Credential-Implant Supply-Chain Poisoning Has Emerged as a Typical Paradigm for Attacking AI Developers

Correlation Analysis of the Malicious Claude Code Extension Distributed Through Cursor

Version 1.0 Compilation Date: November 1, 2026, Beijing Time

This report is a machine-translated version.

Summary

On October 29, 2026, a maliciously modified version of Anthropic.claude-code 2.1.312 and 2.1.313, distributed by the Cursor extension marketplace, was discovered. The official target platforms were confirmed to be Windows and macOS. The attack code entered the development terminal along with the extension and executed when the module was loaded. It attempted to write the attacker's SSH public key, enable the system SSH service, establish reverse SSH forwarding, and leave scripts, key files, and startup items outside the extension directory. Taking the malware package offline, uninstalling the extension, or shutting down the attack server at the time could not replace cleaning up the residual state of these targets. Antiy obtained the corresponding sample immediately and conducted analysis. It quickly released a risk warning and analysis report on the vertical response platform and verified that Antiy IEP effectively protect against and eliminate the sample in this incident. [1-3]

Antiy believes the key paradigm exemplified by this incident is: malicious code in the supply chain implants attacker credentials into the target environment, then uses legitimate remote services to create an independently retainable access point. More precisely, the target stores authorization records allowing login with a specific key; the attacker possesses the corresponding private key. The initial execution code is responsible for creating this authorization state, while the system's SSH service handles subsequent authentication and session execution. The access point can be retained independently of the original extension, but its availability still depends on the service, permissions, network path, and authentication policy.

This incident involves substantial malicious code, which Antiy believes cannot be categorized as "deploying configuration without deploying code", nor is it a fileless attack in the strict sense. Its special feature is that the access capabilities are distributed among the authorization files, legitimate services, reverse forwarding, and startup scripts, and do not require a dedicated remote control server to continuously bear all the capabilities. The fact that the payload did not download a separate malicious executable file does not mean that there was no script release, no disk write credentials, or no risk of subsequent interaction. [3]

By reviewing publicly available historical case materials, it was found that account or key implantation predates the era of large-scale models. There are records of decoy programs being written into system accounts as early as 1998; the tampered WordPress plugin in 2013, the leetlog with the latest announcement record in 2020, and the malicious npm package in 2024-2025 all demonstrate the relevant chain. The sidecar-mcp incident in 2026 and this incident bring this combination directly into the scenario of MCP configuration tools and AI programming extensions. Public

evidence supports classifying it as an attack paradigm worthy of key defense in the AI software toolchain, but it is not yet sufficient to quantify its speed of spread or prove that the early actual occurrence rate was low. [13-20]

Antiy engineers advise that the core principle for handling this incident is to simultaneously restore the credibility of the software source, the identity authorization, the remote access path, and the startup status. The investigation should differentiate between malicious version installation, code execution, authorization persistence, forwarding establishment, and the attacker's actual login; a final judgment cannot be made solely based on version number, public key comments, process existence, or script exit codes.

1. Incident Scope and Evidence Calibration

1.1 Incident Time and Impact Scope

The distribution window given by Cursor is from approximately 02:27 to 13:35 Pacific Daylight Time on October 29, 2026, which is equivalent to 09:27 to 20:35 UTC, and from 17:27 on October 29 to 04:35 on October 30, Beijing time, approximately 11 hours and 8 minutes. This window is the time when known malicious packages can be distributed, not the time when malicious code is actually executed on each host. Cached packages, offline installation, and delayed startup may cause execution to occur after the window. [1]

Table 1-1 Confirmation Scope of Affected Objects and Analysis Boundary Statement

Object	Confirmed Scope	Determination Boundaries
Distribution Channels	Tampered packages in the Cursor extension market	It cannot be expanded to include all VS Code extension markets being simultaneously compromised.
Extended Identifier	Anthropic.claude-code	The publisher string in the manifest is not equivalent to the publisher's authorization certificate.
Version	2.1.312 and 2.1.313	Version number is a screening criterion, but source, content, and hash are better able to distinguish malicious copies.
Officially Announced package Platform	darwin-arm64 and win32-x64	macOS and Windows are affected cannot be inferred solely from cross-platform code.
Linux	The payload includes Linux branches and corresponding IoC.	This pertains to coding skills and does not equate to confirmed Linux distributions or the number of victims.
Other oOfficial Channels	The Microsoft Marketplace, official npm packages, and Open VSX have not been confirmed to be affected by this incident.	Preserve the event and time frame; do not write that these channels are permanently secure.

1.2 Known Facts and Unresolved Issues

Antiy engineers analyzed the data and concluded that while the currently available information supports the identification of malicious packet distribution, payload behavior, and some residual data, it cannot independently determine the specific method by which attackers entered the Cursor distribution system, the organization to which they belonged, whether each step on each terminal was successful, or the final number of victims who successfully logged in. Market reconstruction and key rotation are necessary actions, but they are not proof that "other extensions are necessarily contaminated" or "a certain distribution key is necessarily leaked".

Some initial reports cited estimates of up to approximately 5,000 installers. The number of installers, downloads, code execution hosts, and successful remote access hosts are different metrics. This report does not use this number as the number of compromised hosts; the risk assessment of this incident should be based on technical capabilities and the organization's telemetry, rather than on a scale figure that has not been verified by a standardized approach. [4]

2. From Extension Distribution to Target Execution

2.1 How the Trust Boundary Was Breached?

Developers typically trust publishers, extension names, and normal update processes. In this case malicious content borrowed the legitimate extension identity and distribution channels, allowing users to receive attacker-modified packages during installation or automatic updates. Even if the upstream project itself is not compromised, the breach of downstream mirrors, aggregation marketplaces, or redistribution services is sufficient to disrupt content integrity from publisher to end user. [1,4]

Therefore, supply chain audits cannot simply check whether "extensions with the official names are installed". They should also record the actual download source, package content summary, platform architecture, version, download time, publisher verification information, and execution location. When multiple distribution channels exist, extensions with the same name and version cannot be automatically considered byte-consistent.

2.2 Trigger Timing of Malicious Code

Antiy engineers found that the attacker added a function before the normal extension code in extension.js. It is about 66 lines long and uses Node.js's process, file system, system information, HTTP and cryptographic interfaces. It runs when the extension module is loaded, without waiting for the user to start a Claude conversation. The sample does not rely on npm's preinstall or postinstall lifecycle hooks to be triggered. [3]

The statement "malicious code runs after installation" appearing in some articles may easily lead readers to believe that the installation action itself inevitably triggers execution. A more accurate description is: installation or updating allows a malicious extension to enter the environment, and subsequent loading of the extension module triggers the code. The timeline should include the startup of the editor, restoration of the workspace, or extension activation; the activation process for the specific host version should be based on the logs.

This also explains why prompt filtering, dialogue censorship, or tool call confirmation on the model side were not the direct points of attack in this incident. The malicious code executed within the extended host, outside the model's interaction flow. There is no evidence that this incident relied on prompt injection, model weight manipulation, or large model autonomous decision-making to accomplish the intrusion.

