INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-04-30 19:01 UTC

DEEP#DOOR Python Backdoor Disables Windows Defenses and Harvests Browser and Cloud Credentials via Tunneled C2

THREAT CAMPAIGN | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0245
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	Windows systems; browsers (specific vendors unconfirmed); cloud services (specific vendors unconfirmed)
Published	2026-04-30T08:36:00
Discovery Source	Rss

Executive Summary

DEEP#DOOR is a Python-based backdoor campaign targeting Windows systems by disabling security controls and stealing credentials stored in browsers and cloud services. Attackers use tunneled command-and-control channels to avoid detection by standard network monitoring tools. Note: DEEP#DOOR reporting originates from a single third-tier source with no corroborating primary-tier vendor or government analysis; technical details and IOCs should be treated as preliminary. Organizations exposed to this threat risk unauthorized access to cloud infrastructure, privileged accounts, and sensitive business systems without visible indicators on the network perimeter.

Technical Analysis

DEEP#DOOR is a multi-stage Python backdoor framework. Initial execution triggers a Windows batch script (T1059.003) that disables host-based security controls (T1562.001) before establishing persistence via registry run keys and scheduled tasks (T1547.001, T1053.005). The credential harvesting module targets browser credential stores (T1555.003, T1552.001) and cloud service tokens (T1552). C2 communication uses protocol tunneling (T1572) and proxy chaining (T1090) over standard application-layer protocols (T1071, T1071.001) to blend with legitimate traffic and evade network-based detection. Exfiltration occurs over the same tunneled channel (T1041). No CVE is assigned; this is a campaign-based threat. Relevant CWEs: CWE-312 (cleartext storage of sensitive information), CWE-522 (insufficiently protected credentials), CWE-693 (protection mechanism failure). Threat actor attribution is unconfirmed. Source reporting originates from The Hacker News

(T3); no primary-tier technical analysis is currently available. Technical specifics, including exact browser targets, cloud service targets, and IOCs, should be treated as preliminary pending vendor or CISA confirmation.

Action Checklist

1. Containment, Identify and isolate any Windows hosts showing unexpected batch script execution followed by changes to Windows Defender or firewall configurations. Block outbound connections to suspected tunneling service infrastructure (e.g., ngrok, localhost.run, similar services, pending confirmation of actual C2 infrastructure from vendor or CISA advisories) at the perimeter firewall and proxy layer until behavioral review is complete.
2. Detection, Query EDR and SIEM for the following: PowerShell or cmd.exe spawning Python processes (T1059.001, T1059.003); Windows Security Center API calls disabling antivirus or firewall (Event ID 5001, 5010, 5012 from Windows Defender operational log); scheduled task creation (Event ID 4698) or registry run key writes under HKCU\Software\Microsoft\Windows\CurrentVersion\Run; outbound connections to tunneling domains or non-standard HTTPS destinations on ports 443/80 with high session persistence. Check browser credential store access patterns outside of browser processes.
3. Eradication, No vendor patch applies; this is a malware campaign, not a software vulnerability. Remove identified malicious Python scripts, batch files, scheduled tasks, and registry persistence entries. Re-enable Windows Defender, Windows Firewall, and any disabled security controls. Rotate all credentials accessible from affected systems, including browser-stored passwords and cloud service tokens/API keys.
4. Recovery, After credential rotation, verify cloud service access logs for unauthorized sessions and revoke any active sessions or tokens issued during the suspected compromise window. Re-validate security control status on remediated hosts using your endpoint management platform. Monitor those hosts for 30 days for recurrence of scheduled task creation, Python process spawning, or tunneling service connections.
5. Post-Incident, Assess whether endpoint controls would have prevented batch-script-based defense evasion; if not, evaluate application control policies (e.g., Windows Defender Application Control, AppLocker) to restrict unauthorized script execution. Review policies governing browser-stored credentials in enterprise environments, consider enforcing enterprise password manager adoption and disabling browser credential storage via Group Policy. Audit outbound proxy and firewall rules to block known tunneling services categorically.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to senior leadership, legal, and cloud provider security teams immediately if cloud service access logs confirm unauthorized API calls, data exfiltration events, or privilege escalation using harvested credentials during the compromise window, as this triggers breach notification obligations under applicable data protection regulations (GDPR Art. 33, CCPA, state breach notification laws) if PII or sensitive business data was accessible from affected cloud accounts.

