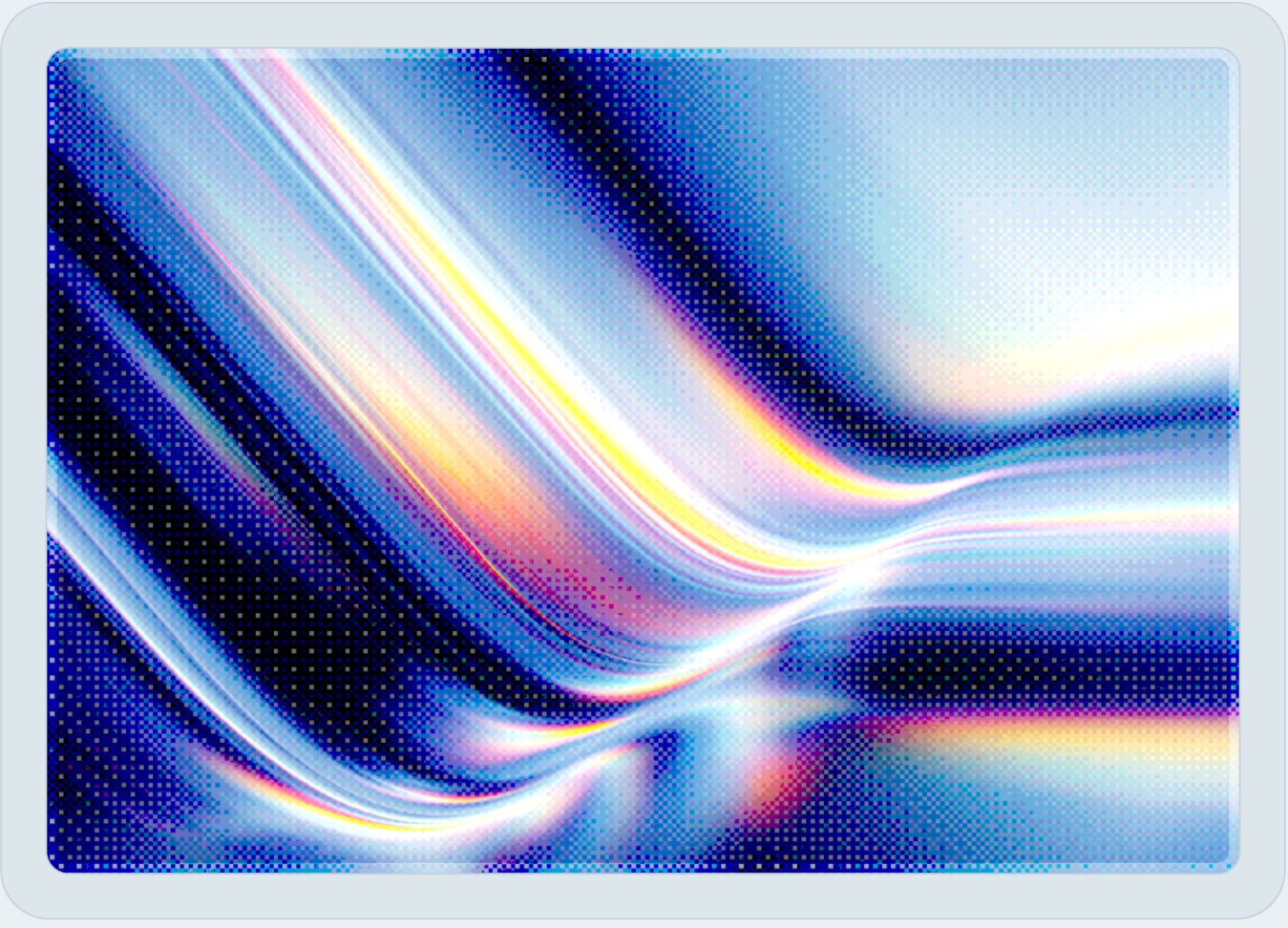
CSAI Foundation | Cloud Security Alliance

Yoba: Signed AI Vendor Binaries as Malware Sideloading Hosts

Security Implications and Guidance for Fake ChatGPT and Claude Desktop Installers

2026-11-01

AI-assisted Rapid Research



CSAI

cloud security alliance®

© 2026 Cloud Security Alliance. Some rights reserved.