2.3 Composition of Delivered Payload and Endpoint Artifacts

Table 2-1 Description of Each Layered Component and Function of the Malicious Payload

Layer	Content in This Incident	Nature and Function
Executable Code within Extension	Pre-processed JavaScript logic	Data collection, file writing, system tool invocation, persistence configuration, and data return.
Data within Extension	Login authorization public key, another set of tunnel identity materials, address and path string	Driver authentication, connectivity, and deployment location selection
Endpoint Authorization State	Add a public key to authorized_keys	Add login privileges to the party holding the corresponding private key.

Endpoint Functional Scripts	.smt, smt.cmd, and the rollback startup script	Rebuild or use the tunnel identity file to initiate SSH connections in a loop.
Endpoint Startup Configurations	LaunchAgent, scheduled tasks, Startup files, Linux cron branches	Continue running the relevant scripts after login, startup, or process exit.
Legitimate System Components	ssh, autossh when available, sshd, shell, PowerShell	Provides connectivity, authentication, forwarding, and command execution capabilities.

The term "legitimate" here describes the origin and function of the component itself, but does not imply that its current call has been authorized by the user. Malicious intent is mainly reflected in who initiated the authorization change, who holds the key, the access destination, and whether it violates the user's intent.

3. Technical Breakdown of the Attack Chain

3.1 Environment Data Collection and Host Registration

Antiy's analysis revealed that the payload collected hostname, username, operating system, architecture, home directory, working directory, Git configuration email, and filtered up to 40 environment variables whose names contained KEY, TOKEN, SECRET, PASS, AWS, or GCP. The returned content was the variable names, not the values of these variables; therefore, it cannot be concluded that the corresponding API keys have been stolen. [2]

Variable names still have reconnaissance value. For example, they might reveal which cloud platform, deployment system, or credential type the endpoint uses. Usernames and home directories help map authorization files to target accounts; hostnames and Git email addresses help distinguish between multiple development endpoints. The working directory is just a path value and cannot be used alone to conclude that the payload has traversed the source code, read all projects, or stolen files.

Public IP addresses are obtained through ifconfig.me or api.ipify.org. These two sites are legitimate address lookup services, and access records alone are insufficient to constitute evidence of intrusion. The HTTP exfiltration target is port 8099/beacon of the attacker's address; requests to the same path as 127.0.0.1 are merely loopback attempts and cannot be considered as having been transmitted to the Internet. [1]

3.2 Write the Authorized Public Key

The payload appends the Ed25519 public key annotated as svc-telemetry to the current user's SSH license file. Unix-like branches also set permissions for the .ssh directory and license file. This changes which public keys the target service accepts for login authentication, rather than replacing the SSH authentication algorithm. [3]

Strictly speaking, the public key is not the login secret that the attacker needs to keep secret; what is truly kept secret is the corresponding private key. Referring to this act as "credential implantation" can be used to summarize the attack chain, but the technical description should be "implanting a public key authorization record that allows the attacker to control the identity." `svc-telemetry` is a public key annotation, not the creation of a local account named `svc-telemetry`. There is also no evidence in this incident that the initial payload necessarily created a new system account.

Typically, a process can modify the license file of its owner user without requiring administrator privileges. Whether it can further allow SSH login depends on service configuration, file format, permissions, account restrictions, and additional authentication requirements. The publicly available remediation script also warns that malicious public keys are appended without ensuring the file ends with a newline, potentially concatenating with existing content on the same line. Therefore, "seeing the string," "having a correctly formatted license record," and "SSH actually accepting the key" must be verified separately. [6,9]

3.3 Roles of the Two Sets of Keys

Antiy's analysis confirmed that the tunnel identity material embedded in the payload decoding uses temporary file extensions derived from the hostname and home directory. The SHA-256 digest of this private key file and the SSH

public key fingerprint are two different objects and cannot be interchanged or matched. Appendix A3 lists the file digest and fingerprint respectively. [2,5]

Table 3-1 Distribution and Usage of SSH Identity Materials in This Incident

Identity Materials	Location/Holder	Practical Purpose
Attacker's public key for logging into the target	Target authorized_keys	Tell the target SSH service which private key's corresponding identity is allowed to log in.
The private key corresponding to the above item	Attacker side	Used for authenticating the target account; it cannot be assumed that it has been written to the target by the payload.
Tunnel connection private key	The payload is embedded and decoded into a .stk_ file, as well as a .smtk or smtk file.	Allow the victim machine to connect to the attacker's server using the tunnel account.
Target SSH host key	System SSH service self-maintenance	This method of verifying server identity differs from the two methods mentioned above.

3.4 Activation of Legitimate SSH Services and Permission Constraints

Cursor attributes most access failures on macOS and Windows hosts to permission requirements and malicious code errors. However, this does not constitute a general rule that "only Cursor started in administrator mode is potentially risky" : if the target already has SSH running and the new public key is in the actual license file, the attack chain does not necessarily require the service to be successfully enabled. [1,9]

Windows also needs to distinguish between the authorization files for ordinary users and the C:\ProgramData\ssh\administrators_authorized_keys commonly used by administrator accounts. Files written to the user's directory are not guaranteed to be read by the administrator account's sshd configuration; file ACLs, account types, and the actual sshd_config should all be verified. The success or failure of Unix chmod cannot replace the Windows ACL judgment. [10] The specific code behavior and judgment comparison of different platforms are shown in the table below:

Table 3-2 Cross-Platform SSH Service Deployment Behaviors and Limitations for Determination

Platform	Actions Attempted by the Code	Key Success Conditions	Conclusions That Should Not Be Drawn
macOS	Check Remote Login and, if necessary, call systemsetup to enable it.	Management permissions, system settings, and account login permissions	Failure does not necessarily mean that the user's public key or LaunchAgent was not written.
Windows	PowerShell installs OpenSSH Server capabilities, configures and starts sshd	Feature installation is available, privilege escalation is enabled, and service and authentication configurations are allowed.	Requesting SYSTEM to run does not equate to having SYSTEM already been obtained.
Linux branch	Check sshd or ssh and try starting the service.	The corresponding service exists and the caller has the required permissions.	The existence of this branch does not mean that the Linux version was actually

			released.

3.5 How Reverse SSH Forwarding Resolves Reachability?

Antiy' analysis revealed that the key command-line segment of the payload is `ssh-N-R '*:0:localhost:22'`, which can be used to forward reverse requests in associated process telemetry.

Where -N indicates that remote commands should not be executed on the tunnel server; -R requests remote port forwarding; 0 requests that a listening port be allocated by the remote end; localhost:22 points to port 22 on the victim host side that initiated the connection. The asterisk indicates a request for the remote binding address, whether it is actually open to other hosts is still subject to server configuration constraints. [8,9]

The victim machine first actively connects to the attacker's SSH server. Subsequently, connections from remote endpoints to this forwarding port can reach the victim machine's SSH service along existing channels. This design reduces the impact of direct inbound connections being blocked by NAT or border policies, but it cannot bypass all network controls: outbound SSH must be allowed, the tunnel server must accept authentication and forwarding requests, and the local service must exist and be connectable.