Recovery Notes	Credential rotation alone is insufficient if cloud service sessions or OAuth tokens issued during the compromise window remain active — explicitly revoke all sessions and tokens via provider admin consoles or CLI, not just password resets. Monitor remediated hosts for a minimum of 30 days using Sysmon Event ID 1 (python.exe, pythonw.exe process creation), Event ID 3 (outbound connections to port 443/80 from Python processes), and Windows Security Event ID 4698 (scheduled task creation), as DEEP#DOOR-style campaigns frequently include secondary persistence mechanisms that survive initial eradication. Confirm cloud provider MFA is enforced on all accounts whose credentials were stored in affected browsers before returning those accounts to production use.
Forensic Artifacts	Windows Defender Operational Event Log (Microsoft-Windows-Windows Defender/Operational) — Event IDs 5001, 5004, 5010, 5012 establish the exact timestamp when DEEP#DOOR's batch script disabled real-time protection and scanning, anchoring the start of the undefended exposure window. | Sysmon Event ID 1 (Process Creation) logs showing python.exe or pythonw.exe spawned by cmd.exe or powershell.exe, with full CommandLine and ParentCommandLine fields capturing the backdoor invocation syntax and any inline C2 configuration parameters. | Browser SQLite credential databases pre-eradication: Chrome/Edge at C:\Users\\AppData\Local\Google\Chrome\User Data\Default\Login Data and Firefox at C:\Users\\AppData\Roaming\Mozilla\Firefox\Profiles\\logins.json — filesystem last-access timestamps on these files confirm whether DEEP#DOOR's harvesting module accessed them and when. | Scheduled Task XML files from C:\Windows\System32\Tasks\ and C:\Windows\SysWOW64\Tasks\ for any tasks created during the intrusion window, which may contain the Python interpreter path, backdoor script path, and trigger schedule used for DEEP#DOOR's persistence mechanism. | Memory image (captured with WinPmem or Magnet RAM Capture before isolation) containing the in-memory Python interpreter runtime, decrypted tunneling service authentication tokens (e.g., ngrok authtoken), active C2 channel state, and any credentials harvested but not yet exfiltrated at time of detection.

Per-Action IR Details

Containment — Identify and isolate any Windows hosts showing unexpected batch script execution followed by changes to Windows Defender or firewall configurations. Block outbound connections to known tunneling service infrastructure (e.g., ngrok, localhost.run, similar services) at the perimeter firewall and proxy layer until behavioral review is complete.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST SI-4 (System Monitoring), NIST SC-7 (Boundary Protection), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 4.5 (Implement and Manage a Firewall on End-User Devices), CIS 13.4 (Perform Traffic Filtering Between Network Segments)

Compensating: On each suspect host, run: 'Get-MpPreference | Select-Object DisableRealtimeMonitoring, DisableIOAVProtection' and 'netsh advfirewall show allprofiles state' to confirm Defender and firewall states. Use Windows Firewall with Advanced Security (wf.msc) to add outbound block rules for known ngrok IP ranges (3.12.0.0/14, 52.0.0.0/8 — validate current ranges at ngrok's docs) and localhost.run domains. On the perimeter, push DNS sinkholes for *.ngrok.io, *.ngrok-free.app, *.localhost.run, and *.serveo.net via your DNS forwarder (e.g., pi-hole, Windows DNS RPZ). Network isolate confirmed hosts by moving them to a quarantine VLAN or disabling the switch port.

