detonation without the password.

We initially discovered 43 of these loader binaries that deliver Vidar stealer and XMRIg. Upon extraction and execution, the loader binary is signed with a certificate (subject CN=justwatch[.]com), creating a false sense of legitimacy before any malicious activity begins.

Our analysis of these loader samples indicates they use the Factory-v3 framework.

Factory-v3/UpdateFactory Builder

Factory-v3 has been [described as a MaaS builder](#) used for different families of stealer malware. All 43 samples we discovered from this campaign contain embedded Go build metadata identifying the Factory-v3 framework.

The Factory-v3 builder's internal name of UpdateFactor is revealed in a developer/build machine path shown in the programming database (PDB) strings of the loader DLL files:

C:\Users\Administrator\Desktop\UpdateFactory\compiler\1.25.9\go\src\runtime\cgo

Figure 2 shows a diagram of information from the build machine metadata.

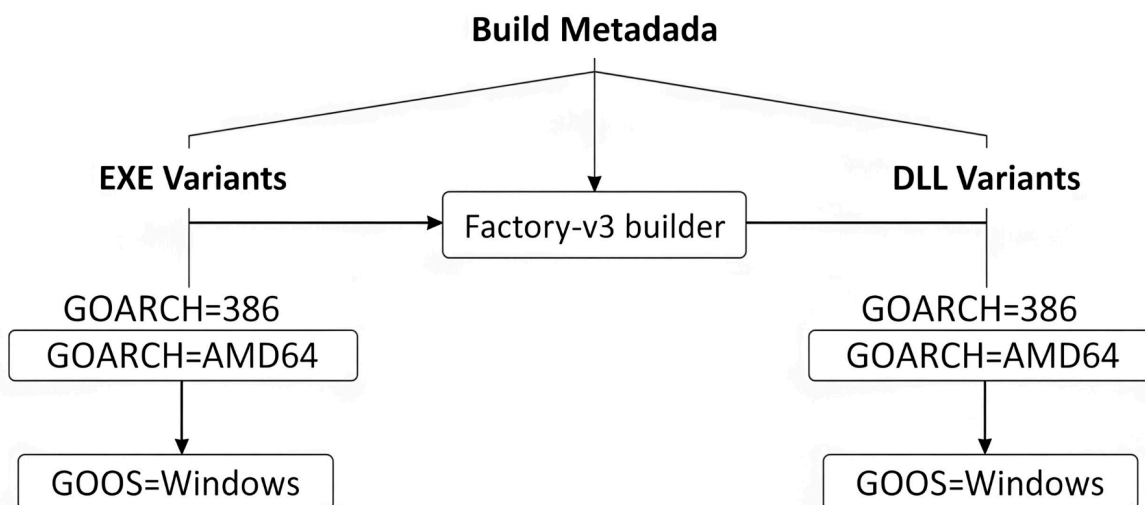


Figure 2. Information from the build machine path.

The builder generates a unique binary per build. For example, we observed 27 unique build UUIDs across 43 samples, defeating hash-based detection. The builder uses [Go version 1.25.9](#), a custom pre-release of tools for the Go programming language.

Anti-forensic measures are consistent across all samples:

- The PE TimeDateStamp is zeroed
- No PE version info is present
- DLL imports are reduced to kernel32.dll only
- User-defined type names are obfuscated to a V##### pattern

The same builder, toolchain and Authenticode certificate infrastructure underpin a concurrent Lumma stealer campaign. This indicates Factory-v3 is used as a service for multiple stealer affiliates.

Rogue Authenticode Certificate

All 43 loader samples carry an Authenticode signature fabricated to impersonate JustWatch GmbH, a legitimate German streaming guide service. JustWatch has not been compromised. The certificate is entirely fabricated using a self-signed root certificate authority (CA) that is not present in any public trust store, as shown below in Figure 3.

```
Subject      CN=justwatch.com
Issuer       CN=WR3 (rogue self-signed CA)
Serial       2f:7e:f0:15:7d:17:62:5c:09:86:91:ce:f1:ff:7d:63
Key  RSA 4096-bit, CA:TRUE, Extended Key Usage: Code Signing + TLS
Valid From   2026-03-09
Valid To     2026-06-07
```

Figure 3. Rogue certificate information.