Therefore, attackers do not need to know the victim's direct inbound address beforehand. Host backhaul provides location information, while reverse forwarding provides the actual access path; they solve different problems. A public IP address is not the same as a direct login address; multiple terminals may exist behind the same public network exit. Calling this behavior a "reverse shell" is misleading and confuses the two layers of capabilities. Initially, SSH reverse port forwarding is established; an interactive shell requires subsequent authentication through the target SSH service. The successful existence of the outer tunnel process does not equate to successful inner login, much less administrator privileges.

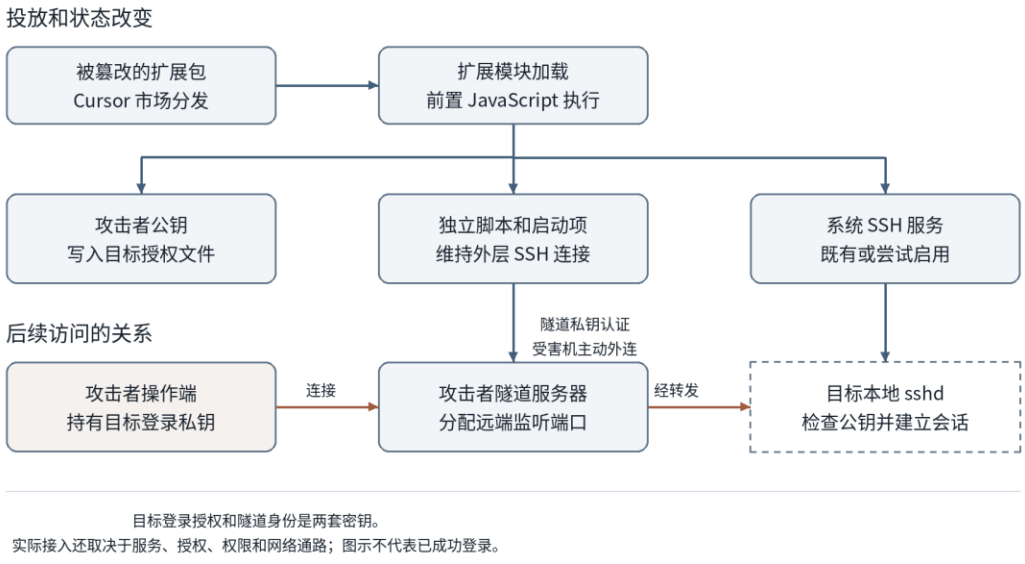


Figure 3-1 SSH Backdoor Attack Chain Diagram of the Malicious Claude Code Extension

3.6 Standalone Scripts and Startup Persistence

macOS uses the user LaunchAgent, Windows attempts to create a SysMon scheduled task running as SYSTEM and provides a Startup script backup; the Linux code branch uses the user crontab's @reboot. The scripts and tunnel keys

are located outside the extension directory, so uninstalling the malicious extension does not automatically remove the backdoor-related configurations, and access permissions are retained.

The Unix branch's .smt can rebuild the persistent .smtk identity file at runtime and then perform a linking loop. Deleting only a key file but retaining its generation script and startup entries may cause the file to reappear. The Windows branch writes the identity file and batch script to disk separately. [2]

"Retry approximately every 60 seconds" should be understood as waiting for reconnection after connection exit or failure, rather than necessarily establishing a new parallel tunnel every 60 seconds, nor is it proven that host details are reported every 60 seconds. The continuously running startup mechanism, tunnel loop, and initial delay beacon belong to different lifecycles. [2]

3.7 Network Backhaul and Follow-up Payload Boundaries

Antiy's analysis revealed a delayed callback attempt to send JSON to loopback and external addresses. Initially, the HTTP responses were not interpreted as code or configuration, nor were any new malicious scripts downloaded through these requests; curl's known purpose is to obtain public IP text. The standalone scripts and keys were generated or decoded from existing content within the extension. [3]

This conclusion is limited to the initial payload analyzed. If subsequent SSH logins are successful, operators can execute commands or transfer files, and these actions may not be reflected in the original extension.js. Conversely, the existence of this capability does not necessarily mean that source code theft, token theft, lateral movement, or further supply chain contamination has occurred.

4. Key Paradigm of Independent Access via Credential Implantation

4.1 How a Single Code Execution Turns into Persistent Authorization

This incident is the execution of code from the supply chain, altering the long-term identity and authorization status of the target environment. The development tool handles entry and execution, the authorization file stores the attacker's identity, the legitimate service handles authentication, and the script and startup items maintain the network path. Each component can exist independently; deleting one does not necessarily eliminate the others.

The statement "the pre-installed backdoor is not in the delivery package" needs to be interpreted judiciously. If it means "the SSH service that ultimately provides remote login capabilities originally belonged to the target system or an installable component of the system", this generalization is valid. However, if it means "the delivery package contains no malicious code, no keys, and no backdoor preparation logic", then it contradicts the evidence. A more appropriate statement would be: the remote session capability is provided by a legitimate service, and the malicious delivery package is responsible for implanting authorization and persisting the connection.

4.2 Analysis of Three Independent Attributes of Authorization Residues

The persistent credentials backdoor in this incident means that when the attacking server goes offline, the return and forwarding to that server will be interrupted, but the public key will not be revoked from the victim host. If the system's SSH is reachable via other network paths, the residual authorization may still be exploitable; if the local service is not enabled and there is no reachable path, the authorization does not constitute a usable entry point temporarily, but it should still be revoked. "Residual authorization" should not be written as "Unconditionally accessible for remote login", nor should "Currently unreachable" be written as "No residual risk".

Table 4-1 Meanings and Dependencies of the Three Independent Attributes in This Incident

Independence	Practical Implication in This Incident	Remaining Dependencies
Independence from the original extension	Authorized entries and external scripts can be retained after uninstallation.	The file was not deleted, and the startup configuration is still effective.
Independence on dedicated remote control programs	Authentication and sessions can be hosted by the system's SSH.	Legitimate services are available, and account policies permit.
Independence from attack infrastructure	The authorization record can continue to exist independently of the server at that time.	The original reverse tunnel still relies on the corresponding server; other access methods require alternative reachable paths.

4.3 Condition Model and Risk Stratification

Based on the characteristics of this incident, we break down the conditions for establishing a valid credential-based persistent entry point. A usable credential-based entry point requires at least: the attacker controls the authentication materials, the target accepts the corresponding authorization, legitimate services are available, and a reachable path exists. Persistent access across restarts also requires that the relevant authorization, services, and necessary network mechanisms remain effective after a restart. The absence of any one of these factors may block current access, but this does not mean that other implants are not present.