Evidence: BEFORE isolating: capture a full memory image using Magnet RAM Capture or WinPmem to preserve in-memory Python interpreter state, injected shellcode, and decrypted C2 channel keys. Export Windows Defender operational log (Microsoft-Windows-Windows Defender/Operational) — specifically Event IDs 5001 (real-time protection disabled), 5004 (real-time protection configuration changed), 5010 (scanning for malware disabled), 5012

(scanning for downloads disabled). Dump active network connections via 'netstat -anob > c:\ir\netstat_pre_contain.txt' and 'Get-NetTCPConnection | Export-Csv c:\ir\tcp_connections.csv' to document tunneled sessions before the host goes offline. Capture DNS cache: 'ipconfig /displaydns > c:\ir\dns_cache.txt' to reveal tunneling service domains that may not appear in firewall logs.

Detection — Query EDR and SIEM for the following: PowerShell or cmd.exe spawning Python processes (T1059.001, T1059.003); Windows Security Center API calls disabling antivirus or firewall (Event ID 5001, 5010, 5012 from Windows Defender operational log); scheduled task creation (Event ID 4698) or registry run key writes under HKCU\Software\Microsoft\Windows\CurrentVersion\Run; outbound connections to tunneling domains or non-standard HTTPS destinations on ports 443/80 with high session persistence. Check browser credential store access patterns outside of browser processes.

NIST Phase: Detection Analysis

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

Controls: NIST SI-4 (System Monitoring), NIST AU-2 (Event Logging), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-12 (Audit Record Generation), NIST IR-5 (Incident Monitoring), CIS 8.2 (Collect Audit Logs), CIS 8.11 (Collect Network Traffic Flow Logs)

Compensating: Deploy Sysmon with SwiftOnSecurity config (minimum: ProcessCreate Event ID 1, NetworkConnect Event ID 3, RegistryEvent Event ID 13). Query Sysmon logs via PowerShell: 'Get-WinEvent -LogName "Microsoft-Windows-Sysmon/Operational" | Where-Object {$_.Id -eq 1 -and $_.Message -match "python"} | Select-Object TimeCreated, Message | Export-Csv c:\ir\python_procs.csv'. For browser credential store access, look for Sysmon Event ID 10 (ProcessAccess) targeting Chrome's 'Login Data' path (C:\Users*\AppData\Local\Google\Chrome\User Data\Default\Login Data) or Edge's equivalent by a non-browser process. Use Sigma rule 'proc_creation_win_python_inline_command' adapted for cmd/PowerShell parent processes. For scheduled task hunting without EDR, run: 'Get-ScheduledTask | Where-Object {$_.Date -gt (Get-Date).AddDays(-14)} | Export-Csv c:\ir\new_tasks.csv' and cross-reference against known-good baselines.

Evidence: Windows Security Event Log Event ID 4688 (Process Creation) with CommandLine auditing enabled — filter for python.exe, pythonw.exe, or py.exe with parent process cmd.exe or powershell.exe. Windows Defender Operational log Event IDs 5001, 5010, 5012 timestamped prior to C2 activity to establish the disable-then-beacon sequence. Scheduled Task XML files stored in C:\Windows\System32\Tasks\ — examine newly created tasks for Python interpreter invocations or batch file references. Registry export of HKCU\Software\Microsoft\Windows\CurrentVersion\Run and HKLM\Software\Microsoft\Windows\CurrentVersion\Run for persistence entries added during the suspected intrusion window. Browser SQLite credential databases (Chrome/Edge: C:\Users*\AppData\Local\Google\Chrome\User Data\Default\Login Data; Firefox: C:\Users*\AppData\Roaming\Mozilla\Firefox\Profiles*.default\logins.json) — check filesystem last-access timestamps against the intrusion timeline to confirm DEEP#DOOR's credential harvesting activity.

