

Fake AI, real malware: Attackers impersonating AI brands

 sophos.com/en-us/blog/fake-ai-real-malware-attackers-impersonating-ai-brands

August 19, 2026

A year of MDR casework shows attackers repeatedly exploiting demand for AI tools



Sophos X-Ops reviewed a year of Managed Detection and Response (MDR) cases tagged as ‘AI activity.’ Of the 34 cases that held up, nearly all were attackers creating fake versions of legitimate AI sites and software to infect users with malware.

Attackers are exploiting the surge in demand for AI software by faking the software itself: names users trust, like Claude, ChatGPT, and Copilot, become delivery vehicles for malware. For defenders that is good news – or at least, not all bad – because malware is a problem that existing controls are designed to address.

Methodology

This analysis draws on 12 months of Sophos MDR casework, from July 2, 2025 to June 29, 2026, alongside Counter Threat Unit (CTU) intelligence and SophosLabs research.

During that period, 86 MDR cases were tagged for AI involvement. We reviewed each case individually against [our AI threat taxonomy](#) and confirmed 34 as genuine adversarial AI activity.

Additionally, we identified a further four cases through analysts' investigations: a [Cursor-assisted detection-evasion case](#), a SonicWall SMA ransomware intrusion, a custom Slack-controlled RAT built with an AI coding agent, and a fake Claude site delivering a previously undocumented backdoor. We included these cases in our dataset, bringing the total to 38.

Of the rest of the cases, 25 were benign AI developer tooling tripping behavioral detections, and 27 included an AI keyword only incidentally.

Our taxonomy splits AI threats into two top-level categories: *malicious use of AI*, where the attacker wields AI as a capability, and *malicious targeting of AI*, where AI products, brands and ecosystems are abused. Each is divided further into sub-categories - from *AI-generated* and *AI-augmented* attacks through to *AI software impersonation* and *agent-initiated compromise*. 35 of the cases fall under *malicious targeting of AI*, and demonstrate tactics X-Ops has documented before from a fake Claude site that sideloaded a backdoor to ClickFix campaigns that bait macOS users with AI-branded lures. Where we did see attackers genuinely use AI as a capability, it was as an assistant with a human in control – most notably the Cursor detection-evasion case.