Table 4-2 Investigation Status, Analytical Conclusions and Response Priorities

Investigation Status	Derived Conclusion	Response Priority
Only relevant versions or caches were found.	There are clues to the exposure.	Verification of source, hash, module loading and installation logs
Payload execution confirmed, but service or forwarding failed.	Local malicious execution has occurred.	Continue to clean up licenses, scripts, startup items, and assess information exposure.
Found a valid malicious authorization with SSH access.	Possesses potential access conditions	Immediately isolate and verify authentication records; do not wait until further malicious files are discovered.
Forwarding confirmed successful	A connection path has been formed.	Continue to verify target key authentication and session execution.
Confirm attacker login or subsequent commands	Interactive intrusion has occurred.	Expanding the investigation of credentials, source code, build environment, and related assets.

4.4 Implications for Malware Detection

For this type of threat, Antiy believes static scanning is suitable for identifying known extended files and scripts, but it's also necessary to monitor "who changed the logonable identity". Differences in the content of authorization files, key fingerprints, service configuration changes, task actions, and network forwarding parameters may be more indicative of risk than whether a malicious binary is still on the disk. Legitimate SSH, PowerShell, or launchctl calls should not be used as a sole criterion for malice. A higher-value combination is: editor extensions deriving system commands from the host, followed by unauthorized public keys, unfamiliar external SSH connections, and startup items pointing to user-writable scripts. Each signal may be normal on its own; only the associated identity and control relationships constitute complete evidence.

5. Historical Precedents and Evolution of Combinations within AI Toolchains

5.1 Grouping Historical Cases by Strength of Commonality

This incident differs from previous analyses by Antiy regarding traditional npm ecosystem credential implantation poisoning attacks and latent pollution attacks on Chinese open-source AI model code repositories. While past malicious packages were largely confined to the package manager's lifecycle, this attack leverages AI programming to extend market distribution. Once the payload is implanted, the entire access point becomes permanently independent of the extension itself, achieving persistent persistence. Furthermore, it targets high-value development environments, significantly amplifying the threat's impact. Antiy has also compiled a list of similar historical cases of identity configuration implantation poisoning, as shown in the table below.

Table 5-1 Comparison of Historical Supply-Chain Poisoning Cases of the Same Type

Time and Cases	Delivery and Authorization Behavior	Commonalities with This Incident	Critical Distinctions Not to Be Overlooked
1998 sshdwarez or sshdexp	The decoy security program, when executed as root, adds an account to /etc/passwd and sends a reply.	Code execution translates into account status change	There was no proof that the legitimate official supply chain had been compromised, and the announcement did not restore the complete access chain. [13]
A tampered WordPress plugin was distributed in 2013 and disclosed in 2014.	Third-party plugin download channels; return site URL, triggering the creation of an administrator.	By embedding an identity through software, and then using normal login...	It is a third-party redistribution pollution; the service is WordPress not SSH [14]
2016 CCTM Malicious Update	Official plugin directory updated; subsequent scripts will establish a support administrator.	Official distribution chain and account integration appear simultaneously	It involves downloading backdoors, web shells, and stealing passwords, making it a hybrid incident. [15]
LeetLog will issue an announcement no later than 2020.	Write a hard-coded SSH public key using npm 0.1.2 and 0.1.3	The public key authorization implantation mechanism is directly the same.	2020, but the initial deployment date, service status, and scale of damage remain unclear. [16]
2024 ethers-mew	Impersonating an Ethers branch and writing the root authorized public key.	Development package execution and SSH identity implantation	Simultaneously stealing Ethereum private keys requires write permissions to the root file. [17]
2024 streamserch and other packages	npm spelling spoofing, embedding authorized public keys and returning identity and public IP address.	Packet delivery, host location, and legitimate SSH entry point combination	The lack of evidence that AI was involved in the attack does not equate to a breach of the legitimate dependency maintainer. [18]
Packages such as node-	This fake Telegram repository	The public key is retained	It belongs to the developer

telegram-utils (2025)	writes two public keys and sends them back to the address.	after uninstallation and will be used for subsequent system SSH reliance.	package ecosystem; there is no evidence of a direct link to AI.[19]
2026 Sidecar-MCP Announcement	MCP configures CLI to write the authorized public key; try enabling macOS Remote Login.	AI toolchain entry point, authorized and retained independently of the package.	The CLI must be run ; it cannot be mistakenly written as "triggering upon installation" or "the official MCP project is faulty" [20].
2026 Cursor event	Tampering with extensions in the market, writing authorization, startup scripts, and reverse forwarding.	Directly combine supply chain delivery with voucher-based continuous entry points	Local SSH has the necessary conditions for success: permissions and access are all met [1-3].

The table above summarizes several historical cases of identity implantation through the software supply chain, comparing their deployment methods and authorization mechanisms in chronological order, extracting commonalities with the current Cursor malicious extension incident, and highlighting key differences to avoid simple analogies. The cases cover different stages, including traditional SSH account poisoning, npm ecosystem spelling spoofing, and AI toolchain MCP scenario poisoning, focusing on key elements such as distribution channels, persistence logic, whether system SSH is reused, and whether it is associated with the AI ecosystem. Two cases require further clarification: The 2013 WordPress tampering plugin case material comes from a 2014 Sucuri investigation report. The report records that tampering plugins such as Restrict Content Pro and Flat Skin Pack first entered third-party download sites in June 2013; the SEOPressor malicious payload would send back the site address, create a new administrator account with a preset password, and reuse the website backend 's normal login channel. This time only represents the initial sample inflow time and cannot be uniformly identified as the distribution time of all samples. [14] The sidecar-mcp incident was reported by Amazon Inspector and included in OpenSSF and OSV, with an announcement date of July 10, 2026. The payload used the MCP configuration process to write the SSH public key and attempted to enable remote login on macOS. The authorization configuration was still retained after the package was uninstalled. This case only proves that the AI tool ecosystem has become a new scenario for supply chain poisoning, and there is no evidence that the model is generated autonomously or executes the attack strategy. [20]

5.2 Risks of the Classic Credential Backdoor Implantation Paradigm

As mentioned above, embedding accounts, public keys, or other authorized materials is a long-standing persistence method; combining it with the software supply chain is not a new phenomenon of the AI era. Compared to cases where accounts are added after exploiting common vulnerabilities, the complete chain of "supply chain deployment—embedding new authorizations—forming an independent entry point through legitimate services" appears as scattered cases spanning multiple years in the early materials retrieved in this study. This observation describes the visibility of publicly available evidence and cannot be directly equated with the actual historical incidence rate.

Materials from 2024–2026 show that this combination is repeatedly used in developer software packages and has appeared in MCP configuration tools and AI programming extensions. This is sufficient to support its elevation to a typical threat model in the AI software toolchain, and its inclusion in detection, emergency response, and supply

chain audit design; it is inappropriate to describe it as a "rapidly spreading" or "primarily AI-driven" threat that has been statistically proven.

This combination has existed in the past, but public records are scattered. As development tools, package managers and AI plugin ecosystems are connected to more high-value work environments, this chain of changing long-term licensing status with a one-time deployment has gained wider applicability and may begin to become a risk that needs to be guarded against according to typical paradigms.