You may download, store, display, view, print, redistribute, and link to this document in its original, unmodified form, provided that attribution to the Cloud Security Alliance is maintained and all trademark and copyright notices remain intact.

This document may not be modified or altered. You may quote portions of the document as permitted by the Fair Use provisions of the United States Copyright Act, provided that attribution is given to the Cloud Security Alliance.

This document may be shared on professional and social media platforms in its original form with attribution.

This document was generated with AI assistance and has not undergone official CSA review and approval processes.

Key Takeaways

Check Point Research (CPR) disclosed on October 27, 2026 a malware campaign it named Yoba, which distributes fake desktop applications for AI products that do not exist, among them "GPT-5.6 Sol," "GPT-6 Astra," and a "Claude Opus" desktop app, through GitHub repositories and GitHub Pages sites that ranked well in Google search results [1][2]. Each download archive pairs a legitimate executable digitally signed by OpenAI or Anthropic with a malicious `winmm.dll`. When the user runs the signed program, Windows loads the attacker's DLL from the same folder. This technique, DLL sideloading, is long established, as the title of CPR's technical report, "Tricks from the Past," acknowledges [2][3]. CPR states that its research gives no indication that OpenAI, Anthropic, ChatGPT, or Claude were compromised. The attackers borrowed the vendors' signatures and brand credibility rather than breaching them [1].

The signature is genuine, but the execution chain is not. A valid Authenticode signature vouches for one executable and says nothing about the DLLs that executable loads from its directory. As a result, controls that allowlist or reputation-score primarily by signer are likely to treat the initial process as trustworthy [1].

The payload is a miner, but the loader can deliver anything. The observed final payload is XMRig 6.26.0 mining over Tor. However, YobaLoader can execute any Windows executable or PowerShell script the operator stages, so CPR advises treating an infection as a broader compromise rather than a simple miner removal [1][2].

Command and control runs on legitimate services. The loader derives its domain from a Telegram channel's profile image, retrieves signed and encrypted tasking through Google DNS-over-HTTPS, and downloads payloads from Catbox file hosting [2]. Each of these hops blends into traffic that many enterprises already permit.

Demand for AI tools is the lure. The campaign follows earlier CSA-documented abuse of AI chatbot recommendations and AI vendor platforms as malware delivery channels [5][6]. It also builds on a June 2026 Yoba variant that ran under a process named `Cursor.exe` [2].

Priority actions. Organizations should restrict AI desktop software to vendor-controlled distribution channels and remove or tightly restrict standing local administrator rights on knowledge-worker and developer endpoints. They should also extend application control and detection from executables to DLL

loads from user-writable and `ProgramData` paths.

Background

AI Desktop Clients as a Lure Category

Major AI vendors have shipped native desktop clients and command-line agents over the past two years, and many users now expect a downloadable desktop application for major AI products. That expectation gives attackers a credible pretext. A user searching for the newest model release may have little basis for distinguishing a vendor-controlled download page from a polished GitHub Pages site carrying the vendor's branding, particularly when the advertised product name sounds like a plausible next version. Yoba exploited this gap with three invented products and repositories under accounts such as `gpt-5-6`, `gpt-6-astra`, and `opus-5-0` [2]. CPR found no evidence that the operator inflated stars or forks. It assessed that the sites' high Google ranking was consistent with SEO poisoning, and Help Net Security reported that the repositories still appeared among top search results at the time of disclosure [1][7].

The campaign also fits a broader pattern that CSA has tracked through 2026. In May, CSA documented a Microsoft-disclosed cryptojacking operation in which AI chatbot answers pointed users to attacker-controlled download domains [5]. In September, CSA analyzed threat actors abusing AI vendors' own domains and shared-conversation links to host malicious instructions [6]. Viewed alongside those cases, Yoba extends the pattern from the AI tool as recommender and the AI platform as host to the AI vendor's signed code as the execution vehicle. These are separate operations by unrelated actors, but together they show how many distinct routes now exist for turning interest in AI tools into malware delivery.