Because the certificate is not chained to a Microsoft-trusted root, Windows SmartScreen and Authenticode validation will flag the binary as untrusted. However, the visual presence of a recognizable brand name in the signature dialog is sufficient to deceive many victims into proceeding.

Sample Clusters

The 43 loader samples fall into four clusters, as Table 1 below shows.

Cluster	CPU Architecture	File Type	Frequency Count	Role
A	x64	EXE	26	Go loader (file-inflated EXE)
B	x64	DLL	13	Fake MpClient.dll sideload
C	x86	EXE	3	Go loader 32-bit (shared with Lumma Stealer campaign)
D	x64	EXE	1	Vidar core payload

Table 1. Sample clusters with filetype.

Cluster B DLL variants export Windows Defender MpClient.dll API functions to enable DLL search-order hijacking ([MITRE ATT&CK® T1574.002](#)). When a legitimate Windows Defender binary attempts to load MpClient.dll, the operating system locates the malicious copy first if it is placed in a higher-priority search path. The exported function names mimicked include MpAllocMemory, MpClientUtilExportFunctions, MpConfigOpen, MpFreeMemory and nine others.

File-Size Inflation

Loaders in Clusters A and C append hundreds of megabytes of null bytes after the last PE section, pushing the total file size to as high as 491 MB. Most automated sandbox environments enforce an upper file-size limit of 50-100 MB and silently skip oversized submissions, meaning the malware never executes in the analysis environment.

The real malicious content in the largest observed sample is only 2.3 MB, and the remaining 489 MB is null byte padding. Defenders should ensure security tooling removes null byte padding before applying size limits, since the same sample compresses to approximately 2.4 MB.

AMSI Bypass

Static disassembly of the Vidar core payload sample (SHA256 hash: 7ed4a256e1d281cb4f194d13ff554fb280dafde0a67a18115ea038ea6c87d) reveals an in-memory Antimalware Scan Interface (AMSI) bypass that executes before any stealer logic runs. The routine loads amsi.dll, resolves the AmsiScanBuffer variable and overwrites its first six bytes with a patch. This patch forces the function to return E_INVALIDARG, which might disable Windows AMSI for all subsequent script and code execution on the victim machine.

Both the DLL name and function name are [XOR-obfuscated](#) with single-byte key 0x05 to evade static string scanning. Table 2 shows the encoded and decoded strings for these names.

Encoded String	XOR Key	Decoded String
dhvl+aii	0x05	amsi.dll
DhvlVfdkGpcc\`w	0x05	AmsiScanBuffer

Table 2. Encoded and decoded strings from the sample.

Figure 4 below shows patched bytes written to the AmsiScanBuffer location (0x80070057).

```
B8 57 00 07 80 → mov eax, 0x80070057 ; HRESULT E_INVALIDARG
C3 → ret
```

Figure 4. AMSI buffer bytes.

The binary uses a second, longer obfuscation layer for larger data blobs. This layer is obfuscated by a 32-byte rotating XOR with the key 69946018ddda1058ce5c2a556c78a747838865c47074dcb165effb0840cb1cf5 applied to the Telegram bot token, Monero wallet address and mining pool hostname for the XMRig payload.

Attack Chain

Figure 5 shows that the attack chain begins with malvertising, luring victims into downloading a password-protected .bin archive disguised as a cracked version of a legitimate program.