5.3 Why AI Toolchains Raise the Defence Priority for This Combination?

Antiy argues that, firstly, AI programming tools are often located adjacent to source code workspaces, terminals, package installers, and remote development environments. Contaminated extensions or configuration tools are more likely to be in process contexts capable of writing to user directories, accessing SSH materials, and invoking system commands. This discussion focuses on the opportunities presented by the deployment architecture, not the claim that all AI tools have the same permissions. Secondly, plugins, MCP servers, and dependencies can enter the development environment via updates, recommendations, or a single configuration command. The increasing number of new access points makes it easier to bury identity authorization changes within the normal task of "configuring development tools". Sidecar-MCP is a direct precedent for this task spoofing; this incident highlights the risks of distorted market distribution content. [1,20] Thirdly, development terminals simultaneously connect to personal workspaces, organizational repositories, build platforms, and cloud resources. Even with only ordinary user privileges, attackers may access data that the user is entitled to read. This risk stems from the actual coverage of identity permissions, without needing to prove administrator privilege escalation or automatically writing the accessible scope as the already compromised scope. Finally, the handling of malicious extensions often revolves around version rollbacks and uninstallation, while the lifecycle of credential-based entry points follows system authorization. The inconsistency between these two lifecycles can create a gap between "software updated" and "terminal authorization restored to trustworthiness". The novelty of the AI era lies primarily in toolchain combinations and asset aggregation, not in the SSH key authentication mechanism itself.

5.4 Tracking Evolutionary Trends of This Attack Pattern

Antiy believes that subsequent observations should be based on independent attack activities rather than packet counts, and should simultaneously record four types of entry points: compromised official release chains, third-party tampering and distribution, spelling imitation, and pure decoy tools. Each case should also be marked with whether a new identity was implanted, whether it relied on legitimate services, whether it was retained after unpacking, whether there were other remote control payloads, and whether the AI was the target scenario, brand decoy, development tool, or attack execution entity.

Under the same monitoring coverage, the proportion of this tactical combination can be compared by using the annual number of supply chain incidents or exposed assets as the denominator. Treating six malicious packets from the same source as six independent activities, or treating reports with AI detection product names as AI attacks, will amplify trends.

6. Investigation, Remediation and Recovery Validation

6.1 Preserve Evidence Before Remediation

Antiy recommends isolating terminals found to have executed or implanted malware on the network, while retaining necessary controlled forensic capabilities. Record the terminal, user, installation source, extension path, version, package digest, relevant timestamps, and time zone; save copies of malicious files, authorization files, startup configurations, and process network information. Original authorization files and tunnel identity files should be managed as sensitive evidence; do not copy private key contents into support tickets or public reports.

Cleaning will alter the timeline, file content, and process status; therefore, necessary forensic evidence should be collected first. If business risks require immediate connection blocking, isolation can be implemented first, followed by local evidence collection; there is no need to keep the attack channel online while waiting for complete evidence collection.

6.2 Move from Single-Indicator Checks to Multi-evidence Judgement

Cursor officially recommends using `grep` or `Select-String` to search for `svc-telemetry`, outputting "machine not impacted" when a match is found. Antiy believes this conclusion is too absolute. A match only indicates that the string was not found within the scope of the successful read; other possibilities include a non-existent file, insufficient permissions, different users, deleted entries, or only partial execution of malicious code. The checking program should output "Hit", "Missing", and "Failed to Check" separately, rather than combining the latter two into a security conclusion. It is recommended to first confirm by public key subject or fingerprint, and then combine with comments. Comments can be modified or forged; authorization files may also contain line-start restriction options, so directly matching entire lines of text is not always reliable. The search scope includes all actually affected users, Windows administrator authorization files, and other AuthorizedKeysFile paths actually configured in sshd. See Appendix C for the initial quick search command.

6.3 Investigation Sequence and Evidence to Collect

An attacker's remote connection, tunneling through a reverse tunnel to the local SSHD, may appear as originating from a loopback address on the target side. Therefore, it's insufficient to simply search the attacker's public IP address in the authentication logs; key fingerprints, session timestamps, usernames, tunneling processes, and subsequent commands should also be considered. This warning is based on inferences about the forwarding mechanism and does not imply that publicly available documentation provides detailed login logs for this event. It is recommended to refer to the table below and Appendix A for investigation.

Table 6-2 Investigation Targets, Validation Items and Questions to Answer

Investigation Target	Verification Content	Question to Answer
Distribution and extended records	Actual source, hash, version, load time, update cache	Did you obtain and load a malicious copy?
Authorization documents	Add key, fingerprint, format, permissions, and actual read path	Has a valid attacker authorization been formed?
SSH service	Installation time, startup method, monitoring,	Whether it was originally enabled, whether it

	login scope, and authentication policy	has been modified, and whether the implanted identity was accepted.
Processes and networks	Parent and child processes, identity file path, forwarding parameters, destination address	Should the tunnel be started? Should the connection be maintained?
Startup item	Name, Action, Runner Identity, File Content	Can it continue running across startups or logins?
Authentication and session	Key acceptance record, username, session, subsequent commands	Does actual login and interaction exist?
Projects and remote systems	Source code reading, token usage, repository or build changes	Whether to extend the impact to other assets

6.4 Remediation Scope and Boundaries for Official Cleanup Scripts

Antiy recommends that cleanup should cover malicious extended copies and caches, unauthorized public keys, running tunnels and retry processes, standalone scripts, tunnel identity files, and startup items. Startup actions must be verified to avoid mistakenly deleting legitimate software based solely on names like SysMon. Only materials confirmed to belong to attackers should be removed from the license file; retain legitimate public keys and necessary file permissions.

Cursor provides fix scripts for macOS and Windows. macOS scripts must be run as the affected user, not root; in multi-user environments, all affected users must be covered. Windows scripts require elevated privileges, comments must be closed before running, and the user configuration directory must be traversed. Scripts should be obtained from the official address linked in the announcement, reviewed, and executed within a controlled change management process, with the version used and output documented. [6,7]

The review of the publicly available script text also revealed two limitations. First, Windows scripts primarily process authorization files in user directories, and this does not presume that C:\ProgramData\ssh\administrators_authorized_keys or custom authorization paths have also been cleaned up. Second, the script did not fully restore the SSH service installation, startup method, and remote login settings. Therefore, after successful execution, it is still necessary to compare it with the original configuration baseline. [6,7]

A non-zero exit code for a script indicates problems it has detected; a zero exit code only indicates that the script version did not report any remnants, and does not prove that there were no missed paths, other payloads, or subsequent intrusions. The original text "If there are still remnants, the script will exit with a non-zero status" should be corrected to "If the script's overridden checks find uncleaned items or corresponding errors, it can return to a non-zero status".

6.5 Credential Rotation and System Rebuild

Meanwhile, Antiy recommends assessing the user's accessible sessions, repository tokens, cloud keys, signature materials, and other secrets for terminals confirmed to be executing payloads. Initial beacons containing only environment variable names do not constitute grounds for "no rotation required"; if actual access cannot be ruled out, conservative rotation and session revocation should be implemented based on asset sensitivity and investigation completeness. Rotation operations should be performed on trusted terminals to prevent new credentials from falling into the wrong hands again.