Campaign Timeline

CPR tracked the campaign for roughly two months before publishing, and its technical report reconstructs the operation from repository commit histories, release timestamps, and binary compile times. The table below summarizes the dated milestones in that report [2][7].

Date (2026)	Event
Early June	Earliest observed YobaDropper variant, sideloaded by a process named `Cursor.exe` and using the same Telegram channel and image

Date (2026)	Event
July 16	First fake repositories (GPT-5.6 Sol, including `gpt-5-6/sol-free-desktop`) created
August 30	`sol-setup.zip` (fake "GPT-5.6 Sol") uploaded; malicious `winmm.dll` compiled the same day
August 31	`Setup-Claude.zip` (fake "Claude Opus" desktop app) uploaded
September 4	`GPT-6-Astra-Setup.zip` uploaded
October 27	Check Point Research publishes blog and technical report

Security Analysis

The Yoba infection chain has three stages, and each one relies on trust that defenders normally extend to legitimate software or services. The first stage sideloads the dropper through a vendor-signed AI executable. The second establishes persistence by repeating the same trick with a Microsoft-signed system binary. The third retrieves tasking and payloads through public services that rarely appear on blocklists. The subsections below follow that sequence, then turn to the final payload and the detection opportunities each stage creates.

Stage One: Sideloading Through a Signed Vendor Executable

Every Yoba archive contains a file named `Setup.exe`. In the ChatGPT-themed packages it is a legitimate executable signed by OpenAI, and in the Claude-themed package it is a legitimate executable signed by Anthropic [2]. CPR's public report does not identify which OpenAI or Anthropic product each binary was taken from. Some secondary coverage has speculated about specific components, but this note does not repeat those claims because the primary source does not support them. Alongside the signed binary sits `winmm.dll`, a library name that normally refers to a Windows multimedia component. The malicious version exports 180 functions that mimic the genuine library. All but the entry

point are empty stubs that simply return [2]. CPR also found decoy files such as `vk_swiftshader.dll` and `vulkan-1.dll`, which make the folder resemble an ordinary Chromium- or Electron-based application install [2].

The attack depends on standard Windows loader behavior. When an application requests a DLL by name, the documented search order checks the application's own directory early, and Microsoft's documentation warns that an attacker who controls a searched directory can plant a malicious copy of a DLL there [4]. MITRE ATT&CK catalogs this as Hijack Execution Flow: DLL (T1574.001), which covers sideloading and search-order hijacking [3]. The practical consequence is that the process a user launches, and that an endpoint agent first evaluates, carries a trusted publisher signature. The malicious code then runs inside that process's address space. CPR draws the conclusion directly: a valid signature can establish the legitimacy of an individual executable, but detection needs to account for what legitimate applications load and execute [1]. Yash Kadakia, quoted by CSO Online, put it more briefly: "A signed executable is not the same thing as a trusted execution chain" [8].

CPR also reported that the ZIP archives were not password-protected. Commodity campaigns often use password protection to defeat gateway scanning, and CPR attributed its absence to the low detection rates the samples already achieved through anti-analysis techniques [2][7].

Anti-Analysis and Targeting Logic

The dropper, which CPR named YobaDropper, has several gating mechanisms aimed at sandboxes and analysts. It hashes the name of its host process and proceeds immediately only if that name is `Setup.exe`. Otherwise it enters a sleep loop lasting about 12.7 days, which outlasts typical automated detonation windows [2]. It checks the keyboard layout and exits on systems configured for Russian and several other CIS-region languages, including Belarusian, Kazakh, Armenian, and Uzbek [2][8]. Ukraine, historically included in such exclusion lists, is absent from this one, and CPR notes that the earliest variant excluded only Russian-language systems [2]. The dropper also resolves Windows APIs by hash rather than by name to frustrate static analysis. Finally, it requires administrator rights. If it is not elevated, it displays "Please run this program as Administrator." and exits [2].

The administrator requirement matters to defenders. A user who expects to install a desktop application often accepts an elevation prompt as routine, and the malware relies on that social expectation. On endpoints where users cannot elevate, CPR's description indicates the chain stops at this point [2]. Privilege management is therefore among the most effective controls against this specific campaign, provided users have no alternative path to elevation.

