INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-09 18:46 UTC

Typosquatted Hugging Face Repository Impersonating OpenAI Delivers Rust Infostealer via AI-Themed Loader

THREAT CAMPAIGN | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0295
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	Hugging Face platform (Windows targets); Chromium-based browsers; Gecko-based browsers; Discord; cryptocurrency wallet software; FileZilla; SSH/FTP/VPN clients
Published	2026-05-09T10:26:03
Discovery Source	Rss

Executive Summary

A malicious repository on Hugging Face's AI model-sharing platform impersonated OpenAI, reached the platform's top trending position, and was downloaded approximately 244,000 times before removal. The package installed a Rust-based credential-stealing program targeting browser sessions, cryptocurrency wallets, SSH keys, and VPN configurations on Windows systems. Organizations whose developers or data scientists installed this package face confirmed credential exposure across multiple sensitive system categories, with infrastructure links suggesting a coordinated actor operating across additional software supply chains.

Technical Analysis

The malicious repository 'open-oss-privacy-filter' on Hugging Face impersonated OpenAI's Privacy Filter project. The attack chain begins with a Python loader (T1059.006) containing AI-themed decoy code that fetches and executes a multi-stage PowerShell chain (T1059.001). PowerShell adds Defender exclusions (T1562.001) before staging a compiled Rust infostealer payload (T1195.001). The Rust payload targets: Chromium-based and Gecko-based browser credential stores and session cookies (T1555.003, T1539); cryptocurrency wallet software; FileZilla, SSH, FTP, and VPN client configurations (T1552.004); and Discord tokens. Exfiltration occurs over C2 channel (T1041). Loader infrastructure uses hardcoded credentials (CWE-798) and disables SSL certificate validation (CWE-295, T1553). The payload is delivered without integrity verification (CWE-494). HiddenLayer researchers identified infrastructure overlaps with a separate npm typosquatting campaign

distributing the WinOS 4.0 implant, indicating a coordinated actor operating across Hugging Face and npm ecosystems. No CVE assigned. Relevant CWEs: CWE-295, CWE-494, CWE-502, CWE-506, CWE-693, CWE-798. Attribution to a named threat actor is unconfirmed. Technical details are derived from secondary coverage of HiddenLayer's primary report; direct review of the HiddenLayer primary report (hiddenlayer.com) is strongly recommended before any forensic action.

Action Checklist

1. Step 1: Containment. Audit all developer and data science workstations, CI/CD pipeline runners, and ML workflow environments for prior installation of 'open-oss-privacy-filter' from Hugging Face. Isolate any system where the package was installed or where the associated Python loader or PowerShell execution was observed. Revoke and rotate all credentials accessible from those systems immediately, including: browser-stored passwords, session tokens, SSH private keys, VPN credentials, and cryptocurrency wallet keys.
2. Step 2: Detection. Search endpoint logs and EDR telemetry for: PowerShell execution spawned from a Python process; Windows Defender exclusion modifications (Event ID 5007 in Microsoft-Windows-Windows Defender/Operational); outbound connections to unfamiliar C2 infrastructure from developer machines; presence of Rust-compiled executables dropped to temp or user-profile directories. Query Hugging Face access logs or artifact download records for 'open-oss-privacy-filter'. Review npm install logs for packages associated with WinOS 4.0 infrastructure if npm is used in the same environment.
3. Step 3: Eradication. Remove the 'open-oss-privacy-filter' package and all artifacts from affected systems. Delete any staged executables, PowerShell scripts, or loader files identified during forensic review. Re-enable and update Windows Defender definitions on any system where exclusions were modified. Verify Defender exclusion policies are restored to baseline (Group Policy or Intune configuration).
4. Step 4: Recovery. After credential rotation, validate that revoked tokens and keys are no longer accepted by target services. Monitor identity provider logs and browser sync accounts for anomalous access using previously valid credentials. Restore affected systems from a known-good image where forensic review confirms payload execution occurred. Monitor C2-associated IPs and domains at the perimeter for 30 days post-remediation.
5. Step 5: Post-Incident. This incident exposes a gap in third-party package vetting for AI/ML workflows, which often operate outside standard software supply chain controls. Implement pre-installation review or allowlisting for Hugging Face repositories used in pipelines. Extend existing SCA (software composition analysis) tooling to cover Hugging Face model and package sources alongside PyPI and npm. Review whether developer and data science environments have appropriate credential isolation from production systems.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE

Escalation Criteria	Escalate to CISO, Legal, and external IR retainer immediately if forensic review confirms the Rust infostealer executed and exfiltrated credentials on any system with access to production environments, customer PII/PHI data stores, source code repositories, or cryptocurrency assets above materiality threshold, as this triggers breach notification obligations under applicable data protection regulations (GDPR Article 33, CCPA, state breach notification statutes) and may constitute a supply chain compromise requiring regulatory disclosure.
Recovery Notes	After credential rotation and system reimaging, run continuous authentication anomaly monitoring against all identity providers (Okta, Azure AD, Google Workspace) for a minimum of 90 days, specifically looking for logins from new geolocations, new devices, or off-hours access patterns using credentials that existed on confirmed-compromised systems at time of infection. Validate that all SSH authorized_keys files on internal and cloud infrastructure have been audited and that any key added or not present in your secrets management system (HashiCorp Vault, AWS Secrets Manager) has been removed and rotated. Block all identified C2 IPs and domains at the perimeter firewall and DNS layer for 30 days minimum, logging all hit attempts — any post-remediation hits indicate either incomplete eradication or that credentials were used to stage persistence on an as-yet-unidentified system.
Forensic Artifacts	Hugging Face local cache directory at `%USERPROFILE%\.cache\huggingface\hub\` — contains download manifests, repository metadata, and cached model/package files with timestamps that establish when 'open-oss-privacy-filter' was installed on each affected workstation | Rust executable dropped to `%TEMP%` or `%APPDATA%` subdirectories — Rust-compiled Windows binaries contain embedded linker metadata and panic handler strings identifiable via `strings` output; capture SHA-256 hash and submit to VirusTotal or run against a YARA rule matching Rust stealer families targeting Chromium `Login Data` SQLite paths | Windows Defender Operational log (`Microsoft-Windows-Windows Defender/Operational`) Event ID 5007 entries — records the exact exclusion path added by the Python loader to permit the Rust stealer to execute without Defender detection, providing both the dropped payload path and the timestamp of loader execution | Chromium browser SQLite databases (`Login Data`, `Cookies`, `Web Data`) at `%LOCALAPPDATA%\Google\Chrome\User Data\Default\` and equivalent Edge path — the Rust infostealer specifically targets these files for credential exfiltration; last-accessed timestamps on these files during the compromise window confirm stealer execution and scope of harvested credentials | Windows Security Event Log Event ID 4688 (Process Creation) with full command-line logging enabled — captures the Python-to-PowerShell execution chain initiated by the malicious loader, providing the exact command-line arguments used to download and stage the Rust executable, including any encoded payloads or C2 URLs passed as arguments

Per-Action IR Details

Step 1: Containment — Audit all developer and data science workstations, CI/CD pipeline runners, and ML workflow environments for prior installation of 'open-oss-privacy-filter' from Hugging Face. Isolate any system where the package was installed or where the associated Python loader or PowerShell execution was observed. Revoke and rotate all credentials accessible from those systems immediately — browser-stored passwords, session tokens, SSH private keys, VPN credentials, and cryptocurrency wallet keys.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy

Controls: NIST IR-4 (Incident Handling), NIST AC-2 (Account Management), CIS 5.3 (Disable Dormant Accounts), CIS 6.2 (Establish an Access Revoking Process)

Compensating: Run the following PowerShell one-liner across all developer workstations via PSRemoting or manual execution to detect prior package installation: `pip show open-oss-privacy-filter 2>&1; Get-ChildItem -Path $env:USERPROFILE -Recurse -Filter 'open-oss-privacy-filter*' -ErrorAction SilentlyContinue`. For CI/CD runners

(GitHub Actions, GitLab, Jenkins), grep pipeline logs for 'open-oss-privacy-filter' and 'huggingface.co' in runner stdout artifacts. Use `ssh-keygen -l -f ~/.ssh/id_rsa` to fingerprint SSH keys pre- and post-rotation to confirm replacement. For browser credential revocation without enterprise tooling, manually clear Chrome/Edge Login Data SQLite files at `%LOCALAPPDATA%\Google\Chrome\User Data\Default\Login Data` and force re-authentication on all SaaS platforms.