If attacker login, additional persistence, unexplained system modifications, or gaps in evidence make it impossible to restore the host's trusted state, it should be rebuilt from a trusted image. Before rebuilding and restoring, verify backups, extended caches, and development configurations to prevent the restoration of malicious authorizations or startup scripts.

6.6 System ecovery Verification

Antiy advises that restoring an attacked system requires verifying at least the following: malicious extensions are no longer loaded; attacker keys are not present in any authorized locations; startup items and running processes no longer point to malicious scripts; SSH service and network policies conform to the original baseline; related artifacts are no longer generated after logout or restart; associated repositories, cloud services, and sessions have been properly handled according to risk management procedures. Restarting only verifies partial restoration of persistence and does not prove that no intrusion occurred in the past.

7. Defense Construction for Software Toolchains

This incident serves as a further reminder that distribution platforms should verify the correspondence between content and publisher authorization, and retain traceable records of version, platform, hash, and source. For mirror or aggregated marketplaces, abnormal differences between them and authoritative upstream sources should be detected; alerts should be issued for versions not appearing upstream, changes to the same version's content, and changes to signature relationships. Update buffer periods and small-scale verification can reduce the risk of rapid spread, but cannot guarantee that all malicious packets will be detected after a delay.

Antiy engineers recommend that endpoints include authorization files, remote services, and startup status in joint monitoring. Detection rules should record the process initiating the change, the target user, key fingerprint, differences in service status before and after, and startup actions. The developer tool host writing `authorized_keys` and connecting to a new SSH destination address is a more specific investigative clue than simply "using SSH". The network side should restrict remote management connections from developer terminals to unauthorized destinations, correlating long-term outbound SSH connections, abnormal forwarding parameters, and changes in terminal authorization. Blocking only the current IP address can cut off known pathways, but it cannot replace revoking authorization, nor can it cover the same method after address relocation. Legitimate public IP lookup services should be used as contextual signals and should not be directly listed as globally malicious domains.

Meanwhile, current traditional credential security perspectives have significant limitations. Some organizations' identity governance awareness remains stuck in the traditional approach of privileged account management. Security product design also largely focuses on scenarios of credential theft and leakage, failing to build a complete detection, monitoring, and defense loop around "malicious implantation of new authorization credentials". Attackers no longer solely aim to steal existing credentials; instead, they directly implant attacker-controllable credentials into endpoints through the supply chain, posing new challenges to existing credential security systems. In the era of AI development toolchains, the scope of credentials extends far beyond traditional system accounts and passwords. SSH public key authorization, MCP configuration authorization, extended identity materials, and built-in access credentials in development tools have all become key targets for attackers. Regarding the AI tool ecosystem, Antiy believes that extensions, MCP programs, skill execution scripts, and package dependencies should all be managed as executable software. Model tool call approval can constrain actions initiated by the model, but it may not constrain extension initialization code. Defense design should clearly define the actual permissions, writable directories, network exits, and access scope of identity materials at each execution layer.

In general, the attack paradigm presented in this incident can be summarized as follows: malicious code in the supply chain implants attacker credentials into the target environment, and relies on legitimate remote services of the system to build an independently persistent access point. Implanting credentials is a long-standing basic tactic in attack activities, but this complete tactical combination was not common in early historical cases; in the context of AI ecosystems and development toolchains, this attack chain may gradually evolve into a rapidly increasing typical paradigm.

Appendix A: IOCs

A1: Usage Rules

The following metrics are only used for this Cursor event. Addresses, keys, and filenames from historical cases are not included in the current event metric set. The exact hash and public key body are highly correlated; paths, task names, and command fragments need to be considered in context. Network addresses are used according to historical disclosure records and do not indicate that they were still under attacker control at the time of report compilation. No active connection verification was performed on the relevant addresses. [1]

A2: Extension Package Names and Payload Summary

Object	Summary
2.1.312 darwin-arm64 VSIX	SHA-256: fcbb327e38e61c4fec795e0d0591f52cd1014b813d20a5912104953c3a17c364 MD5: e0d64d53e802a0217ba6f0e656ed2f20
2.1.312 win32-x64 VSIX	SHA-256: 3de1722901165769893bef57c72594e9a54459faa23f3091b136b79966edd081 MD5: bd3b71ddf3d0cc3af76249afd2c0eda1
2.1.313 darwin-arm64 VSIX	SHA-256: 80402fdcad8799285fd93044976973ca000da7995fb93363f819bb8e4f84ee6c MD5: 34e491161e056c9b899a79b602ac73b9
2.1.313 win32-x64 VSIX	SHA-256: 6786ec90cbd45949695bd7805c9fbb26cfc91bc5fc897de0183d9bcb9cb21731 MD5: 7838a3be2ca0ff08c1bdb160ef516d39
extension.js	SHA-256: ecb58cb29078d9646ad1dcdbe51526f2d9cfea06be5fb2ab183e8ecc3ab6bf15 MD5: f49bcd657ef1e626e85ce858958c1769
Decoded tunnel private key file	SHA-256: 7e0a52885fadee05544a9e99eb18c792bec05900a5e1a66d68f8383decb0e278

The digest of a single sample cannot guarantee coverage of all renamed, repackaged, or byte-different variants. MD5 is only used for historical compatibility retrieval; SHA-256 is preferred.

A3: Network and SSH Identity Data

Type	Value	Explanation
Historical attack addresses	134[.]209[.]82[.]225	The announcement stated that the website has been taken offline; historical search results will be retained.
Outer SSH	tunnel@134[.]209[.]82[.]225 TCP 22	The victim's account and address for connecting to the remote server
Exfiltration	hxxp://134[.]209[.]82[.]225:8099/beacon	HTTP POST
Loopback Request	http://127.0.0.1:8099/beacon	Local requests cannot be considered as outbound callbacks in isolation.

Type	Value	Explanation
Authorized public key annotation	svc-telemetry	The associated string is first verified by checking the public key body and fingerprint.
Tunnel key annotation	svc-telemetry-tun	Publicly available research supplements the association strings[5].
Legal address lookup service	ifconfig.me and api.ipify.org	Not all cases of independent malicious IoC should be banned based on this.

Attacker logs in with authorization public key

ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIDdvylCEJ/k2TpezcTfedrkHyFSK9IZ3vmP6vhY4Ss0P svc-telemetry

Authorized public key fingerprint

SHA256:0Z6R+DF64Do2nwQic8a7kkfDLNNk4IiK+2U2Rci4htk

Reverse tunnel identity fingerprint

SHA256:KOAU57+ZG6u3MG4WwzPzPX2O5YEe9qdOC1Bl+6MVyDQ

A4: Files and Startup Items

Platform	File or Location	Search Instructions
macOS	~/.ssh/authorized_keys	Check for malicious public keys, but do not treat the entire authorization file as malicious.
macOS	~/.config/.smt and ~/.config/.smtk	Scripts and tunnel identity files
macOS	$TMPDIR/.stk_<10hex>	Temporary identity file mode
macOS	~/Library/LaunchAgents/com.apple.sysmond.plist	Tag com.apple.sysmond; Check actual actions
Windows	%USERPROFILE%\.ssh\authorized_keys	User authorization path
Windows	C:\ProgramData\ssh\administrators_authorized_keys	The administrator account's supplementary investigation path does not equate to confirmation that the payload is always written.
Windows	%USERPROFILE%\.config\smt.cmd and smtk	Scripts and tunnel identity files
Windows	%TEMP%\.stk_<10hex>	Temporary identity file mode, accommodating both multi-user and session temporary directories
Windows	Scheduled Tasks SysMon	Determine whether the action points to .config\smt.cmd.