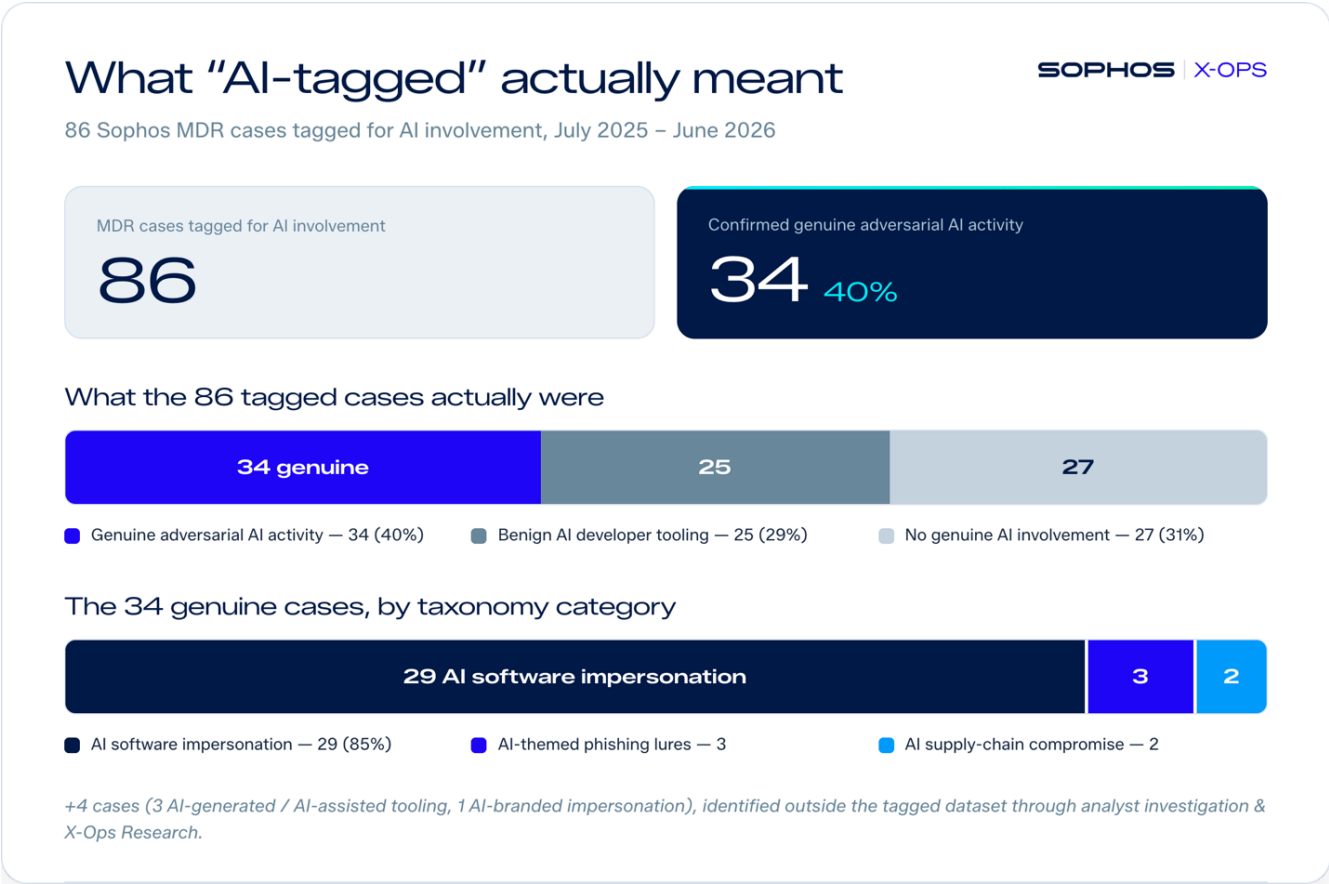


Figure 1: Overview of MDR cases with AI involvement

AI software impersonation: fake AI installers

AI software impersonation accounted for 30 of the 38 total cases, and fake AI installer campaigns were the largest cluster. The Claude brand was the most frequently abused lure, leveraged in 26 of the cases we reviewed.

Many of these incidents involved a technique known as ‘InstallFix,’ a variant of the [ClickFix](#) technique, in which the pretext is software installation. A typical attack chain involves a user searching for an AI coding tool, and landing on a typosquatted site, often through [a malicious ad or poisoned search results](#).

Whereas ClickFix attacks mimic an error or verification step, such as a fake CAPTCHA, an InstallFix page may present a polished, step-by-step installation guide. Both end with the user copying and running (often obfuscated) commands that ultimately result in a malware infection (see Figure 2 for an example).