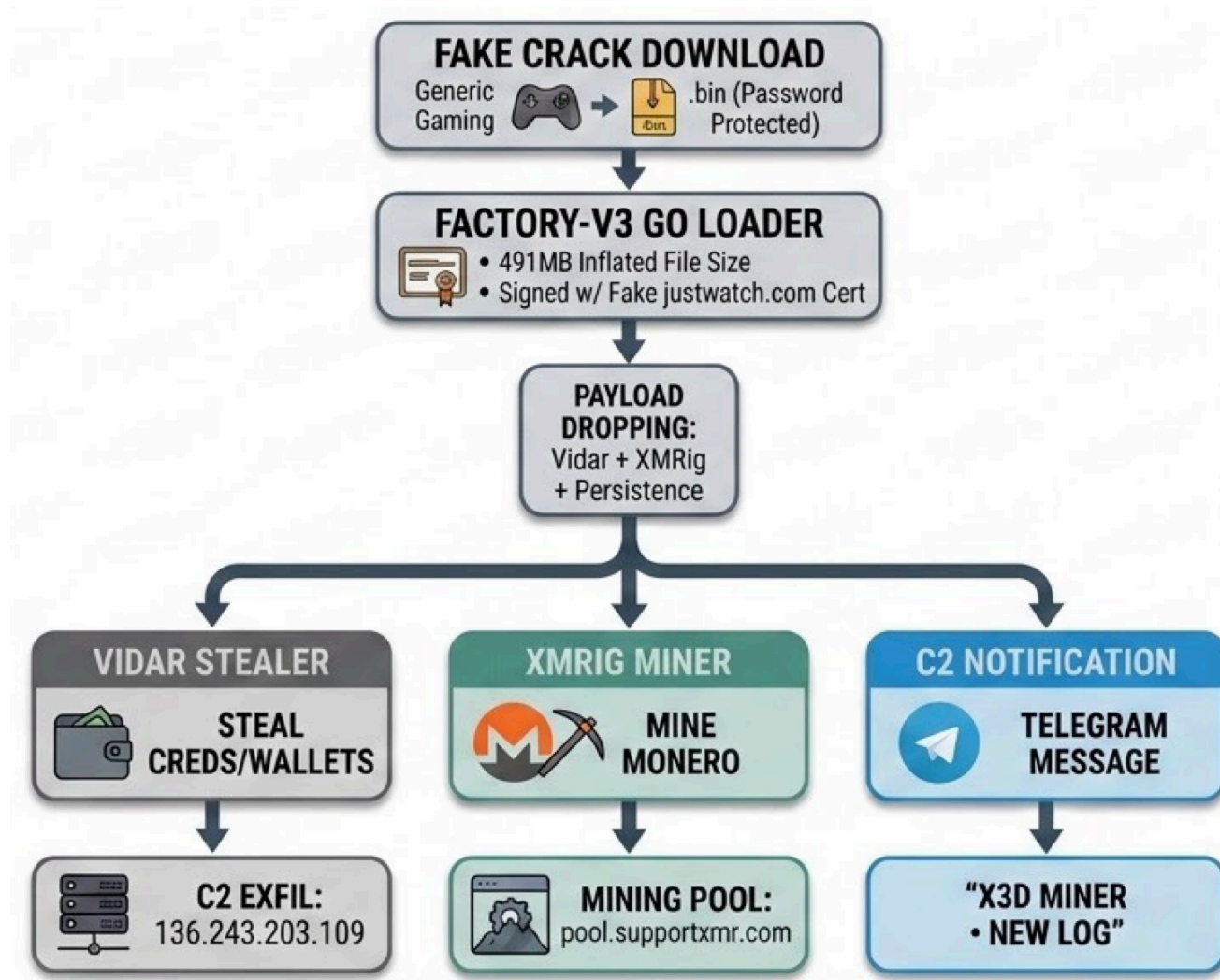


Figure 5. Execution chain X3D MINER/Vidar Stealer via Factory-v3.

The loader extracted from the .bin archive exhibits the following features:

- It leverages the Factory-v3 Go framework

- It is signed with a fake JustWatch certificate
- It is padded with null bytes to reach a large file size of hundreds of MB to evade detection

The malware employs anti-analysis techniques such as process enumeration, alongside an AMSI bypass where the AmsiScanBuffer function is patched to prevent detection by some types of security software.

Subsequently, the malware drops multiple payloads including Vidar stealer and the XMRig cryptocurrency miner, while establishing persistence mechanisms through registry modifications and scheduled tasks.

The next stage involves reconnaissance, as the malware gathers information about files, hardware IDs (HWID) and bypasses proxies. Vidar stealer then exfiltrates sensitive data such as credentials and cryptocurrency wallets, communicating with a command-and-control (C2) server at 136.243.203[.]109.