Stage Two: Persistence Through a Second Sideloading Pair

With elevated rights, YobaDropper creates `C:\ProgramData\WMProcess`, copies the legitimate Windows Media Player binary `WMPDMC.exe` from System32 into it, and writes a decrypted payload alongside it as `wmpdui.dll` [2]. It then registers a scheduled task that lists "Microsoft Corporation" as author, describes itself as collecting "device metadata for Windows Media Player," runs as Local System, and fires three minutes after each boot [2][7]. The persistence mechanism thus repeats the initial trick with a different trusted binary: a Microsoft-signed system executable runs from an unusual directory and sideloads the second-stage loader. CPR reported a string used as an XOR key in this stage. String artifacts of this kind can be planted to mislead attribution, and this note draws no conclusion from it. CPR itself states that the actor shows indicators suggesting Russian origins and appears to operate primarily in the UTC-4 time zone [2]; this note treats that assessment as tentative.

Stage Three: Command and Control Through Legitimate Services

The second-stage YobaLoader also verifies its host process name before running. It then fetches the public Telegram channel `t.me/yoba_fanclub`, extracts the profile image, and uses the image's final 50 bytes to seed a domain generation algorithm [2]. The algorithm produces domains of the form `adjective-noun-verb-NNN.dedyn.io`. The loader queries a TXT record for that domain through Google's DNS-over-HTTPS resolver, verifies the record with an Ed25519 signature against a hardcoded public key, and decrypts it with AES [2]. The decrypted record names an execution method and a Catbox download URL. The three methods are injection into `svchost.exe`, dropping and running an executable from `%TEMP%` after stripping its Mark-of-the-Web stream, and running a hidden PowerShell script with execution policy bypassed [2].

This design has defensive implications. In effect, rotating the operator's infrastructure only requires changing a Telegram profile picture, and every network hop uses commonly permitted services: Telegram, `dns.google` over HTTPS, a dynamic DNS provider, and a public file host [2]. The Ed25519 verification also means a defender who sinkholes a generated domain cannot push forged tasking to infected hosts. Network-layer detection therefore depends on behavioral context, such as a Windows Media Player binary making DNS-over-HTTPS requests, more than on the reputation of individual destinations.

Final Payload and Risk Posture

In the observed campaign the delivered payload was XMRig 6.26.0, injected into `svchost.exe` and configured to use 85% of CPU threads, to pause while the user is active, and to route mining traffic through bundled Tor and obfs4 bridge components [2]. CPR also observed conditional loading of the WinRing0 driver for processor optimization [2]. Cryptomining is often triaged as a nuisance, but CPR stresses that YobaLoader is a general-purpose staging mechanism and that later waves may deliver other malware [1][2]. A host that has run a Yoba package has granted Local System persistence to an operator who can push arbitrary code. If the victim used the machine for AI-assisted development, that operator could reach source code, API keys for model providers, and cloud credentials. Incident response should reflect that exposure.

Detection Opportunities

Because each stage abuses something trustworthy, effective detection looks at relationships between artifacts (which process loaded which DLL, from where, and what it then contacted) rather than at any single file or destination. The table below maps campaign behaviors to observable signals. The behaviors come from CPR's technical findings and the detection guidance reported by CSO Online [2][8]; the observable signals are CSA's operationalization of those reported behaviors.

Stage	Behavior	Observable Signal
Delivery	ZIP releases from newly created GitHub repos impersonating AI products	Downloads of AI-branded archives from `github.com` releases or `*.github.io` outside approved software channels
Initial execution	Vendor-signed `Setup.exe` loads `winmm.dll` from its own directory	Image-load events where a signed process loads an unsigned or mismatched-signer DLL from Downloads or a temp extraction path
Privilege	Dropper requires elevation	UAC elevation of an AI-branded installer from a user-writable directory