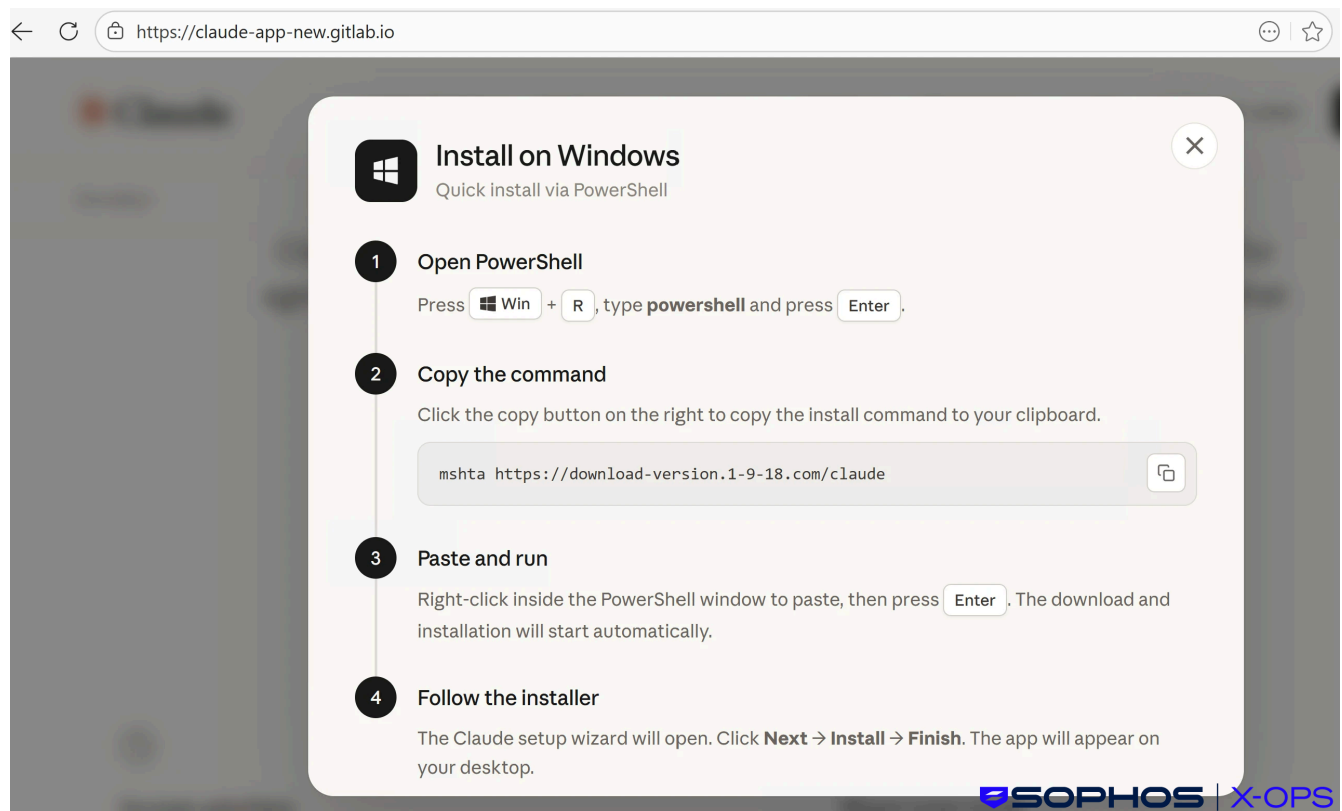


Figure 2: Fake Claude ‘InstallFix’ site. Users would be directed to this site via a malicious ad

In one case, a fake Claude site instructed the victim to run an **mshta** one-liner that pulled its payload from **download-version[.]1-9-18[.]com**. That payload was a Windows app package

installer named **claudio** or **claudio.msixbundle**. Next, an **irm <url> | iex** one-liner executed code in memory, conducting process-hollowing attempts against the browser.

In other, related cases we observed LummaStealer being delivered through the same AI-branded infrastructure, using the classic ClickFix method of a fake CAPTCHA page, rather than the InstallFix technique described above.

Variations on the theme included a **Claudio Setup.zip** archive that staged a malicious **libcef.dll**, and a repackaged **claudio.exe** that was actually a malware loader. Outside our MDR casework, the most developed example we observed was a fake Claudio site that delivered a DLL-sideload chain ending in [a previously undocumented backdoor](#) we dubbed 'Beagle.'

From a defensive perspective, in the impersonation cases we reviewed, the decisive protections were based on conventional delivery and payload behaviors, rather than AI-specific characteristics. The earliest and best defense remains unchanged: install AI tooling only from confirmed vendor domains.

AI software impersonation: malicious AI-themed browser extensions

We observed several browser extensions impersonating AI assistants – for instance, one was marketed as an “AI Sidebar with DeepSeek, ChatGPT, Claudio” – that functioned as infostealers and beacons to command-and-control (C2) infrastructure. Because users often grant extensions broad access to browser sessions and stored credentials, infostealers are an attractive proposition for threat actors in this context, enabling them to harvest sensitive data.

In a recent case Sophos investigated, four customers installed a fake Perplexity extension that functioned as a browser hijacker, intercepting searches, and exfiltrating browsing telemetry in real time through attacker-controlled infrastructure.

The extension was distributed through the Chrome Web store, redirected browser searches through **perplexity-ai[.]online**, and displayed a Tilda-hosted landing page to users after installation. The activity we saw correlated with [Microsoft's reporting on the campaign](#).