Evidence: BEFORE isolating, capture: (1) pip freeze output and `%LOCALAPPDATA%\pip\Cache` directory listing to confirm package version installed; (2) PowerShell transcription logs or `Get-WinEvent -LogName 'Windows PowerShell'` for Python-spawned PowerShell invocations; (3) Windows Prefetch files (`C:\Windows\Prefetch\`) for Rust executable execution evidence (prefetch hashes for unknown .exe files in `%TEMP%` or `%APPDATA%`); (4) full memory acquisition via WinPmem or DumpIt before network isolation if the Rust infostealer process is still resident; (5) `netstat -anob` output timestamped at isolation to capture any active C2 connections from the stealer process.

Step 2: Detection — Search endpoint logs and EDR telemetry for: PowerShell execution spawned from a Python process; Windows Defender exclusion modifications (Event ID 5007 in Microsoft-Windows-Windows Defender/Operational); outbound connections to unfamiliar C2 infrastructure from developer machines; presence of Rust-compiled executables dropped to temp or user-profile directories. Query Hugging Face access logs or artifact download records for 'open-oss-privacy-filter'. Review npm install logs for packages associated with WinOS 4.0 infrastructure if npm is used in the same environment.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis

Controls: NIST SI-4 (System Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-2 (Event Logging), CIS 8.2 (Collect Audit Logs)

Compensating: Deploy Sysmon with SwiftOnSecurity config to capture Event ID 1 (Process Create) filtering on `ParentImage` containing `python.exe` or `pythonw.exe` and `Image` containing `powershell.exe`. Hunt for Rust executables in temp paths using: `Get-ChildItem -Path $env:TEMP,$env:APPDATA -Recurse -Include '*.exe' | Select-String -Pattern 'rustc' -Encoding Byte` (Rust binaries embed compiler metadata). Use Sigma rule `proc_creation_win_powershell_parent_python.yml` from the SigmaHQ repository for SIEM-less correlation via Hayabusa log parser against collected Windows EVTX files. For Defender exclusion hunting without EDR, query `HKLM:\SOFTWARE\Microsoft\Windows Defender\Exclusions` registry hive on all affected hosts using `Get-ItemProperty 'HKLM:\SOFTWARE\Microsoft\Windows Defender\Exclusions\*'`.

Evidence: Collect before analysis concludes: (1) `Microsoft-Windows-Windows Defender/Operational` event log (Event ID 5007 = exclusion added, Event ID 1116 = malware detected) exported as EVTX from all developer machines; (2) Windows Security Event Log Event ID 4688 (Process Creation with command-line auditing enabled) filtered for `powershell.exe` with parent `python.exe` or `pip.exe`; (3) Sysmon Event ID 3 (Network Connection) logs filtered for outbound connections on non-standard ports from `%TEMP%\*.exe` or `%APPDATA%\*.exe` processes; (4) Hugging Face download receipts from `%USERPROFILE%\.cache\huggingface\` directory, which caches repository metadata and download manifests; (5) DNS query logs from the local DNS resolver or Windows DNS Client event log (`Microsoft-Windows-DNS-Client/Operational`) for domains resolved by the Rust stealer's C2 beacon.

Step 3: Eradication — Remove the 'open-oss-privacy-filter' package and all artifacts from affected systems. Delete any staged executables, PowerShell scripts, or loader files identified during forensic review. Re-enable and update Windows Defender definitions on any system where exclusions were modified. Verify Defender exclusion policies are restored to baseline (Group Policy or Intune configuration).

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), NIST SI-3 (Malicious Code Protection), NIST CM-6 (Configuration Settings), CIS 2.3 (Address Unauthorized Software), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Run `pip uninstall open-oss-privacy-filter -y` and follow with `pip cache purge` to clear cached wheel files. Use the following PowerShell to locate and remove all Rust stealer artifacts dropped by the loader: `Get-ChildItem -Path $env:TEMP,$env:APPDATA,$env:LOCALAPPDATA -Recurse -Include '*.exe','*.ps1','*.bat' | Where-Object {$_.CreationTime -gt (Get-Date).AddDays(-30)} | Select-Object FullName,CreationTime,LastWriteTime | Export-Csv

dropped_artifacts.csv`. Restore Defender exclusions by running `Remove-MpPreference -ExclusionPath ` for each unauthorized exclusion identified via the registry audit in Step 2. Validate Defender definitions are current with `Get-MpComputerStatus | Select-Object AntivirusSignatureLastUpdated,AntivirusSignatureVersion`.