Eradication — No vendor patch applies; this is a malware campaign, not a software vulnerability. Remove identified malicious Python scripts, batch files, scheduled tasks, and registry persistence entries. Re-enable Windows Defender, Windows Firewall, and any disabled security controls. Rotate all credentials accessible from affected systems, including browser-stored passwords and cloud service tokens/API keys.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST IR-4 (Incident Handling), NIST SI-2 (Flaw Remediation), NIST SI-3 (Malicious Code Protection), NIST SI-7 (Software, Firmware, and Information Integrity), NIST IA-5 (Authenticator Management), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 5.2 (Use Unique Passwords)

Compensating: Use PowerShell to enumerate and remove DEEP#DOOR persistence: 'Get-ScheduledTask | Where-Object {$_.TaskName -like "**"} | Unregister-ScheduledTask -Confirm:$false'. Remove registry run keys: 'Remove-ItemProperty -Path "HKCU:\Software\Microsoft\Windows\CurrentVersion\Run" -Name "*"'. Re-enable Defender: 'Set-MpPreference -DisableRealtimeMonitoring $false; Set-MpPreference -DisableIOAVProtection $false'. For credential rotation, use a scripted approach — export the list of URLs from each browser's Login Data SQLite file

using DB Browser for SQLite (free), enumerate all unique cloud service domains, and force password resets for those services. For cloud API key rotation, use each provider's CLI (aws iam delete-access-key / az ad sp credential reset / gcloud iam service-accounts keys delete) and invalidate OAuth tokens via provider admin consoles. Build a YARA rule targeting the Python backdoor's known string patterns and scan all user-writable directories: 'yara64.exe deep_door.yar C:\Users\ /r'.

Evidence: Before removing any artifact, forensically image or hash (SHA-256) every malicious file using: 'Get-FileHash -Algorithm SHA256'. Export the full scheduled task XML before deletion: 'Export-ScheduledTask -TaskName "" > c:\ir\task_backup.xml'. Capture registry state before cleanup: 'reg export HKCU\Software\Microsoft\Windows\CurrentVersion\Run c:\ir\run_keys_pre_clean.reg'. Collect all Python scripts and batch files identified — these are primary evidence for attribution and behavioral analysis. Document Defender and firewall states (disabled configurations, timestamps of changes) from Windows Defender Operational log before re-enabling, as this establishes the attacker's evasion sequence and dwell time window.

Recovery — After credential rotation, verify cloud service access logs for unauthorized sessions and revoke any active sessions or tokens issued during the suspected compromise window. Re-validate security control status on remediated hosts using your endpoint management platform. Monitor those hosts for 30 days for recurrence of scheduled task creation, Python process spawning, or tunneling service connections.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST IR-5 (Incident Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AC-17 (Remote Access), NIST SI-4 (System Monitoring), CIS 5.1 (Establish and Maintain an Inventory of Accounts), CIS 6.2 (Establish an Access Revoking Process), CIS 8.2 (Collect Audit Logs)

Compensating: For cloud session validation without a CASB: pull AWS CloudTrail logs filtering for 'ConsoleLogin' and 'AssumeRole' events in the compromise window ('aws cloudtrail lookup-events --lookup-attributes AttributeKey=EventName,AttributeValue=ConsoleLogin --start-time --end-time '); for Azure, query Sign-in logs in Azure AD for the same window. Revoke all active sessions with 'aws iam delete-access-key' for each affected IAM user and 'az account clear' followed by re-authentication. For host re-validation without an endpoint management platform, use a PowerShell baseline comparison script: capture 'Get-MpComputerStatus', 'netsh advfirewall show allprofiles', 'Get-ScheduledTask', and 'Get-ItemProperty HKCU:\Software\Microsoft\Windows\CurrentVersion\Run' and diff against known-good snapshots. For 30-day monitoring without EDR, configure Sysmon Event ID 1 alerts (python.exe, pythonw.exe process creation) forwarded to Windows Event Forwarding (WEF) collector, and set a daily cron-equivalent scheduled task that runs 'netstat -anob | findstr ESTABLISHED > c:\ir\daily_netstat_$(Get-Date -Format yyyyMMdd).txt' for manual review.