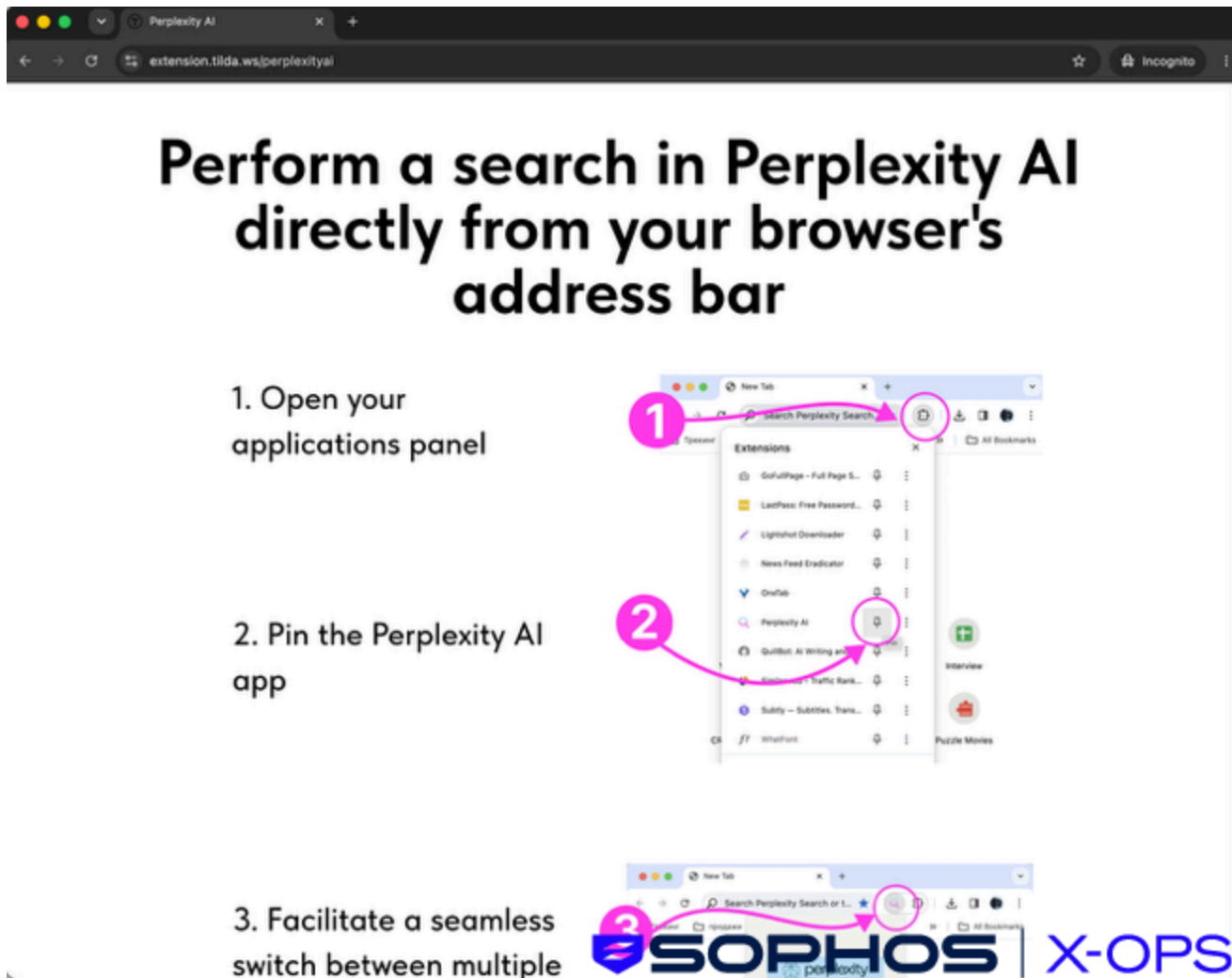


Figure 3: Fake Perplexity extension landing page shown after installation

Search for perplexity ai

4.7 ★ (67 ratings) ⓘ [Share](#)

Extension

Tools

10,000 users



Figure 4: The extension's Chrome Web Store listing. A 4.7-star rating across 67 reviews and a 10,000-user install count could give it the appearance of legitimacy

AI-themed social engineering

Attackers are also using AI branding as bait in social engineering flows. Unlike the installer campaigns above, no fake AI software waits at the end of these chains. Instead, the brand appears only in the lure. For example, we saw AI-themed domains used in ClickFix-style lures that walked victims into running `finger.exe` via the Windows Run dialog, and AI brands leveraged for credential phishing – including a fake Microsoft Copilot document share that funneled victims to an EvilProxy adversary-in-the-middle (AitM) kit, and campaigns impersonating the OpenAI brand to harvest logins.

These techniques are similar to a progression we noticed across [three ClickFix campaigns targeting macOS users](#), where lures shifted from fake-brand download sites to shared conversations on the legitimate ChatGPT platform – abusing the trust users may place in AI tools.

The cases we observed were distinct in their approaches. In one set, for example, the lure was built on the AI brand: victims reached a Claude-themed page through a malicious search ad and were persuaded to run a PowerShell one-liner.

In another set, the source was an ordinary compromised site with no AI angle at all – the AI theme appeared only in the command-and-control URL (for example, **code[.]verification-claude-cdn[.]beer** was served from a compromised advisory firm's site).

We also investigated a case using a [ChatGPT](#) share-link lure that matched [public reporting](#). Separately, we observed a case in which a ChatGPT Atlas user was tricked into running a `curl` command that installed an infostealer.

AI supply chain compromise: the on-ramp to agent-initiated attacks

We noted two cases in which threat actors were targeting the AI software supply chain. In the first, a poisoned [LiteLLM PyPI package](#) executed on an endpoint, and in the second, a **claude-mem** npm plugin installed through NPX fetched and ran code from a remote GitHub repository.

In both cases we could confirm the malicious package executed, but we could not confirm whether a human or an AI agent initiated the installation. The distinction matters because our taxonomy's *agent-initiated compromise* subcategory describes what happens when agents do the pulling: the window between 'published' and 'executing in an environment' collapses, with no human pausing to read a changelog.

As agentic development tools that fetch and run dependencies without a human in the loop become mainstream, this type of threat may become more common. Countering it calls for policy, dependency review, and secure-by-design controls.

Possible AI-generated attack tooling: a ransomware intrusion via a compromised SonicWall SMA appliance

One ransomware case carried more indicators of AI-generated tooling than any other in our dataset. The threat actor acquired initial access through a SonicWall SMA appliance, by exploiting the vulnerability chain disclosed as [CVE-2026-15409](#) and [CVE-2026-15410](#).

The threat actor connected to the victim network from a macOS device, which is itself unusual: in our casework, the operator machines we see in hands-on-keyboard intrusions are almost always Windows or Kali Linux.