Platform	File or Location	Search Instructions
Windows	%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\sysmon.cmd	User-initiated backup script
Linux code branches	~/.ssh/authorized_keys and ~/.config/.smt and .smtk	Behavioral capability indicators, unconfirmed affected platforms
Linux code branches	/tmp/.stk_<10hex>	Temporary identity file mode
Linux code branches	cron @reboot /bin/sh "<HOME>/.config/.smt"	User crontab entries should retain irrelevant tasks.

Temporary filenames can be initially screened using the regular expression `.stk_[0-9a-f]{10}`; a pattern match does not equate to a successful malicious detection. The full path to smtk on Windows is `%USERPROFILE%\.config\smtk`.

A5: Process and Command Line Fragments

Fragments	Corresponding Behavior
ssh -N -R '*:0:localhost:22'	Reverse port forwarding request
autossh -M 0	A fragment of connection maintenance tools used in Unix branches
schtasks /create /tn SysMon	Scheduled task creation attempt
systemsetup -setremotelogin on	macOS Remote Login Enabled Attempt
Add-WindowsCapability -Online -Name OpenSSH.Server~~~~0.0.1.0	Windows SSH Server capability installation attempt
StrictHostKeyChecking=no	Relax host key trust checks and automatically accept new keys; some restrictions still apply to known key changes.
UserKnownHostsFile=/dev/null	Instead of using the usual known_hosts file
ServerAliveInterval=30 和 ServerAliveCountMax=3	Connection keep-alive parameters
ExitOnForwardFailure=yes	The tunnel exits if the requested forwarding cannot be established; failure of the final destination in a subsequent connection does not necessarily cause the tunnel to exit.
IdentitiesOnly=yes	Limited authentication materials used

The above is a related search excerpt and should not be used to conclude that a process is malicious; in particular, the identity file path, parent process, destination address, and startup source should be verified. For the precise meaning of the connection parameters, please refer to the OpenSSH client configuration manual. [1,22]

Appendix B: ATT&CK Tactics and Technique Mapping for This Incident

B1: Coverage Status Across All Tactics

Tactics	Event Status	Judgment Basis
TA0043 Reconnaissance	Pre-intrusion reconnaissance	Host information acquisition after payload operation is categorized under Discovery.
TA0042 Resource Development	The infrastructure exists, but the method of acquisition is unclear.	Having servers and keys is insufficient to determine whether they were purchased, leased, or hijacked.
TA0001 Initial Access	Confirmed distribution event	Tampered software supply chain content enters the end user
TA0002 Execution	The code's capabilities are clearly defined; verification is required for execution on each machine.	JavaScript and system interpreter calls
TA0003 Persistence	The code and residual mechanism are clearly defined.	Authorized public key, startup item and extended script
TA0004 Privilege Escalation	A request to execute with high privileges was made, but its success was not confirmed.	SYSTEM scheduled tasks and service configurations requiring administrative privileges
TA0005 Stealth	There is a corresponding code behavior.	Task name spoofing, Unix dot files, hidden windows, etc.
TA0112 Defense Impairment	Insufficient evidence found	Silent error capture is not the same as disabling protection or clearing logs.
TA0006 Credential Access	The initial payload did not confirm the theft of secret values.	Implanting an attacker's identity is not stealing the victim's credentials.
TA0007 Discovery	Clear coding ability	System, user, network address, and configuration clues
TA0008 Lateral Movement	Unconfirmed	Building a reverse tunnel is not the same as being logged into other hosts within the organization.
TA0009 Collection	Additional business data collection not confirmed	No evidence was found of initial payload batch reading of source code, documentation, etc.
TA0011 Command and Control	Forwarding capabilities are clearly defined, but actual access is conditional.	SSH tunnels provide potential operational pathways.
TA0010 Exfiltration	Host metadata return attempt to clarify	External HTTP POST ; Unconfirmed theft of confidential and business data.
TA0040 Impact	Unconfirmed sabotage activities	No confirmation of encryption, erasure, or business disruption was made based on this.

B2: Key Technology Mapping

Technology	Tactics	Corresponding Behavior and Evidence Status

T1195.002 Compromise Software Supply Chain	Initial Access	Cursor Marketplace distribution was tampered with; facts of the incident
T1059.007 JavaScript	Execution	JavaScript is placed before the extension module ; public sample analysis
T1059.001 PowerShell	Execution	Windows service configuration and connection loop; code and attempts
T1059.003 Windows Command Shell	Execution	smt.cmd ; the existence of a file does not prove that it has been executed.
T1059.004 Unix Shell	Execution	Shell calls and .smt scripts; Linux code branches (limited)
T1082 System Information Discovery	Discover	Hostname, operating system, architecture
T1033 System Owner/User Discovery	Discover	Current username and identity context
T1016 System Network Configuration Discovery	Discover	Obtain a public IP address ; do not automatically identify network scans.
T1098.004 SSH Authorized Keys	Persistence	Adding the attacker's public key does not automatically indicate successful authentication or privilege escalation.
T1572 Protocol Tunneling	Command and Control	Requesting SSH reverse forwarding; success status pending telemetry confirmation.
T1543.001 Launch Agent	Persistence	User LaunchAgent ; not system-level LaunchDaemon
T1053.003 Cron	Persistence and Execution	Linux user @reboot branch; This does not expand the actual scope of impact.
T1053.005 Scheduled Task	Persistence and Execution	The SysMon task attempted to request SYSTEM ; success with high privileges is unconfirmed.
T1547.001 Registry Run Keys / Startup Folder	Persistence	Use the Startup Folder section; there is no evidence to suggest that the Run key has been modified.
T1140 Deobfuscate/Decode Files or Information	Stealth	Decoding embedded tunnel identity material; not decrypting the victim's key.
T1036.004 Masquerade Task or Service	Stealth	Names like SysMon and com.apple.sysmond are used for deception.
T1564.001 Hidden Files and Directories	Stealth	Unix dot files; Windows dot prefixes are not equivalent to hidden attributes.
T1564.003 Hidden Window	Stealth	Windows code behavior for hiding PowerShell windows

Appendix C: Expression of the Security Conclusion for the Initial Screening Command

The following commands perform a read-only initial screening for the current account, without modifying the authorization file or establishing a remote connection. They search the public key body and clearly distinguish between undiscovered and uncheckable cases. The existence of a file but a miss does not mean the payload has not run; a current user check cannot replace a multi-user investigation. Command output must be combined with version, hash, process, startup items, and network evidence.