Evidence: Before removal, preserve: (1) SHA-256 hashes of all identified Rust executables using `Get-FileHash -Algorithm SHA256 ` for IOC sharing and YARA rule development; (2) copy of modified Defender exclusion registry keys (`HKLM:\SOFTWARE\Microsoft\Windows Defender\Exclusions`) exported via `reg export` before restoration; (3) full directory listing of `%USERPROFILE%\.cache\huggingface\hub\` to document which other Hugging Face repositories were accessed from this workstation; (4) strings output from the Rust executable (`strings.exe` from Sysinternals) to extract embedded C2 URLs, wallet regex patterns, or targeted browser profile paths before deletion.

Step 4: Recovery — After credential rotation, validate that revoked tokens and keys are no longer accepted by target services. Monitor identity provider logs and browser sync accounts for anomalous access using previously valid credentials. Restore affected systems from a known-good image where forensic review confirms payload execution occurred. Monitor C2-associated IPs and domains at the perimeter for 30 days post-remediation.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST AC-2 (Account Management), NIST SI-7 (Software, Firmware, and Information Integrity), CIS 6.2 (Establish an Access Revoking Process), CIS 4.6 (Securely Manage Enterprise Assets and Software)

Compensating: Validate SSH key revocation by testing the old key against target hosts: `ssh -i old_key.pem user@host -o BatchMode=yes` — a rejection confirms revocation. For browser-stored credential validation, check Google Account security events at myaccount.google.com/security-checkup and Microsoft account sign-in logs at account.microsoft.com/security for sessions initiated after the compromise window. For C2 perimeter monitoring without a commercial threat feed, create firewall ACL deny rules for identified C2 IPs and configure DNS sinkhole entries for C2 domains; log all blocked attempts via pfSense or Windows Firewall audit logs (`Microsoft-Windows-Windows Firewall With Advanced Security/Firewall`) and alert on any hits using a cron job parsing firewall logs for C2 IOCs.

Evidence: Before restoring from image: (1) collect Windows Vault credential store dump (`vaultcmd /listcreds:'Windows Credentials' /all`) to document what credentials were accessible to the stealer at time of execution; (2) export Chrome/Edge `Login Data`, `Cookies`, `Web Data`, and `Local State` SQLite databases from `%LOCALAPPDATA%\Google\Chrome\User Data\Default\` — the Rust stealer specifically targets these files and comparison against a baseline reveals what was exfiltrated; (3) enumerate SSH `known_hosts` and `authorized_keys` files to map lateral movement risk from stolen keys; (4) capture a list of installed cryptocurrency wallet extensions from Chrome `Extensions` directory before wiping, as the stealer targets extension storage for wallet seed phrases.

Step 5: Post-Incident — This incident exposes a gap in third-party package vetting for AI/ML workflows, which often operate outside standard software supply chain controls. Implement pre-installation review or allowlisting for Hugging Face repositories used in pipelines. Extend existing SCA (software composition analysis) tooling to cover Hugging Face model and package sources alongside PyPI and npm. Review whether developer and data science environments have appropriate credential isolation from production systems.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity

Controls: NIST IR-4 (Incident Handling), NIST IR-8 (Incident Response Plan), NIST SA-12 (Supply Chain Protection), NIST SI-2 (Flaw Remediation), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Implement a Hugging Face repository allowlist by creating a `pip.conf` or `pyproject.toml` index restriction that blocks unapproved Hugging Face package sources; document approved repositories in a markdown-controlled allowlist in your git repo. For SCA coverage of Hugging Face without enterprise tooling, add a pre-commit hook using `pip-audit` (free, maintained by PyPA) that runs against `requirements.txt` before every commit,

and extend it with a custom script that queries the Hugging Face API (`https://huggingface.co/api/models/`) to verify publisher identity against an approved-publisher list. Separate ML environment credentials from production using dedicated service accounts with scoped permissions: create isolated conda environments or Docker containers for ML workloads with no access to production SSH keys or browser profiles.

Evidence: For the lessons-learned record: (1) download count timeline from Hugging Face (244,000 downloads documented) to quantify blast radius and support breach notification scope assessment; (2) git history of pipeline configuration files (`requirements.txt`, `environment.yml`, `Dockerfile`) to determine when `open-oss-privacy-filter` was first introduced and by which commit/contributor; (3) CI/CD pipeline execution logs showing every environment where the package was installed as a dependency, including transitive installs; (4) identity provider authentication logs covering the 30-day window prior to detection, filtered for accounts whose credentials were accessible on confirmed-compromised systems, to identify any unauthorized access that occurred pre-containment.