Simultaneously, XMRig begins mining Monero using the mining pool at pool.supportxmr[.]com. Finally, the attacker is notified of new activity via Telegram, with messages labeled "X3D MINER • NEW LOG," ensuring the operator stays informed about successful infections and stolen data. This means the threat actor behind Vidar is deploying the X3D MINER XMRig package.

Dynamic analysis of the Vidar core payload confirmed the following execution sequence:

1. **Geolocation beacon:** GET request to ip-api[.]com/json resolves the victim's public IP address and country, which are embedded in the subsequent Telegram alert
2. **Payload drop:** It drops MicrosoftUpdate.exe in the %TEMP% directory as part of the Vidar stealer component and it places the following files in %AppData%\Temp\AppData%\Temp
 1. MicrosoftEdgeUpdate.exe (the XMRig launcher)
 2. libuv-1.dll (an XMRig dependency)
 3. WinRing0x64.sys (an XMRig kernel driver)
 4. mgwthmc2.dat (an XMRig Monero configuration file)
 5. It copies itself to this folder as NisSrv.exe for persistence
3. **Vidar stealer:** MicrosoftUpdate.exe is written to %TEMP% and executed. It targets browser credential stores, cookies and crypto wallet data, packaging everything into a ZIP for exfiltration to 136.243.203[.]109:443.

4. **XMRig miner:** MicrosoftEdgeUpdate.exe is launched with --config=mgwthmc2.dat. The configuration is built entirely in memory before being written to disk, with the Monero wallet address and pool details decrypted from encrypted blobs. Each victim's C:\ volume serial number is hashed into an 8-character HWID that is appended to the auth_token field, allowing the operator to track per-victim mining output in the Monero pool dashboard.

Persistence Mechanisms

The payload establishes persistence through three parallel mechanisms:

Registry Run key:

- HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run value name SystemAgentService, data "...\\AppData\\Roaming\\Microsoft\\Windows\\Temp\\NisSrv.exe" -s
- **Scheduled task:**
schtasks /create /tn "SystemAgentService" /tr "NisSrv.exe -s" /sc onlogon /f

Startup folder batch script:

C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp\FEbJCNWOCKMJ.bat

All three mechanisms point to the malware file named NisSrv.exe. Attackers chose this filename to blend in with legitimate Windows Defender components, since the real NisSrv.exe is the Network Inspection Service binary.

X3D MINER

The tag X3D MINER appears in Telegram operator notifications sent for every new victim infection. This moniker is used by a group associated with [XMRig and binding XMRig with other programs](#).

The operator behind this campaign runs a dual-monetization scheme. Criminals sell credentials and session cookies stolen by Vidar stealer on criminal log markets, while XMRig provides passive income from hijacked victim CPU cycles. The Factory-v3 builder is assessed to be a separate upstream service used by at least two distinct stealer affiliates.

Variant B

Pivoting on information from the initial 43 loader samples, we identified 56 additional samples of a subsequent variant on April 24, 2026. This variant retains the same builder, delivery and C2 infrastructure.

The operating characteristics, Go loader, Factory-v3 builder, Telegram dead-drop and payload delivery are identical to the original campaign. The single differentiating factor is the Authenticode certificate.

The operator transitioned from using a self-signed certificate mimicking JustWatch to another unauthorized certificate designed to resemble the BleacherReport[.]com certificate. This technique is known as [Code Signing Impersonation](#).

Attackers craft certificates to mimic legitimate, trusted publishers (e.g., Microsoft or Google) by cloning their metadata as Table 3 below shows. It is important to note that Bleacher Report has not been compromised.

	Variant A	Variant B
Subject	CN=justwatch[.]com	CN=*.bleacherreport[.]com
Issuer	CN=WR3 (rogue self-signed CA)	CN=GlobalSign Atlas R3 DV TLS CA 2026 Q1 (Cloned issuer name)
Chain is trusted?	No	No
Certificate type	Fake Authenticode (CA:TRUE)	Certificate chain could not be built to a trusted root authority (fake certification)

Table 3. Certificate data from the old and new variants of Vidar stealer.

We observed these samples contacting the Telegram channel ci0iiif. New C2 servers in this cluster include 138.199.246[.]13, 116.203.243[.]208 and 136.243.203[.]111.