Interestingly, we found a reference to the device hostname – **macdeMac-Studio** – in a Mandarin-language README file, in a Hugging Face repository for fine-tuning Qwen3.5. We cannot confirm this was actually the same device or operator, so we treated it only as a weak contextual signal.

训练走

```
python fine_tune_qwen35.py \  
  --dataset ./my_dataset.jsonl \  
  --output_dir ./qwen35_finetuned \  
  --steps 80 \  
  --learning_rate 1e-4 \  
  --target_modules all \  
  --dtype fp32
```

推理结果

```
(.venv) mac@macdeMac-Studio ai_nlp_q1 % python infer_qwen35.py \  
  --model_path ./qwen35_finetuned/merged_16bit \  
  --prompt '$解析下面句子的意思\n\n年轻人说 HHHACCC 是什么意思' \  
  --device mps \  
  --dtype fp32 \  
  --temperature 0 \  
  --max_new_tokens 64  
Loading model: ./qwen35_finetuned/merged_16bit  
Device: mps, dtype: torch.float32  
The fast path is not available because one of the required library is not installed  
Loading weights: 100%  
  
=== Answer ===  
是你有点可爱的可爱的意思
```

 SOPHOS | X-OPS

Figure 5: Hugging Face Qwen3.5 fine-tuning README referencing macdeMac-Studio

Abliterated variants of popular open-weight models are often published on model-hosting platforms, so a threat actor's path to an uncensored model can be as trivial as a download, rather than an engineering effort. The hostname link is admittedly circumstantial, but it could indicate that the operator's machine was connected to LLM fine-tuning tooling.

This model has previously been of interest to at least one threat actor: The Gentlemen ransomware group's [leaked internal chats](#) show an uncensored Qwen3.5 build being promoted to members in April 2026 (see Figure 5).

The shared repository was an ‘abliterated’ build ([abliteration](#) is a weight-level modification technique for open-weight models: it identifies the internal activation pattern associated with refusals and edits the model weights to remove it, stripping safety behavior without the cost of retraining or the fragility of a jailbreak prompt).

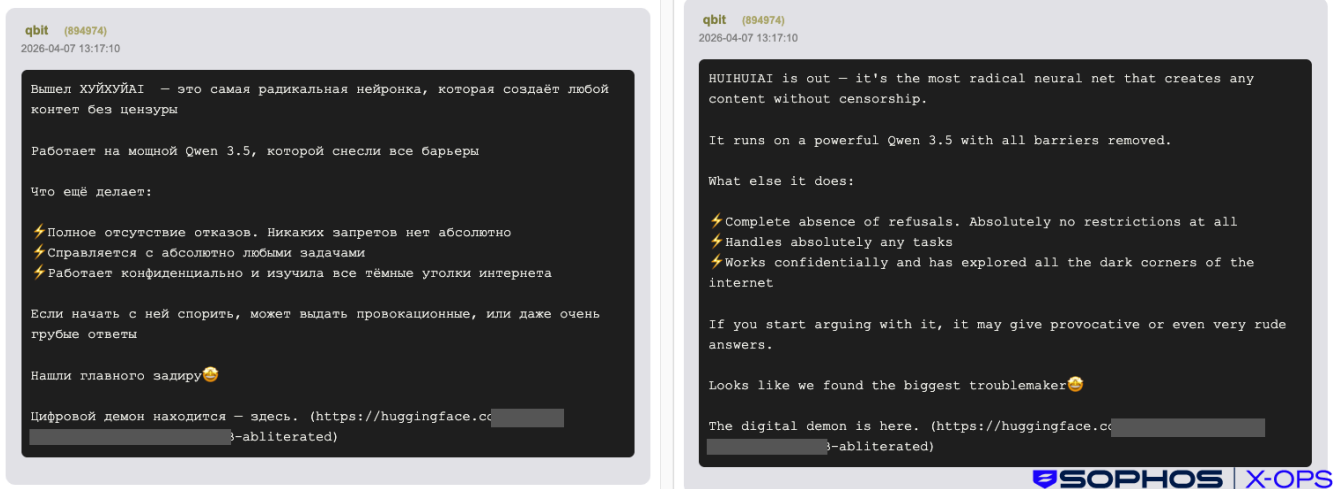


Figure 6: Uncensored Qwen3.5 build promoted in The Gentlemen ransomware group’s leaked internal chats, April 2026

We noted that a Base64-encoded PowerShell command used in the attack contained verbose comments narrating each step, and had a uniform, templated structure consistent with LLM-generated code.

In another Base64-encoded PowerShell script, used for internal network discovery, contained code comments in Mandarin. In our experience, comments of any kind are rare in obfuscated malicious PowerShell, and the use of Mandarin – also seen in the README file referenced earlier – provides another potential signal.