Detection Guidance

Primary behavioral indicators: (1) Python process spawning PowerShell (parent-child process chain; visible in EDR process trees and Windows Security Event ID 4688 with command-line auditing enabled). (2) Windows Defender exclusion changes, query Microsoft-Windows-Windows Defender/Operational Event ID 5007 for exclusion additions on developer endpoints. (3) Rust-compiled binary execution from user-writable paths (AppData, Temp) on systems with recent Hugging Face package activity. (4) Outbound HTTPS connections to non-standard infrastructure from ML workflow hosts. IOC note: specific C2 domains, IPs, and payload hashes are documented in HiddenLayer's primary report (hiddenlayer.com/insight/malware-found-in-trending-hugging-face-repository-open-oss-privacy-filter); retrieve and ingest those IOCs directly rather than relying on this summary. Confidence on IOC values in this item is medium; direct HiddenLayer report verification is required before blocking production traffic.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`https://huggingface.co/open-oss-privacy-filter`	Malicious Hugging Face repository — now removed; flag any reference in download or access logs	HIGH
DOMAIN	`See HiddenLayer primary report for confirmed C2 infrastructure`	Specific C2 domains and payload hashes are documented in HiddenLayer's technical report; not reproduced here to avoid unverified IOC propagation	LOW

Framework Mappings

MITRE-ATTACK

- **T1071.001** — Web Protocols
- **T1497** — Virtualization/Sandbox Evasion
- **T1539** — Steal Web Session Cookie
- **T1555.003** — Credentials from Web Browsers

- **T1059.001** — PowerShell
- **T1553** — Subvert Trust Controls
- **T1552.004** — Private Keys
- **T1027** — Obfuscated Files or Information
- **T1041** — Exfiltration Over C2 Channel
- **T1562.001** — Disable or Modify Tools
- **T1036.005** — Match Legitimate Resource Name or Location
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1588.002** — Tool
- **T1059.006** — Python
- **T1552.001** — Credentials In Files
- **T1204.002** — Malicious File

NIST-800-53R5

- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-7** — Software, Firmware, and Information Integrity
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **SI-10** — Information Input Validation
- **IA-5** — Authenticator Management
- **SC-8** — Transmission Confidentiality and Integrity
- **SC-17** — Public Key Infrastructure Certificates
- **CM-3** — Configuration Change Control

OWASP-TOP10-2021

- **A08:2021** — Software and Data Integrity Failures
- **A07:2021** — Identification and Authentication Failures
- **A02:2021** — Cryptographic Failures

CIS-V8

- **16.10** — Apply Secure Design Principles in Application Architectures
- **3.10** — Encrypt Sensitive Data in Transit
- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **6.3** — Require MFA for Externally-Exposed Applications

ISO-27001-2022

- **A.8.28** — Secure coding

HIPAA-SECURITY

- **164.312(d)** — Person or Entity Authentication

SOC2-TSC

- **CC6.1** — Logical access security software, infrastructure, and architectures

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1071.001	Web Protocols	Command-And-Control
T1497	Virtualization/Sandbox Evasion	Defense-Evasion
T1539	Steal Web Session Cookie	Credential-Access
T1555.003	Credentials from Web Browsers	Credential-Access
T1059.001	PowerShell	Execution
T1553	Subvert Trust Controls	Defense-Evasion
T1552.004	Private Keys	Credential-Access
T1027	Obfuscated Files or Information	Defense-Evasion
T1041	Exfiltration Over C2 Channel	Exfiltration
T1562.001	Disable or Modify Tools	Defense-Evasion
T1036.005	Match Legitimate Resource Name or Location	Defense-Evasion
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1588.002	Tool	Resource-Development
T1059.006	Python	Execution
T1552.001	Credentials In Files	Credential-Access
T1204.002	Malicious File	Execution

Sources

Source	URL	Tier
Security News	https://www.bleepingcomputer.com/news/security/fake-openai-reposito...	T3

Source	URL	Tier
Malware Found in Trending Hugging Face Repository "Open-OSS ...	https://www.hiddenlayer.com/insight/malware-found-in-trending-huggi...	T3
Hugging Face, ClawHub Abused for Malware Distribution	https://www.securityweek.com/hugging-face-clawhub-abused-for-malwar...	T3
Hugging Face platform hijacked to send out Android malware	https://www.techradar.com/pro/security/hugging-face-platform-hijack...	T3
Hugging Face says it detected 'unauthorized access' to its AI model ...	https://finance.yahoo.com/news/hugging-face-says-detected-unauthori...	T3

Generated 2026-05-09 18:46 UTC by TJS Security Command Center