Conclusion

This campaign demonstrated a multi-layer evasion approach. This approach combined the following characteristics:

- Rogue certificates

- Go-compiled loaders with per-build unique hashes
- Binaries inflated to hundreds of MB with null byte padding
- An in-memory AMSI bypass

Loader samples sharing these characteristics were all delivered via a MaaS platform that also serves other malware stealer families.

The operator's shift from using a self-signed certificate to leveraging an unauthorized certificate in Variant B demonstrates the actor's ability to adapt rapidly.

We recommend that organizations enforce strong Authenticode chain validation and supplement it with:

- Certificate serial blocklisting
- Configuring security tooling to scan files regardless of size
- Monitoring for MpClient.dll loading from non-standard paths.

Defenders should also:

- Hunt for the persistence indicators and file-drop patterns described above
- Block outbound connections to all C2 addresses and pool.supportxmr[.]com immediately

Palo Alto Networks Protection and Mitigation

Palo Alto Networks customers are better protected from the threats discussed above through the following products:

- [Cortex XDR](#) and [XSIAM](#) help to prevent the threats described in this article, by employing the [Malware Prevention Engine](#). This approach combines several layers of protection, including [Advanced WildFire](#), Behavioral Threat Protection and the Local Analysis module, to prevent both known and unknown malware from causing harm to endpoints.
- The [Advanced WildFire](#) machine-learning models and analysis techniques have been reviewed and updated in light of the indicators shared in this research.
- [Advanced URL Filtering](#) and [Advanced DNS Security](#) identify known domains and URLs associated with this activity as malicious.
- [Prisma Browser](#) provides additional protection layers against advanced web threats including dynamic scans of every loaded web page, to prevent execution of new and unknown malicious attacks such as the malvertising campaign described above, and to protect company assets.

If you think you may have been compromised or have an urgent matter, get in touch with the [Unit 42 Incident Response team](#) or call:

- North America: Toll Free: +1 (866) 486-4842 (866.4.UNIT42)
- UK: +44.20.3743.3660
- Europe and Middle East: +31.20.299.3130
- Asia: +65.6983.8730
- Japan: +81.50.1790.0200
- Australia: +61.2.4062.7950
- India: 000 800 050 45107
- South Korea: +82.080.467.8774

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

Indicators of Compromise

Vidar C2 Server IP Addresses

- 116.203.243[.]208
- 136.243.203[.]109
- 136.243.203[.]111
- 138.199.246[.]13

Code Signing Certificate Information

(Read: Field - Value)

- Subject Common Name (CN) - justwatch[.]com
- Issuer Common Name (CN) - WR3
- Serial Number - 2f:7e:f0:15:7d:17:62:5c:09:86:91:ce:f1:ff:7d:63
- Validity Period - 2026-03-09 to 2026-06-07
- SSL Certificate SHA1 Hash (C2) - ab92f731ab20774dfdb95664ee41a2fbafe2a284

File Paths

(Read: File path - Description)

- %TEMP%\MicrosoftUpdate.exe - Vidar stealer component
- %AppData%\Roaming\Microsoft\Windows\Temp\MicrosoftEdgeUpdate.exe - XMRig launcher
- %AppData%\Roaming\Microsoft\Windows\Temp\NisSrv.exe - Persistence copy of loader

- %AppData%\Roaming\Microsoft\Windows\Temp\libuv-1.dll - XMRig dependency
- %AppData%\Roaming\Microsoft\Windows\Temp\WinRing0x64.sys - XMRig kernel driver
- %AppData%\Roaming\Microsoft\Windows\Temp\mgwthmc2.dat - XMRig Monero configuration
- %Startup%\FEbJCNWOCKMJ.bat - Startup batch script for persistence

Registry Key for Persistence

- Registry key: HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
- Value name: SystemAgentService
- Data: "...AppData\Roaming\Microsoft\Windows\Temp\NisSrv.exe" -s

Scheduled Ask for Persistence

- Task name: SystemAgentService
- Trigger: On user logon (/sc onlogon)
- Action: ...AppData\Roaming\Microsoft\Windows\Temp\NisSrv.exe -s

Import Hash (imphash) Analysis

Table 4 shows the imphashes with the associated sample clusters and the role of those clusters in this activity.

