INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-01 07:11 UTC

Deep#Door Python RAT Deploys Credential Stealer via Obfuscated Batch Loader on Windows

THREAT CAMPAIGN | **HIGH** | CVSS 8.0

SCC Item ID	SCC-CAM-2026-0251
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	8.0
Affected Products	Windows systems (specific versions not specified in available source data)
Published	2026-05-01
Discovery Source	Gemini

Executive Summary

Deep#Door is an active malware campaign deploying a Python-based Remote Access Trojan on Windows systems to steal browser passwords, cloud authentication tokens, SSH private keys, and Wi-Fi credentials. The malware disables host security controls and routes command-and-control traffic through a public tunneling service, making it harder to detect with standard network monitoring. Organizations with Windows endpoints are at risk of credential compromise that could enable downstream access to cloud environments, internal systems, and sensitive data.

Technical Analysis

Deep#Door uses an obfuscated batch file loader (T1059.003, CWE-506) to stage and execute a Python-based RAT on Windows hosts. The loader disables host-based security controls (T1562.001, CWE-693) before deploying a credential-harvesting implant that targets browser-stored passwords (T1552.001, CWE-522), cloud service authentication tokens (T1539), SSH private keys (T1552.004), and Wi-Fi credentials. Stolen data is exfiltrated via the bore[.]pub tunneling relay service (T1041, T1040), which obscures C2 traffic and bypasses conventional firewall and proxy inspection. Persistence is established via Windows startup mechanisms (T1547). Additional obfuscation techniques are applied to the payload (T1027, CWE-312). PowerShell is also used in the execution chain (T1059.001). No CVE identifier is assigned. Specific affected Windows versions are not confirmed in available source data. No vendor patch is applicable, this is malware, not a software vulnerability. Attributed threat actors and full campaign scope are unknown at time of analysis.

Source Confidence Note: Primary source is T3 (GBHackers); no T1 or T2 source has confirmed Deep#Door-specific technical details. Technical claims should be treated as preliminary pending confirmation from a higher-authority source.

Action Checklist

1. Containment: Identify any Windows hosts where bore.pub DNS resolution or outbound connections to bore.pub have occurred. Isolate those hosts from the network immediately. Block bore.pub and associated subdomains at DNS, proxy, and firewall egress points. Audit accounts with credentials stored in browsers or SSH key files on potentially exposed hosts.
2. Detection: Search endpoint logs and EDR telemetry for: execution of obfuscated batch files (.bat) spawning Python interpreter processes; Python processes making outbound connections on non-standard ports; registry or startup folder modifications (T1547 persistence artifacts); Windows Defender or AV service stop/disable events (Event ID 7036, 7040 for security service state changes); DNS queries or connections to bore.pub or bore[.]pub subdomains. Query SIEM for PowerShell execution (Event ID 4104) combined with batch file execution in close time proximity.
3. Eradication: On confirmed infected hosts: terminate malicious Python and batch processes; remove persistence entries from registry Run keys (HKCU\Software\Microsoft\Windows\CurrentVersion\Run), Startup folders, and scheduled tasks created by the malware; remove dropped Python scripts and batch loader files; re-enable any disabled security controls (Windows Defender, host firewall). Re-image hosts where full scope of access is uncertain.
4. Recovery: After remediation, rotate all credentials that were stored in browsers on affected hosts, including cloud service tokens and SSH private keys. Audit cloud provider access logs (AWS CloudTrail, Azure AD sign-in logs, GCP Audit Logs) for anomalous authentication events using credentials from affected hosts. Restore security tooling to verified-clean state and confirm EDR agent integrity before returning hosts to production.
5. Post-Incident: This campaign exploits credential storage in browsers and lack of network egress filtering for tunneling services. Control improvements to evaluate: enforce enterprise credential vaults (remove browser-saved passwords via policy); deploy DNS filtering to block known tunneling relay services; implement application allowlisting to prevent unauthorized Python interpreter execution; review egress firewall rules to restrict outbound connections to non-approved services.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to senior IR leadership, legal counsel, and privacy officer immediately if cloud provider audit logs (AWS CloudTrail, Azure AD, GCP) confirm that stolen credentials were used to authenticate to production systems, or if SSH private keys with access to regulated data environments (PCI, HIPAA, SOX) are confirmed compromised, triggering breach notification obligations.