Stage	Behavior	Observable Signal
Persistence	`WMPDMC.exe` copied to `C:\ProgramData\WMProcess` with `wmpdui.dll`	System binaries executing outside System32; new scheduled task claiming Microsoft authorship running as SYSTEM
C2 discovery	Telegram fetch, then DoH TXT query to `dns.google` for a `*.dedyn.io` name	DoH traffic from non-browser processes; TXT lookups for dynamic DNS domains
Payload	Downloads from `files.catbox.moe`; injection into `svchost.exe`; Tor traffic	`svchost.exe` with sustained high CPU and outbound Tor or mining-pool connections; WinRing0 driver service creation

CPR's report also publishes SHA-1 hashes, repository and email indicators, mining endpoints, and two YARA rules (`yoba_dropper` and `yoba_loader`), which teams should import into their detection tooling [2].

Recommendations

The recommendations below move from confirming exposure, to closing the specific gaps Yoba relied on, to the longer-term question of how organizations and software publishers treat code-signing trust.

Immediate Actions

Security teams should first establish whether Yoba has reached their environment. They should hunt endpoint telemetry for the `C:\ProgramData\WMProcess` directory, for `WMPDMC.exe` executing from any path other than System32, and for scheduled tasks that claim Microsoft authorship with a Windows Media Player metadata description [2]. They should also search proxy and DNS logs for the indicators in CPR's report, including the `yoba_fanclub` Telegram channel, the fake repositories, `*.dedyn.io` lookups, and Catbox downloads, and should deploy the published YARA rules [2].

© 2026 Cloud Security Alliance. AI-assisted Rapid Research.

CPR advises treating an infection as a broader compromise rather than focusing only on removing the miner [1]. Given the loader's ability to run arbitrary code as Local System, CSA recommends treating confirmed hosts as fully compromised. Response should include rotating credentials and AI provider API keys that were accessible from the machine, then reimaging the host. In parallel, organizations should direct staff to obtain AI desktop clients only from the download pages linked on each vendor's official website or from an IT-approved catalog, consistent with CPR's advice to use vendor-controlled channels rather than search rankings, GitHub pages, or third-party links [1]. Staff should also be told that standalone desktop builds named for model versions, such as "GPT-6 Astra" or a "Claude Opus" desktop app, offered through third-party GitHub repositories are not official vendor releases.

Short-Term Mitigations

Over the following weeks, organizations should close the gaps the campaign relied on. Based on CPR's description of the dropper's elevation check, removing standing local administrator rights from user and developer endpoints would likely have stopped this chain unless users could obtain elevation through another path [2]. CPR and other commentators recommend restricting or reviewing such privileges, a step that also reduces exposure to many similar installers [1][8]. Application control policies should extend to DLL enforcement, not only executable allowlisting, and should block execution from user-writable locations where practical. MITRE lists application control that can block DLLs loaded by legitimate software as a primary mitigation for T1574.001 [3].

Detection and trust configuration need the same attention. Endpoint detection rules should flag signed binaries loading unsigned DLLs, or DLLs signed by a different publisher, from their application directory, and should flag system binaries running from copies outside System32 [8]. Teams should also examine the trust logic in their allowlisting and EDR exclusions, because any rule that trusts a process because of its publisher signature, regardless of path or loaded modules, is the condition this campaign exploited. Finally, IT should publish an approved catalog of AI desktop clients and coding agents with direct vendor download links. As CSO Online's expert source noted, making sanctioned tools easy to obtain reduces the incentive to search for them [8].

Strategic Considerations

Yoba shows that code-signing trust is transitive in practice even though it is not transitive in design. Any signed application that loads a library by bare name from its own directory can become a sideloading host, and the growing number of signed desktop clients and helper utilities from AI vendors adds to the pool of recognizable signed binaries an attacker can choose from. Software publishers, including AI vendors, can reduce the usefulness of their binaries as sideloading hosts by adopting the safer loading practices Microsoft documents, such as restricting DLL search paths with

`SetDefaultDllDirectories` and loading system libraries from System32 explicitly [4]. They can also monitor for their signed binaries appearing in malicious archives. Neither CPR nor the press coverage reports whether the abused binaries' signatures were revoked or whether the vendors were notified, and organizations should not assume revocation will occur [2][8]. CSO Online reported that OpenAI and Anthropic did not immediately respond to requests for comment [8].