Evidence: Cloud provider access logs covering the full suspected compromise window — specifically AWS CloudTrail, Azure AD Sign-in Logs, or GCP Cloud Audit Logs showing API calls, resource access, and OAuth token issuances from affected credential sets. Browser credential store files post-eradication to confirm harvested passwords have been fully rotated (verify by checking if stored credentials in Login Data or logins.json reference the same domains as the attacker-accessed cloud services). Endpoint security control status report (Defender real-time protection state, firewall profile states) from the endpoint management platform or from PowerShell Get-MpComputerStatus output, timestamped post-remediation to confirm controls are restored and stable.

Post-Incident — Assess whether endpoint controls would have prevented batch-script-based defense evasion; if not, evaluate application control policies (e.g., Windows Defender Application Control, AppLocker) to restrict unauthorized script execution. Review policies governing browser-stored credentials in enterprise environments — consider enforcing enterprise password manager adoption and disabling browser credential storage via Group Policy. Audit outbound proxy and firewall rules to block known tunneling services categorically.

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 SI-3 (Malicious Code Protection), NIST SI-7 (Software, Firmware, and Information Integrity), NIST CM-7 (Least Functionality), NIST CM-11

(User-Installed Software), NIST IA-5 (Authenticator Management), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 2.3 (Address Unauthorized Software), CIS 4.6 (Securely Manage Enterprise Assets and Software), CIS 6.3 (Require MFA for Externally-Exposed Applications), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: For WDAC/AppLocker without enterprise tooling: deploy an AppLocker audit-mode policy via GPO targeting script rules — block execution of python.exe and pythonw.exe from all paths except IT-approved directories (e.g., C:\Program Files\Python*), and block .bat/.cmd execution from user-writable paths (C:\Users*, C:\Temp*, C:\ProgramData*). Export the AppLocker event log (Microsoft-Windows-AppLocker/Script) to validate coverage. For browser credential policy: push the 'PasswordManagerEnabled' registry key via GPO: 'HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Google\Chrome\PasswordManagerEnabled = 0' (Chrome) and equivalent for Edge ('HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Edge\PasswordManagerEnabled = 0'). For tunneling service blocking without a next-gen firewall: maintain a DNS blocklist updated weekly using Cisco Umbrella's free threat intelligence feed or the CISA Known Bad Domains list, covering *.ngrok.io, *.ngrok-free.app, *.localhost.run, *.serveo.net, *.pagekite.me, and *.telebit.cloud. Document all control gaps identified in this incident in the IR lessons-learned report per NIST 800-61r3 §4.

Evidence: Lessons-learned documentation capturing: (1) which existing endpoint controls failed to alert on batch-script-executed Python backdoor deployment; (2) timeline from initial batch script execution to Defender disable to first C2 beacon, establishing dwell time; (3) list of all browser-stored credentials and cloud tokens confirmed or suspected to have been harvested, categorized by service type and privilege level. Gap analysis report comparing pre-incident AppLocker/WDAC policy coverage against DEEP#DOOR's execution chain (cmd.exe → batch script → python.exe → tunneled C2) to document which policy rules would have broken the kill chain and at which stage.

Detection Guidance

Primary detection signals: (1) cmd.exe or PowerShell executing .bat files that invoke sc.exe, netsh, or Set-MpPreference to disable security services, correlate with Windows Security Event Log and Defender Operational Log (Event IDs 5001, 5004, 5010, 5012). (2) Python.exe spawning as a child process of cmd.exe or wscript.exe, flag in EDR process tree analysis. (3) Scheduled task creation (Event ID 4698) or registry writes to run keys by non-standard parent processes. (4) Outbound HTTPS or TCP connections with high session duration to tunneling service domains (ngrok.io, localhost.run, similar), query proxy and DNS logs. (5) Access to browser credential store files (e.g., Login Data in Chrome profile directories, logins.json in Firefox profiles) by processes other than the browser binary. Caveat: The following patterns are based on campaign description; without confirmed IOCs, detection rules should include additional environmental context to reduce false positives. Coordinate with threat intelligence updates from CISA or vendors before broad deployment. Note: specific IOC values (hashes, domains, IPs) are not confirmed in available source reporting. Detection should rely on behavioral patterns until authoritative IOCs are published by CISA or a primary-tier vendor.