Later, ahead of the eventual ransomware deployment, the threat actor tested write access by creating a directory of dummy files (text, JSON, SQL dump, backup, document and email formats) and piping a directory listing to a temp file:

```
cmd.exe /C mkdir C:\test_enc & echo Hello World > C:\test_enc\test1.txt & echo Second file > C:\test_enc\test2.txt & echo {"data":"test"} > C:\test_enc\data.json & echo SQL dump > C:\test_enc\db.sql & echo backup data > C:\test_enc\backup.bak & echo This is a test document > C:\test_enc\readme.doc & echo email content > C:\test_enc\email.msg & dir C:\test_enc > C:\Windows\Temp\pMGiNNjc.tmp 2>&1
```

In our experience, human operators rarely probe write access this methodically; the command reads like generated tooling enumerating the filetypes a payload would target.

The ransomware that followed was notably crude. It encrypted its own extortion note, leaving the victim unable to read the ransom demand. In our assessment, this was in contrast with the competence demonstrated earlier in the attack, and may be consistent with a payload generated on demand.

We assess this as a possible AI-generated tooling case, with a human operator driving the intrusion – in which case it would fall under *malicious use of AI* in our taxonomy, under the *AI-generated* sub-category. However, the evidence is circumstantial and should not be treated as proof that AI wrote the payload.

AI-generated attack tooling: a custom Slack-controlled RAT built with an AI coding agent

In contrast to the above case, this incident didn't require any analytical assessment. We recovered the malware's source code, along with a commit history showing the developer building the tool with an AI coding agent.

A financial services organization was compromised through SQL injection against a bespoke PHP application running on a device not enrolled in MDR. The attacker deployed web shells and used them to run discovery commands, attempt credential access, and execute an off-the-shelf build of [Chisel](#), an open-source TCP tunnelling tool.

When the threat actor pivoted to MDR-protected devices, their activity triggered protections, and our incident response process took over.

On the protected devices, we identified a custom RAT using Slack as a C2 mechanism. Case artifacts allowed us to correlate the deployed RAT with a public GitHub repository under active development. The repository had two contributors: a human account we assess was associated with the threat actor, and a 'claude' account: Claude operating as a coding agent directed by a human (see Figure 7).

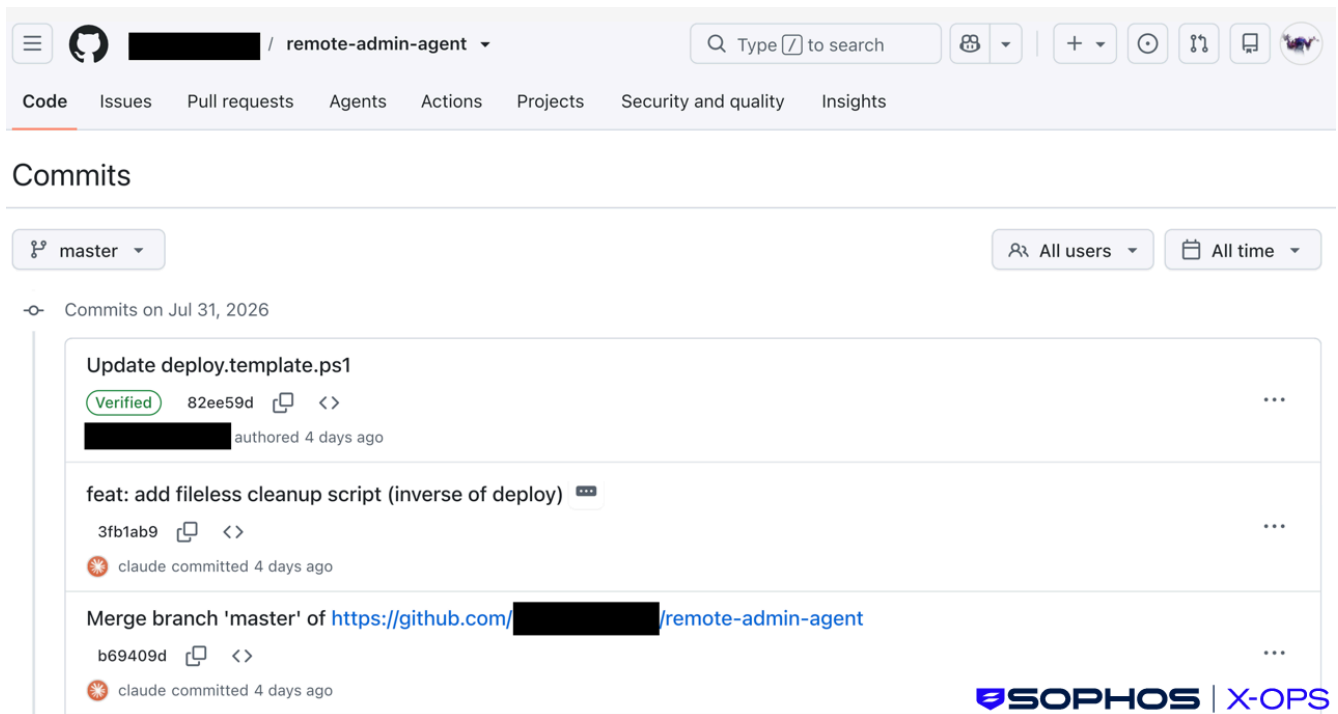


Figure 7: Commit history for the malware repository, showing the 'claude' account committing alongside the operator