Imphash	Cluster	Role
d42595b695fc008ef2c56aabdb8efd68e	A - x64 EXE (26 samples)	Go loader EXE
d8b31f8c03e0c76ff245ed05a15ffe6c	B - x64 DLL (13 samples)	Fake MpClient.dll
1aae8bf580c846f39c71c05898e57e88	C - x86 EXE (three samples)	Go loader x86
c10333c92889b65c3590ef2b3819b420	D - Vidar core (one sample)	Vidar core payload

Table 4. Imphash information associated with this activity.

SHA256 Hashes - Original Cluster

Cluster A - x64 EXE Loaders (26 Samples)

- 097a87cfa4a5186aba3bba096866692951bde59c6f0c2e8c1c4a599246d14da8
- 201594c9d173bba6cb509407ecba378c19b93da0a81a2182a913c480e6dbb54e

- 20bf39e1e67152039e70a01ad9e7b23c08d23d2a724ef9c44903f3d4353a2275
- 35dde1b2482b12582820a861e7c46f10721af6b75052fc872c05d2230a4e8ca1
- 43920ef7d2742d140a1ab2a1ef172c716903474c73561377dc4f1534d2c5f581
- 47d6d1a38534ba897a5a1e293e3d5df303bbd8e0526e756ad08887ffc1417bef
- 68ced9d7c1b1ff8ffb5f56c7d3f849d4fd16a1b95324426811424b40043d6d25
- 6b7ff061eebeb9ead8812c410247768a7ba90786aeeb1bafa6412cc5b08237b5
- 739cdedb20de39aeb1f15dc8c2dbbf15fa993250fd879bf87443ff9aeaf4997b
- 74df77b6a83d89fa137fd285a2efde36b1d62c00b3be81cc93df7d1e6e94837b
- 8b40cc7d173efd27fb60f3d260acef28f58d67d1f39597e1d611db311a305f62
- 9656d3301f63ef6114289739a1c44082206298f787238fc6c190ad87eab24751
- 9b3df1b6c1b98c201de09a7719066f7bcae6b66a3173b703a617f53fddf67d51
- a17a972a05afe387ed32aa2986d5be8bca2f22619d0aedfa834c6963abfab3bf
- a4f979b4a5d7bc8bc455dd4c09b44e51a389576fccce35a2c8da3ce680237565
- a64843ebfbc39e96ec7613003b1b5c3a9b878874ea15a05e1d34ce91781ebfb6
- aaa2bc1128d8b8b2da76262bf87ede19bac053cca6576efba6aaa71c9438c304
- b830f043076a12748b6a2dc0810ece85439ee77434d991ae7d84201b09ead756
- bb30cc2b302d9a6963109b201b78d4163bb6c2d7bc8bf5a66e9a744b62fc2717
- c328b78c21060e2203ac517833fce41572b91878e187f85fa434cd6914659834
- c7a4a547eb7f6b0b4b75bb6dd8955244bb2618ba234ae740cdedd7c2d30e3465
- d384c403c084967d8c967501ee6332b050af04ef424f13a3f5a88d155389d98c
- d6446f2803444bd2200d48a01a9ad7d487e67e8e831c9cd13f89cbfec17fd4e2
- d7c9c9469c513c05aa431fae34f414f91fc3f794d3e76b6e4d0b92c4cd3ff2e
- d8c1f96107a3349e62b3ab9afc60f62af9c89b6961b637a26b71e1230f2b3b8a
- f0dcb7e407de85d8de8e2221df8dddecac8aec88af8975c9f07e14100f6edb88

Cluster B - x64 DLL Loaders/fake MpClient.dll (13 Samples)