The broader lesson is that brand demand for AI tooling is now a durable social engineering lever, and it operates across search engines, chatbot answers, AI vendor platforms, and code-hosting sites [5][6]. Governance programs should therefore treat AI client software as a managed software category with an owner, an approved source, and update monitoring, rather than as a personal productivity choice. Threat intelligence programs should likewise track AI-branded lures as a standing category. The June variant, which ran under a process named `Cursor.exe` and which secondary coverage describes as a fake Cursor AI coding-agent lure [2][7][8], suggests the operator may have used AI coding-tool branding before turning to ChatGPT and Claude.

CSA Resource Alignment

Yoba connects to several strands of CSA's recent research, and the alignment below shows where prior guidance applies directly and where this campaign extends it.

CSA's research note *When Trusted AI Platforms Become Malware Delivery Infrastructure* (September 2026) is the closest prior work. It analyzed threat actors abusing AI vendors' own domains and shared-conversation links, and recommended treating AI platform artifacts and chatbot-generated instructions as untrusted, externally influenced content [6]. Yoba extends that finding from AI-hosted content to AI vendors' signed code. Defenders should treat a vendor's signature as evidence about one file's origin, not as a judgment about the archive or folder it arrives in.

Poisoned AI Recommendations: Chatbots as Malware Delivery Vectors (May 2026) documented an earlier case in which AI-generated answers served as distribution channels for a cryptojacking campaign [5]. Read together, the three notes suggest that cryptomining operators have been early adopters of AI-themed lures. Because loaders like YobaLoader can deliver arbitrary payloads, the same infrastructure could be repurposed for higher-impact malware, which is consistent with CPR's advice to scope infections as broader compromises.

Poisoned Foundations: The AI Developer Toolchain Attack Surface (June 2026) addresses the developer-endpoint dimension of this risk [9]. The `Cursor.exe`-hosted June variant and the Claude- and ChatGPT-branded lures target the users most likely to hold model-provider API keys, source code

access, and cloud credentials. That paper's recommendations on governing developer tooling sources apply directly to AI desktop clients and coding agents.

At the control level, CSA's AI Controls Matrix (AICM) v1.1 provides the framework for the recommendations above [10]. Its Supply Chain Management, Transparency, and Accountability controls support approved-source requirements for AI client software. Its Threat and Vulnerability Management and Universal Endpoint Management domains cover application control, DLL-load monitoring, and privilege restriction on endpoints that run AI tools. Its Identity and Access Management controls support removing standing administrator rights and rotating credentials after a compromise.

References

[1] Check Point (Eli Smadja). "Yoba Malware Abuses Signed ChatGPT and Claude Binaries to Target AI Users." Check Point Blog, October 27, 2026.

[2] Check Point Research. "Tricks from the Past Deployed via Signed ChatGPT and Claude Binaries." Check Point Research, October 27, 2026.

[3] MITRE. "Hijack Execution Flow: DLL (T1574.001)." MITRE ATT&CK, accessed November 1, 2026.

[4] Microsoft. "Dynamic-link library search order." Microsoft Learn, accessed November 1, 2026.

[5] Cloud Security Alliance AI Safety Initiative. "Poisoned AI Recommendations: Chatbots as Malware Delivery Vectors." CSA Labs, May 27, 2026.

[6] Cloud Security Alliance AI Safety Initiative. "When Trusted AI Platforms Become Malware Delivery Infrastructure." CSA Labs, September 14, 2026.

[7] Help Net Security (Sinisa Markovic). "Yoba malware hides in fake desktop versions of ChatGPT and Claude." Help Net Security, October 29, 2026.

[8] CSO Online (Gyana Swain). "Fake ChatGPT, Claude apps spread malware via signed AI binaries." CSO Online, October 28, 2026.

[9] Cloud Security Alliance AI Safety Initiative. "Poisoned Foundations: The AI Developer Toolchain Attack Surface." CSA Labs, June 30, 2026.

[10] Cloud Security Alliance. "AI Controls Matrix (AICM) v1.1." Cloud Security Alliance, June 22, 2026.