The repository included documentation for [superpowers](#), an open-source agentic skills framework for AI coding agents, and a planning document that set out the malware's architecture: a Rust Windows binary that polls a Slack channel for host-tagged commands, which included options for running commands, opening a reverse shell, downloading files from URLs, downloading DPAPI-encrypted configurations, and deployment as a scheduled task.

The commit history tracked the build over several days - adding and later removing a reverse-shell capability, encrypting the deployed configuration, hardening the standard-user deployment path, and renaming the project's internal strings from 'rat-agent' to the blander 'svc.' The public README described the tool as a remote-administration agent for the author's own Windows machines (Figure 8).

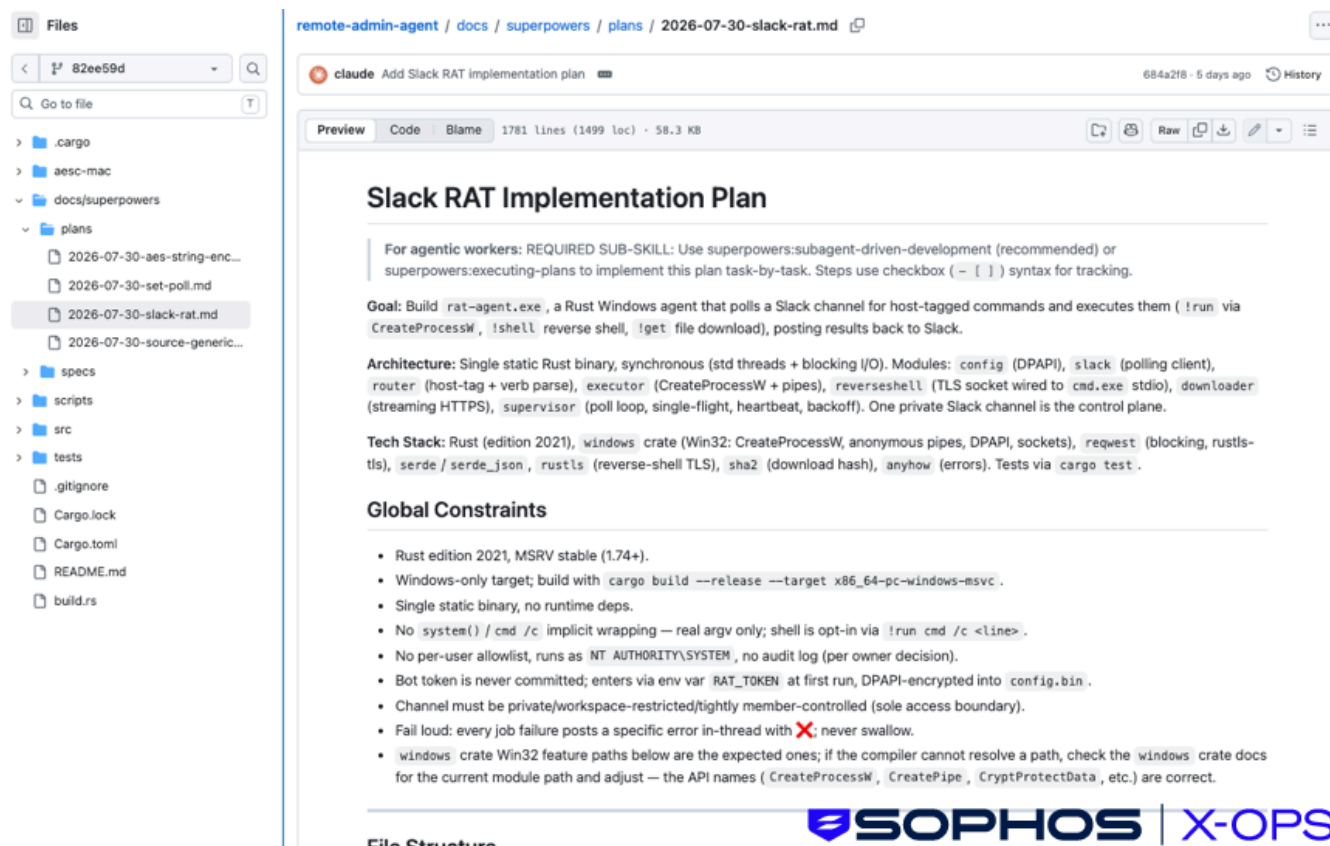


Figure 8: The malware's implementation plan from the repository, specifying the Slack-polling Rust agent and its use of the superpowers agentic framework

This is the clearest case of AI-generated attack tooling in our data, and the only one where we could trace the development directly, from the attacker's own source code and commit history: a human operator used an AI coding agent to produce a working custom RAT.