Indicators of Compromise

Type	Value	Context	Confidence
DOMAIN	`ngrok.io (category)`	Tunneling service infrastructure used for C2 — specific domains not confirmed in available reporting; block tunneling service categories at perimeter	LOW

Type	Value	Context	Confidence
HASH	`not available`	No file hashes published in available source reporting; monitor for behavioral indicators until primary-tier IOCs are released	LOW

Framework Mappings

MITRE-ATTACK

- **T1572** — Protocol Tunneling
- **T1547.001** — Registry Run Keys / Startup Folder
- **T1547** — Boot or Logon Autostart Execution
- **T1552** — Unsecured Credentials
- **T1059.003** — Windows Command Shell
- **T1041** — Exfiltration Over C2 Channel
- **T1053.005** — Scheduled Task
- **T1562.001** — Disable or Modify Tools
- **T1090** — Proxy
- **T1071** — Application Layer Protocol
- **T1555.003** — Credentials from Web Browsers
- **T1071.001** — Web Protocols
- **T1059.001** — PowerShell
- **T1552.001** — Credentials In Files

NIST-800-53R5

- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **SI-7** — Software, Firmware, and Information Integrity
- **IA-5** — Authenticator Management

OWASP-TOP10-2021

- **A04:2021** — Insecure Design
- **A07:2021** — Identification and Authentication Failures

CIS-V8

- **5.2** — Use Unique Passwords
- **6.3** — Require MFA for Externally-Exposed Applications
- **8.2** — Collect Audit Logs

HIPAA-SECURITY

- **164.308(a)(5)(ii)(D)** — Password Management
- **164.312(d)** — Person or Entity Authentication

SOC2-TSC

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

NIST-CSF-2

- **DE.CM-01** — Networks and network services are monitored

ISO-27001-2022

- **A.5.23** — Information security for use of cloud services

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1572	Protocol Tunneling	Command-And-Control
T1547.001	Registry Run Keys / Startup Folder	Persistence
T1547	Boot or Logon Autostart Execution	Persistence
T1552	Unsecured Credentials	Credential-Access
T1059.003	Windows Command Shell	Execution
T1041	Exfiltration Over C2 Channel	Exfiltration
T1053.005	Scheduled Task	Execution
T1562.001	Disable or Modify Tools	Defense-Evasion
T1090	Proxy	Command-And-Control
T1071	Application Layer Protocol	Command-And-Control
T1555.003	Credentials from Web Browsers	Credential-Access
T1071.001	Web Protocols	Command-And-Control
T1059.001	PowerShell	Execution
T1552.001	Credentials In Files	Credential-Access

Sources

Source	URL	Tier
Security News	https://thehackernews.com/2026/04/new-python-backdoor-uses-tunnelin...	T3
Detect New Vulnerabilities and Newly Supported Vendor Applications	https://docs.forescout.com/bundle/admin-guide-8-5-2/page/gitdoc-pla...	T3
CS0-003 CompTIA CyberSecurity Analyst CySA+ Certification Exam ...	https://www.marks4sure.com/cs0-003-comptia-cybersecurity-analyst-cy...	T3
Windows shortcut vulnerability exploited by nation-state hackers	https://www.facebook.com/groups/737897983014227/posts/3628185467318...	T3
[PDF] Locating Vulnerabilities Out Of Vendor Patches Automatically	http://media.blackhat.com/bh-us-10/whitepapers/Jeongwook_Oh/BlackHa...	T3

Generated 2026-04-30 19:01 UTC by TJS Security Command Center