- 0a6a67a2fc4d79ec1cd8afc5b8b7a5e69a406e53d57a7334e097c5d0644de5f6
- 488d941b7b4428b0f4a0e5495e3857b9b96215fb3e7f164b06640d59096425e6
- 5d7324d8b5a25f862ef8223c6766d0e80af3ad168e17312b265e13a3a68e0ded
- 7720e83c02a027d70ae201c393c1956aa2fa8199879a3a4c4fd1d20b03022cfd
- 7e49da0ae2f81e14841f356b4d69f0480c2d9ce3fab5a3fa91b0036d9a36fa0f
- b8b5f6991a3a61083461d5269245bebf28b90934c328848ba8c1e084a5a6216c
- b927d265fa29e471c1ae0d31516e480c09c0fb17f480ad08ea8d5b73e84b7a1b
- b9b6893fa6b04ee8daa29e515c08239ac5204af1a1fa2bc10006eede1b41329b
- c7c37a973b14edd5b6b2da4a1497c593e43640735ff54aecc9a3288fa5e548e3
- d2148a458da46e81702136aa915312d360805f083d1f37ff5531db9fbdb8ad6d
- d7b56818c829960b692de9ad5a14e52669d953e9f074f7218c3fe34ede4a11a0
- db2a872f712fbdb1e347d06e29a9ed8278d86710ffc14ff04422be76e47124f4
- e9e5e748ec5c0b811c8e60b0e55059edb4d2df86ff3ca45969e57d5fecb11a38

Cluster C - x86 EXE Loaders (Three Samples)

- 03e6f4f49cec3af38bbec9ed64c195c7a85a630ec989efb3669f04a2993c1dd7
- 914c18a04a2727bba9cecab78a1d516ec3c7a3f667e0e5a6081aa0e9206a69fe
- d78082dc33c6dca98316e865efa9829c6eb5a97c2ca3cd4ea6c2123a5f6ae45b

Cluster D - Vidar Core Payload (One Sample)

7ed4a256e1d281cb4f194d13ff554fd4fb280dafde0a67a18115ea038ea6c87d

SHA-256 Hashes - Variant B Cluster (56 Samples)

- 15489bcd6e4602b41c9a787ec8d7ab027d5e45d400938048bb1c702ad5937980
- 169a330353e53a409e0109c914404354741ff1e1c64e501738dc05e58ea92abc
- 2a02ec4af5ed591afdf1236a443e3b68642ee133f38a2857d1ead51246ab498
- 2b7297a5f502a2e9a59066f0a370bc5a8b28addd0e27975db3d770f801c15397
- 2c0b344af415b787b396c8e23bbbe112bd471a1ca1d12cf357c48e2ee1ae068c
- 2c6e8f86c05781af12b323311e83e011f1a603928e2086c48e2ca59e33d90dbe
- 2e11a16f94484e0f43eb4572f800f26f0b4a1314cbdae3c44c1ae35f376906d8
- 2f1400a91c853d61622f4d21ed97d96ea1093c0fa1586669bea6f6baa331251f
- 314ce675c040c63b825f213965f5c76a3bd09bf70e138708367e2a84e9e84b30
- 32172e4d8d2ab9fb29b36c9b279117be6ff611b5b91ff7b1c42501a5ec969f2b
- 330efeebba3782994612fdfe20ff96c930af33a83b88a342b6622461511921b2
- 35b51bbe42edd15918b015eaa1b4f0e6b5c94f186d71d887e39f1da69a4dec3f
- 3c3f12531045b7eedfe25e0f291d4792b0d8c8366f8de043e2fa8ecf34ccb913
- 3db33b0423bb9278db267a7adb036ecbd6aeebd7909d06d824919708b1e12e1b
- 3e906ae47e9836a591f44d4b743e961d634a404fa8fd8bfae64f1d54c853be2b
- 4bf770a59d367b532dec32668f86003b17d93918dba5ef5fd2b19c5394252436
- 4f456142caf590d98fb11ca247800bb417766714527e5a4707ac2f5d01542626
- 53d263b292be387843fadb7131c2d538b4262c81f5b95cfacafac2d5446c06e
- 5494909e0f5221db75e933b28981b2d0e118f227b7d8a5980d88b500b76dfc2e
- 54dc05ab56244444f86d69b8274a6075906f7ba2307b08e08d3884abde255495
- 559f46ceb801a3540eace594476718e1486b5b4423cfb4ff64530ff8fb4a3815
- 5838ae6c748dcbdfa13c6529c654cb821897d29835d3e7e05ca23fb2f3794f02
- 59b9153c4c9e155c976db1a2fd4d1b28fa10bb9c4dcdfdc4758b352c037e3d86
- 5b6a466b65d479b77a03b15a95ac097b45e23ff7ae5ef6282985b2a503deb691
- 613e5314a7ded3155cdec49fd34e852e181f4651d78bd8bf3adad2f4dbf22b0d
- 62877a5096828c4bc2fca7cbee7d38b11a0c90fd0d3fc8c37981581e9988c919
- 634e89d8592d7c9e2bc1c098217a813947b44a4f80bc569e9a15c1e8b0864b91
- 67569adec99fd38b114ae07e2e549e6c16f75368f3c5373022c84934ed1c8e84
- 6d49233b1fca22f3823e856e4c16749e9c45f384ea57055fead16df35b217226