Recovery Notes	After credential rotation and host re-imaging, maintain continuous monitoring of cloud provider authentication logs for a minimum of 30 days for anomalous sign-ins by previously affected accounts, as the Deep#Door RAT may have exfiltrated tokens with long expiration windows that were not immediately revoked. Validate that Windows Defender and host firewall services return to running state (Event ID 7036 showing 'running') and confirm EDR agent check-in from re-imaged hosts before returning them to production. Implement a 14-day watch period on DNS and proxy logs for any renewed bore.pub resolution attempts, which would indicate reinfection or a missed persistence mechanism.
Forensic Artifacts	Python RAT process memory dump (ProcDump -ma on python.exe PID) — contains in-memory C2 configuration, encryption keys, and collected credential buffers specific to the Deep#Door stealer module before process termination destroys them | Browser SQLite credential stores at '%LOCALAPPDATA%\Google\Chrome\User Data\Default\Login Data' and equivalent Edge/Firefox paths — hash-preserved copies establish which credential origins were accessible to the RAT's browser password harvesting routine | Sysmon EVTX log (Microsoft-Windows-Sysmon/Operational) with Event IDs 1, 3, 11, 13, and 22 — captures the full Deep#Door execution chain: batch loader spawn of python.exe (ID 1), bore.pub tunnel connection initiation (ID 3), dropped .py script file creation (ID 11), Run key persistence write (ID 13), and bore.pub DNS query (ID 22) | Windows System Event Log entries for Event IDs 7036 and 7040 — documents the exact timestamp Windows Defender (WinDefend) and host firewall (MpsSvc) were disabled by the Deep#Door batch loader, establishing the window during which the host operated without host-based protection | SSH directory artifact set from '%USERPROFILE%\.ssh\' including id_rsa/id_ed25519 private key files, known_hosts (identifying target servers), and authorized_keys — combined with Wi-Fi profile XML exports ('netsh wlan export profile key=clear') to establish full scope of credential classes exfiltrated by the stealer component

Per-Action IR Details

Containment — Identify any Windows hosts where bore.pub DNS resolution or outbound connections to bore.pub have occurred. Isolate those hosts from the network immediately. Block bore.pub and associated subdomains at DNS, proxy, and firewall egress points. Audit accounts with credentials stored in browsers or SSH key files on potentially exposed hosts.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST SC-7 (Boundary Protection), NIST SI-4 (System Monitoring), 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 Windows hosts without EDR: run 'Get-DnsClientCache | Where-Object {$_.Entry -like "*bore.pub*"}' in PowerShell to identify cached bore.pub resolutions. On network perimeter without a commercial proxy, push a null-route or sinkhole entry for bore.pub in your internal DNS server (e.g., Windows DNS or BIND) pointing to 0.0.0.0. Use Wireshark or 'netstat -anob' to enumerate active outbound connections from candidate hosts — look for established TCP sessions on non-standard ports (bore.pub commonly exposes tunneled ports in the 7000–9000 range). Block at the host firewall using 'netsh advfirewall firewall add rule name="Block bore.pub" dir=out action=block remoteip='.

Evidence: Before isolating the host, capture: (1) full DNS cache output via 'Get-DnsClientCache' or 'ipconfig /displaydns' to preserve bore.pub resolution history; (2) live network connection snapshot via 'netstat -anob > connections_snapshot.txt' to document active tunneled sessions; (3) browser credential store paths — Chrome: '%LOCALAPPDATA%\Google\Chrome\User Data\Default\Login Data', Edge: '%LOCALAPPDATA%\Microsoft\Edge\User Data\Default\Login Data', Firefox: '%APPDATA%\Mozilla\Firefox\Profiles*.default\logins.json' — hash these files with 'Get-FileHash' before any

remediation; (4) SSH private key files under '%USERPROFILE%\.ssh\' and any Wi-Fi profile XML exports via 'netsh wlan export profile folder=C:\evidence key=clear'; (5) Windows Firewall and proxy logs showing outbound connections to bore.pub resolved IPs.

Detection — Search endpoint logs and EDR telemetry for: execution of obfuscated batch files (.bat) spawning Python interpreter processes; Python processes making outbound connections on non-standard ports; registry or startup folder modifications (T1547 persistence artifacts); Windows Defender or AV service stop/disable events (Event ID 7036, 7040 for security service state changes); DNS queries or connections to bore.pub or bore[.]pub subdomains. Query SIEM for PowerShell execution (Event ID 4104) combined with batch file execution in close time proximity.