Initial screening of current users on macOS or Linux

```
key='AAAAC3NzaC1lZDI1NTE5AAAAIDdvylCEJ/k2TpezcTfedrkHyFSK9IZ3vmP6vhY4Ss0P'
file="$HOME/.ssh/authorized_keys"
if [ ! -e "$file" ]; then
  echo 'File does not exist; other malicious activity cannot be ruled out on this basis'
elif [ ! -r "$file" ]; then
  echo 'Unreadable; this check could not complete'
else
  grep -qF -- "$key" "$file"
  rc=$?
  if [ "$rc" -eq 0 ]; then
    echo 'Matching public key material found; preserve evidence and verify fingerprint and authorization format'
  elif [ "$rc" -eq 1 ]; then
    echo 'No match in this file; the host cannot be deemed unaffected on this basi'
  else
    echo 'Read or search failure; this check could not complete'
  fi
Fi
```

Initial screening of current user and administrator authorization paths in Windows

```
$key = 'AAAAC3NzaC1lZDI1NTE5AAAAIDdvylCEJ/k2TpezcTfedrkHyFSK9IZ3vmP6vhY4Ss0P'
$paths = @(
  "$env:USERPROFILE\.ssh\authorized_keys",
  "$env:ProgramData\ssh\administrators_authorized_keys"
)
foreach ($path in $paths) {
  try {
    $item = Get-Item -LiteralPath $path -ErrorAction Stop
    $hit = Select-String -LiteralPath $item.FullName `
      -Pattern $key -SimpleMatch -Quiet -ErrorAction Stop
    if ($hit) { "Matching public key material found: $path; verify fingerprint and format" }
    else { "No match in this file: $path; compromise cannot be ruled out" }
  } catch [System.Management.Automation.ItemNotFoundException] {
    "File does not exist: $path; other malicious activity cannot be ruled out"
  } catch {
    "Check incomplete for: $path; review permissions and read errors"
  }
}
```

If you use ssh-keygen to calculate the fingerprint of the saved public key file, you should refer to the authorized public key fingerprint in Appendix A; you must not mistake the tunnel identity fingerprint for the target authorized

fingerprint. The above commands are preliminary screening examples provided in the report and have not been verified on the affected macOS or Windows terminals.

References

[1] Cursor. Security notice — malicious versions of Anthropic.claude-code 2.1.312 and 2.1.313 identified in the Cursor VS Code Extension Marketplace. Published on 2026-10-29, with the page indicating an IoC update on 2026-10-30 11:30 PT. https://cursor.com/blog/claude-code-extension-security-notice

[2] Microsoft Security Research. Inside a malicious Claude Code-branded extension and its persistent SSH backdoor. 2026-10-30. https://www.microsoft.com/en-us/security/blog/2026/10/30/inside-malicious-claude-code-branded-cursor-extension-persistent-ssh-backdoor/

[3] OpenSourceMalware. Trojanized Claude Code Extension Plants an SSH Backdoor. Analysis using its revised channel attribution and public samples. https://opensourcemalware.com/blog/trojanized-claude-code-extension-plants-persistent-ssh-backdoor

[4] Socket Research Team. Cursor Removes Malicious Claude Code 2.1.313 Extension From Its Marketplace. 2026-10-30. https://socket.dev/blog/claude-code-extension-compromise

[5] Netskope Threat Labs. Trojanized Claude Code Extension on the Cursor Marketplace. 2026-10-30. Only its payload and tunnel identity metrics are cited; channels and versions are subject to change. https://threatlabs.netskope.com/blog/2026/10/trojanized-claude-code-extension/

[6] Cursor's official macOS fix script text. Retrieve copy SHA-256 as c15900de25dd12d92adca3fa75b40253ae11edc96e70fbef0fd44252b292443a. https://cursor.com/static/security/claude-code-extension/remediate-macos.sh

[7] Cursor's official Windows fix script text. A copy with SHA-256 is retrieved as 4189b85194cadee9f3624493ec126b76a69f64b15c515bec7d2dd3d30bca4e7b. The script may continue to be updated; the summary is used to distinguish this review version and is not an indicator of malicious activity. https://cursor.com/static/security/claude-code-extension/remediate-windows.ps1

[8] OpenBSD OpenSSH Manual ssh. Port forwarding and parameter semantics. https://man.openbsd.org/ssh

[9] OpenBSD OpenSSH manual sshd_config. Configuration semantics for AuthorizedKeysFile, AuthenticationMethods, GatewayPorts, etc. https://man.openbsd.org/sshd_config

[10] Microsoft Learn. Key-based authentication in OpenSSH for Windows. https://learn.microsoft.com/en-us/windows-server/administration/openssh/openssh_keymanagement

[11] MITRE ATT&CK Enterprise tactics. Current list of tactics. https://attack.mitre.org/tactics/enterprise/

[12] MITRE ATT&CK Version History, updated April 2026. https://attack.mitre.org/resources/versions/ ; https://attack.mitre.org/resources/updates/updates-april-2026/

[13] Tatu Ylönen. Comments on the sshdwarez exploit. Published emails November 17, 1998, archived by LWN. https://lwn.net/1998/1119/sshdwarez.html

[14] Denis Sinegubko, Sucuri. Unmasking Free Premium WordPress Plugins. 2014-03-26. https://blog.sucuri.net/2014/03/unmasking-free-premium-wordpress-plugins.html

[15] Denis Sinegubko, Sucuri. When a WordPress Plugin Goes Bad. 2016-03-04. https://blog.sucuri.net/2016/03/when-wordpress-plugin-goes-bad.html

[16] OSV and GitHub Advisory Database. GHSA-gfm8-g3vm-53jh, Malicious Package in leetlog. Announcement on 2020-09-03. https://osv.dev/vulnerability/GHSA-gfm8-g3vm-53jh

[17] Phylum, now hosted by Veracode. Trojanized Ethers Forks on npm Attempting to Steal Ethereum Private Keys. 2024-10-18. https://www.veracode.com/blog/trojanized-ethers-forks-on-npm-attempting-to-steal-ethereum-private-keys/

[18] Socket. Malicious npm Packages Inject SSH Backdoors via Typosquatted Libraries. 2024-11-22. https://socket.dev/blog/malicious-npm-packages-inject-ssh-backdoors-via-typosquatted-libraries

[19] Socket .npm Malware Targets Telegram Bot Developers. 2025-04-18. https://socket.dev/blog/npm-malware-targets-telegram-bot-developers

[20] OSV and OpenSSF Malicious Packages. MAL-2026-10161, sidecar-mcp; original discoverer: Amazon Inspector. Announcement dated July 10, 2026. https://osv.dev/vulnerability/MAL-2026-10161

[21] MITRE ATT&CK official technical page. Each technology listed in the text can be viewed by its number; sub-techniques are accessed via the path /techniques/parent number/sub-number/. A complete list of links can be found in the accompanying ATTACK_mapping.csv.

[22] OpenBSD OpenSSH manual ssh_config. StrictHostKeyChecking, ExitOnForwardFailure, and IdentitiesOnly parameters. https://man.openbsd.org/ssh_config