- 71c79e8bf71ed257435ea9b8b91e118ba03ec681860651190f7d7457804313ee
- 77469615c5f548063922b469a8c0a4116511395d013e5a798e123e9c119acc4b
- 7828e17e674507ab13dfd84b31b361fa19b9cb27ee130620ba9211feef746d31
- 8dbcde2a28a0b3de201214d7e3bd43acc97561924daa247c05c4b0536d42be85
- 901a43b42f997710147295a0625e20c935207f8c531daf5311449ec119a37dcc
- 94db6fa14b4e487dffba709b87e8a7e25483300ed409de243b19fff7cf2f0978
- 95cd48130247525d8a7e966bd3fa07e9d6c39ebbe3058eccc336f66bb8e3d1e
- 96bb418128deeb2b9d2e4b66b98cae07b238b326b6456cc9b86802e67c504a03
- 98cce1e69873de25e5139aa848f469bef2af345a8a49d15000b5b5e72b582896
- a1039de7ec690d64db9d7d91f3d777d308e49e958de4154aa0b62ded7820f1fe
- a785fc61fc4ff7cff0ddb540bf7ff12111ed0d6031f78f48387a6c16cb3c5451
- aa0083f662f055e8d911c5de3a8f3a31b3c84cacc7dccc30c98f2be14dba4102
- b58814fb3ce5a085014ee6e8d89f7cc1380b234b97170fd5f3398031281c6a77
- b6912c23cccc4b0964d55608916297f6978f0b38c80a4beac472004a786cef7
- bd3230e4ceaf32ad2248ab069b164bd2144401967ac69de0a4cd1734fe429d9c
- c25799facb3e788830bcf614f33411d3bcfc0edd4a2200e160b5eb4ce700039f
- c39fedb662259bd76b11616966c41ff1fbda58d9b129b9c1bd818700eea92b29
- ca8a00c9d36c64e5dcf562c7ae2b8df4bd6455fe0b41b32ee3a2a528ddc2d155
- ce379de03e35e0ea2c88744c29b9e2678165214065f9b957177002c6bbe69084
- d18369be4487d7cd0e4bd3dd0da720672e56e13ca43627305e26767e26925551
- d7745513034af14617436ad6b3fc125fd0343218411d0c79bda56b0dad86b2b
- d8ac0c08e4c698017558e532974cf749135d3d49757f05001e6127dc6e07cf17
- dccc9f008b42a04f7e69d3bbf7b5ce81e71308545d6176cc4763920a424e5ac1
- e5341edb7c039c456d46c39f194be86ce4b41725d7ad12d297d18aa99cddd675
- e88c41a6f769cd760e323b4f7c01835433cd4059cd59630cb1a9eb1181b350ed
- f13f9cef5cc020bf673c7f4e19c93c312a043867f46796a8f01927a9a14c2533
- f760bc16a585325ba9d74917f9e0994d3a4164c1141158c799b619d2c823e818

Additional Resources

- [Invoke-ClickFixAnalysis](#) – 0xVanasche
- [SSL Certificates for SHA1 Fingerprint ab92f731ab20774dfdb95664ee41a2fbafe2a284](#) – SSL blacklist by Abuse.ch
- [Technique: Hijack Execution Flow: DLL](#) – MITRE ATT&CK
- [Technique: Obfuscated Files or Information](#) – MITRE ATT&CK