However, there's some cause for optimism here. The malware infected an unmonitored device and used a relatively unusual C2 channel. On a monitored endpoint, behavioral detection flagged it on arrival.

AI-augmented and AI-orchestrated: what we are not yet seeing

It's important to note what was absent from our telemetry and casework. The Sophos X-Ops taxonomy's *AI-augmented* sub-category covers malware that calls out to a model at runtime to generate its commands. We have not, as of this writing, confirmed a case in our own telemetry, although a prominent public example is [LAMEHUG](#), which CERT-UA attributed with medium confidence to APT28/UAC-0001 (IRON TWILIGHT) and described as using Qwen2.5-Coder-32B-Instruct via Hugging Face to generate commands at runtime.

The *AI-orchestrated* sub-category covers an agent driving the kill chain, with minimal human involvement. We found no confirmed instances of this, although the recent [OpenAI-](#)

[HuggingFace incident](#) and [Anthropic disclosures](#) suggest the capability exists.

Where we have seen attackers genuinely *use* AI as a capability, it sat at the lightest-touch end of the scale, in the *AI-generated* sub-category, with a human in control. In our analysis of a threat actor pointing Cursor at detection-evasion tooling, AI accelerated the development and testing of the tooling, but a human drove the workflow. The SonicWall SMA ransomware intrusion above fits the same sub-category: AI appears to have written the tooling, but a human ran the operation. That matches what we've found on cybercrime forums, where, as our [dark web reporting](#) has previously documented, there is undoubtedly enthusiasm about AI, but adoption and execution remain uneven and often clumsy.

However, it's worth noting two things. First, observed adoption trails available capability. [OpenAI's test agents](#) reportedly escaped a research sandbox through a zero-day and reached Hugging Face's production infrastructure; Anthropic's [review](#) of 141,006 evaluation runs found three cases where a model crossed a test boundary into real production systems.

As Sophos CISO Ross McKerchar wrote, these remain the best available evidence to date that a model can find a zero-day, chain it, and move through an organization's estate. Stealth is a potential constraint, however: in both cases, detections already in place caught the activity.

Second, the intrusions above are a few months old, a long interval in this field. Anthropic's three cases spanned three model generations that behaved differently on reaching real systems; whereas the oldest kept attacking, the newest stopped on its own. We expect threat actor behavior to follow the capability, and we are monitoring for it.

What defenders should do now

- Treat AI software like any other high-demand install target
- Restrict acquisition to verified vendor domains and block known typosquats
- Watch for the delivery techniques these campaigns rely on, because the AI branding does not change the behavior you detect on.
- Audit browser extensions, particularly AI-themed ones, against publisher reputation
- Bring AI dependencies into the same supply-chain governance as the rest of your software
- Close coverage gaps before anything else. In our clearest AI-generated case, the attacker's entry point was the one device not enrolled in MDR. Unmanaged and legacy systems are often where intrusions start

- Treat bespoke and legacy web applications as a priority attack surface. The same intrusion began with SQL injection against a custom PHP application, so application-layer testing, patching, and a web application firewall matter alongside endpoint controls.

We've labeled each case against our taxonomy because, in this fast-developing field, nuanced distinctions matter. An impersonation lure and an agent-initiated supply-chain compromise are both 'AI threats,' but they call for different defenses: download hygiene and existing endpoint detections for the first, dependency governance for the second. Treating them as one category tells you little useful about either. Our advice: when a case comes in, label it specifically, and let the data speak for itself.

Sophos countermeasures

The following Sophos countermeasures relate to the threats described in this article:

- Mal/HTMLGen-A
- Troj/WebShel-BU
- Troj/Beagldr-A
- ATK/DonutLdr-B
- OSX/InfoStl-FQ
- WIN-EVA-PRC-MSHTA-HTTP
- WIN-EXE-PSH-SUSP-IEX-1
- WIN-PROT-BEHAVIORAL-MALWARE-EXEC-19C-T1218-005
- WIN-PROT-BEHAVIORAL-MALWARE-C2-10A-T1071-001
- WIN-PROT-AMSI-AMSI-PROTECTION-AMSI-PSDLR-G
- WIN-EXE-PRC-POWERSHELL-WITH-BASE64-START-1
- Troj/RAT-NG
- Troj/WebShel-KB
- PHP/WebShell-CD
- ATK/Chisel-F / WIN-PROT-VDL-MALWARE-ATK-CHISEL-F
- WIN-CRD-PRC-PSQL-SUSP-VEEAMBACKUP-1

IOCs relating to the incidents described in this article are available in [our GitHub repository](#).