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-12 (Audit Record Generation), NIST IR-5 (Incident Monitoring), CIS 8.2 (Collect Audit Logs), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Deploy Sysmon with SwiftOnSecurity's config (https://github.com/SwiftOnSecurity/sysmon-config) to capture Event ID 1 (Process Create) for cmd.exe or powershell.exe spawning python.exe or pythonw.exe, and Event ID 3 (Network Connection) for python.exe initiating outbound TCP. Without SIEM, parse Sysmon logs locally: 'Get-WinEvent -LogName "Microsoft-Windows-Sysmon/Operational" | Where-Object {$_.Message -like "*python*"}'. Use the Sigma rule for T1547.001 (Registry Run Keys) and convert to a PowerShell query against saved Sysmon EVTX logs using the free Chainsaw tool (https://github.com/WithSecureLabs/chainsaw). For service tampering, query Windows System Event Log: 'Get-WinEvent -FilterHashtable @{LogName="System"; Id=7036,7040} | Where-Object {$_.Message -like "*Windows Defender*" -or $_.Message -like "*Security Center*"}'.

Evidence: Collect before any host changes: (1) Sysmon Event ID 1 records showing parent-child process chain where cmd.exe or wscript.exe spawned a .bat file which then spawned python.exe — export with 'wevtutil epl Microsoft-Windows-Sysmon/Operational sysmon_backup.evtx'; (2) Windows Security Event Log Event ID 4688 (Process Creation with command line auditing enabled) filtered on python.exe and obfuscated batch file names — look for high-entropy filenames or base64 strings in command-line arguments; (3) System Event Log Event IDs 7036 and 7040 timestamped near the python.exe execution, showing Windows Defender (WinDefend) or MpsSvc (Windows Firewall) transitioning to stopped state; (4) PowerShell Script Block Logging (Event ID 4104) from 'Microsoft-Windows-PowerShell/Operational' log for any invocation launching or downloading the batch loader; (5) DNS debug log or Sysmon Event ID 22 (DNS Query) showing queries to bore.pub or *.bore.pub subdomains correlated by timestamp with python.exe network events (Sysmon Event ID 3).

Eradication — On confirmed infected hosts: terminate malicious Python and batch processes; remove persistence entries from registry Run keys (HKCU\Software\Microsoft\Windows\CurrentVersion\Run), Startup folders, and scheduled tasks created by the malware; remove dropped Python scripts and batch loader files; re-enable any disabled security controls (Windows Defender, host firewall). Re-image hosts where full scope of access is uncertain.

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 CM-7 (Least Functionality), CIS 2.3 (Address Unauthorized Software), CIS 4.6 (Securely Manage Enterprise Assets and Software)

Compensating: Before terminating processes, capture a full memory image of the python.exe process using ProcDump: 'procdump.exe -ma python_memdump.dmp' to preserve in-memory RAT configuration, C2 keys, and any injected shellcode. Remove Run key persistence: 'Remove-ItemProperty -Path "HKCU:\Software\Microsoft\Windows\CurrentVersion\Run" -Name '. Enumerate and remove scheduled tasks: 'Get-ScheduledTask | Where-Object {$_.TaskPath -notlike "\Microsoft*"} | Export-Csv suspicious_tasks.csv' then 'Unregister-ScheduledTask -TaskName -Confirm:$false'. Re-enable Windows Defender: 'Set-MpPreference -DisableRealtimeMonitoring $false' and restore firewall: 'netsh advfirewall set allprofiles state on'. For startup folder

persistence, check '%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup' and '%ProgramData%\Microsoft\Windows\Start Menu\Programs\Startup' for any .bat, .vbs, or .pyw files dropped by the campaign.

Evidence: Capture before eradication: (1) Full registry export of HKCU\Software\Microsoft\Windows\CurrentVersion\Run and HKLM\Software\Microsoft\Windows\CurrentVersion\Run via 'reg export HKCU\Software\Microsoft\Windows\CurrentVersion\Run run_keys_hkcu.reg' to document the Deep#Door persistence entry name and target path; (2) Scheduled task XML exports via 'schtasks /query /fo XML /v > all_tasks.xml' — Deep#Door-created tasks will reference python.exe or the obfuscated batch loader path; (3) Hash (SHA-256) and copy of all .bat files found in %TEMP%, %APPDATA%, %USERPROFILE%\Downloads, and any path referenced in Run keys before deletion; (4) Hash and copy of all .py and .pyc files dropped outside of legitimate Python installation directories (e.g., outside C:\Python3x\); (5) Windows Security Event Log Event ID 4657 (Registry value modification) if object access auditing is enabled, showing the timestamp the malware wrote its Run key entry.

Recovery — After remediation, rotate all credentials that were stored in browsers on affected hosts, including cloud service tokens and SSH private keys. Audit cloud provider access logs (AWS CloudTrail, Azure AD sign-in logs, GCP Audit Logs) for anomalous authentication events using credentials from affected hosts. Restore security tooling to verified-clean state and confirm EDR agent integrity before returning hosts to production.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST IA-5 (Authenticator Management), NIST AC-2 (Account Management), NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 5.2 (Use Unique Passwords), CIS 6.2 (Establish an Access Revoking Process), CIS 6.3 (Require MFA for Externally-Exposed Applications)

Compensating: For SSH key rotation without an enterprise vault: generate new Ed25519 key pairs on a clean host ('ssh-keygen -t ed25519'), push new public keys to all servers where the compromised user had access ('ssh-copy-id'), then immediately revoke old keys by removing the compromised public key from each server's '~/.ssh/authorized_keys'. For cloud token revocation without enterprise IAM tooling: AWS — 'aws iam delete-access-key --access-key-id ' and 'aws iam list-access-keys' to identify all keys for affected users; Azure — revoke refresh tokens for affected accounts via 'Revoke-AzureADUserAllRefreshToken -ObjectId ' using Azure AD PowerShell module (free); GCP — 'gcloud iam service-accounts keys delete '. Query AWS CloudTrail free tier for the 90-day default retention window filtering on 'sourceIPAddress' not matching known corporate egress IPs for affected IAM identities.

Evidence: Before rotating credentials, document: (1) Chrome/Edge SQLite Login Data file — copy and query with 'sqlite3 "Login Data" "SELECT origin_url, username_value, date_created FROM logins;"' to produce an inventory of exactly which credential origins the RAT had access to (passwords are encrypted but the URL list establishes breach scope); (2) AWS CloudTrail 'LookupEvents' API results filtered on the affected IAM user's access key ID for the 30 days prior to detection, exported to JSON for anomalous API call review; (3) Azure AD Sign-In Logs filtered on the affected UPN for sign-ins from non-corporate IPs or unfamiliar user agents during the infection window — export via Microsoft Entra portal or 'Get-AzureADAuditSignInLogs'; (4) SSH known_hosts file at '%USERPROFILE%\.ssh\known_hosts' documenting which servers the compromised private key had authenticated to; (5) Wi-Fi profile exports ('netsh wlan export profile') to identify which network PSKs were exposed, requiring AP password rotation.

Post-Incident — This campaign exploits credential storage in browsers and lack of network egress filtering for tunneling services. Control improvements to evaluate: enforce enterprise credential vaults (remove browser-saved passwords via policy); deploy DNS filtering to block known tunneling relay services; implement application allowlisting to prevent unauthorized Python interpreter execution; review egress firewall rules to restrict outbound connections to non-approved services.

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-7 (Software, Firmware, and Information Integrity), NIST CM-7 (Least Functionality), NIST SC-20 (Secure Name/Address Resolution Service), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 2.3 (Address Unauthorized Software), CIS 4.2 (Establish and Maintain a Secure Configuration Process for Network Infrastructure), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Browser credential storage: enforce via Group Policy — navigate to 'Computer Configuration > Administrative Templates > Google Chrome > Password Manager' and set 'Enable saving passwords to the password manager' to Disabled; equivalent Edge GPO path is 'Microsoft Edge > Password Manager and Protection'. Python interpreter allowlisting without commercial tooling: use Windows Software Restriction Policies (SRP) or AppLocker (available on Windows Pro/Enterprise) to deny execution of python.exe and pythonw.exe from all paths except explicitly approved directories — 'New-AppLockerPolicy -RuleType Publisher -FilePath "C:\Python3x\python.exe" -User "Domain Users" -Action Deny'. DNS filtering: deploy Pi-hole (free, https://pi-hole.net) as internal DNS resolver with a blocklist entry for bore.pub and reference the public tunneling service blocklist maintained by the community (e.g., 'hagezi/dns-blocklists' on GitHub). For egress filtering, implement a default-deny outbound firewall rule allowing only ports 80, 443, and 53 to approved resolvers — block all other outbound TCP/UDP at the perimeter.

Evidence: For the post-incident lessons learned report, compile: (1) Timeline reconstruction from Sysmon Event ID 1 and Event ID 3 logs showing the full attack chain — batch loader execution timestamp → python.exe spawn → first bore.pub DNS query → first outbound tunnel connection — to establish dwell time; (2) Inventory of all browser credential origins exposed (from SQLite Login Data query) to quantify breach scope and support breach notification assessment; (3) List of all cloud API calls made during the infection window from CloudTrail/Azure AD/GCP Audit Logs to determine if compromised tokens were actively used for lateral movement to cloud resources; (4) Registry and scheduled task artifacts preserved during eradication showing persistence mechanism details (key name, target path, trigger) for detection rule development; (5) Network flow or firewall log summary of all bore.pub tunnel sessions (timestamps, duration, bytes transferred) to estimate data exfiltration volume for impact assessment.

Detection Guidance

Primary behavioral indicators: (1) batch file execution spawning a Python interpreter, particularly from user-writable directories (AppData, Temp); (2) Python processes initiating outbound TCP connections, especially to bore.pub or subdomains of bore.pub; (3) security service disable events, Windows Event IDs 7036 and 7040 for services including WinDefend, MpsSvc; (4) registry modifications to HKCU or HKLM Run keys by non-standard processes; (5) file system access to browser credential stores (e.g., Chrome Login Data, Firefox logins.json), SSH key directories, and Wi-Fi credential files by unexpected processes. Network indicators: DNS queries or connections to bore[.]pub, this domain is a public TCP tunneling relay and has limited legitimate enterprise use. SIEM correlation: chain batch execution (Event ID 4688 with .bat extension) + Python process spawn + outbound connection + security service state change within a short time window. Note: specific file hashes and IP-based IOCs are not available from current source data; detections should rely on behavioral patterns until higher-fidelity IOCs are published.

Indicators of Compromise

Type	Value	Context	Confidence
DOMAIN	`bore.pub`	Public TCP tunneling relay service used by Deep#Door to route C2 traffic and exfiltrate stolen credentials; limited legitimate enterprise use	MEDIUM

Framework Mappings

MITRE-ATTACK

- **T1547** — Boot or Logon Autostart Execution
- **T1059.003** — Windows Command Shell
- **T1552.001** — Credentials In Files
- **T1562.001** — Disable or Modify Tools
- **T1552.004** — Private Keys
- **T1539** — Steal Web Session Cookie
- **T1027** — Obfuscated Files or Information
- **T1040** — Network Sniffing
- **T1041** — Exfiltration Over C2 Channel
- **T1059.001** — PowerShell

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
T1547	Boot or Logon Autostart Execution	Persistence
T1059.003	Windows Command Shell	Execution
T1552.001	Credentials In Files	Credential-Access
T1562.001	Disable or Modify Tools	Defense-Evasion
T1552.004	Private Keys	Credential-Access
T1539	Steal Web Session Cookie	Credential-Access
T1027	Obfuscated Files or Information	Defense-Evasion
T1040	Network Sniffing	Credential-Access
T1041	Exfiltration Over C2 Channel	Exfiltration
T1059.001	PowerShell	Execution

Sources

Source	URL	Tier
gemini	https://gbhackers.com/deepdoor-stealer-targets-passwords-tokens-ssh...	T3
SamsStealer: Unveiling the Information Stealer Targeting Windows ...	https://www.cyfirma.com/research/samsstealer-unveiling-the-informat...	T3
Phantom Stealer Campaign Abuses ISO Mounting to Compromise ...	https://coesecurity.com/phantom-stealer-campaign-abuses-iso-mountin...	T3
Attackers use Windows App-V scripts to slip infostealer past ...	https://www.helpnetsecurity.com/2026/01/27/malware-delivery-via-win...	T3
Disrupting Lumma Stealer: Microsoft leads global action against ...	https://blogs.microsoft.com/on-the-issues/2025/05/21/microsoft-lead...	T1

DISCLAIMER

Generated 2026-05-01 07:11 UTC by TJS Security Command Center