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

EtherRAT Weaponizes Ethereum Smart Contracts and GitHub Facades to Hunt Enterprise Administrators

THREAT CAMPAIGN | **CRITICAL** | CVSS 9.5

SCC Item ID	SCC-CAM-2026-0243
Type	Threat Campaign
Severity	CRITICAL
CVSS Base Score	9.5
Affected Products	Windows administrative tools (PsExec, AzCopy, Sysmon, LAPS, Kusto Explorer, ProcDump, Autoruns, Process Explorer, Process Monitor, TCPView, RAMMap, WinDbg, DebugView, BgInfo, Disk2vhd, Windows ADK, RSAT, IIS Crypto, Dameware, SecureCRT, SuperPuTTY, ScreenConnect, Bitvise SSH Client, TeraTerm, FSLogix, AppLocker, PRTG Network Monitor, Beyond Compare, KDiff3, VMware Tools, and others); GitHub platform; Ethereum RPC infrastructure; Node.js runtime
Published	2026-04-30T07:30:00
Discovery Source	Rss

Executive Summary

An active campaign is deploying a JavaScript remote access trojan that impersonates 44 widely used administrative tools, targeting enterprise administrators, DevOps engineers, and security analysts through poisoned search results. The malware uses Ethereum blockchain infrastructure to resolve its command-and-control server, making traditional domain-blocking and law enforcement takedown ineffective. Every successful infection lands on a privileged account, creating immediate risk of lateral movement, credential theft, and enterprise-wide compromise.

Technical Analysis

EtherRAT is a JavaScript-based RAT distributed through a dual-stage GitHub distribution chain and SEO-poisoned search results. The campaign impersonates 44 legitimate Windows administrative tools including PsExec, AzCopy, Sysmon, LAPS, ProcDump, RSAT, and others commonly used by IT and security staff. Victims are lured via typosquatted or cloned GitHub repositories and fake download pages that serve trojanized Node.js packages or scripts. Post-execution, the malware queries an Ethereum smart contract via RPC calls to resolve its current C2 address, a technique related to EtherHiding, bypassing DNS-based detection and domain takedown. Because the C2 pointer lives on the blockchain, it cannot be disrupted by sinkholing or registrar

action. Relevant CWEs: CWE-829 (inclusion of functionality from untrusted control sphere), CWE-693 (protection mechanism failure), CWE-494 (download of code without integrity check). MITRE coverage includes T1568 (dynamic resolution), T1102/T1102.002 (web service C2), T1036/T1036.005 (masquerading), T1059.007 (JavaScript execution), T1204.002 (malicious file execution), T1608.001/T1608.004 (staged capabilities via GitHub), T1078.002 (valid privileged accounts post-compromise), T1021 (lateral movement), T1027 (obfuscation), T1071/T1071.001 (application layer protocol), T1195/T1195.002 (supply chain compromise). No CVE assigned. No vendor patch applicable, the malware exploits trust in legitimate platforms, not a software vulnerability. Tracked by Atos Threat Research Center; parallel tracking by KISA/KrCERT. Threat actors remain unattributed.

Action Checklist

1. Containment: Block Ethereum RPC endpoints at the network perimeter. Deny outbound connections to common public Ethereum nodes (infura.io, cloudflare-eth.com, mainnet.infura.io, eth.llamarpc.com) for all endpoints that have no legitimate blockchain development need. Alert on any outbound traffic to TCP port 8545 or 8546 (Ethereum JSON-RPC defaults) from non-developer systems.
2. Detection: Search endpoint logs and EDR telemetry for: (1) Node.js processes spawned from user download directories or temp folders, (2) GitHub repository clones or ZIP downloads of tools matching the 44 impersonated names followed within minutes by JavaScript execution, (3) outbound HTTPS to Ethereum RPC providers from endpoints not running blockchain software, (4) new or unsigned scheduled tasks and services created by node.exe or wscript.exe. Review proxy logs for HTTP POST requests containing JSON-RPC method calls (eth_call, eth_getLogs) from non-developer hosts.
3. Eradication: Remove any identified trojanized tool installations. Verify the cryptographic hash of all 44 impersonated tools against official vendor sources (Microsoft Sysinternals, Microsoft Azure, vendor download pages). Revoke and rotate credentials for any administrator account that downloaded or executed unverified tools. Disable and remove persistence mechanisms (scheduled tasks, services) created by the malware. Quarantine affected systems before returning them to production.
4. Recovery: Re-image compromised endpoints rather than attempting in-place cleanup, given the privileged account targeting and lateral movement risk. Reissue credentials for all accounts confirmed or suspected to have run on the compromised system. Restore from known-good backups if file system integrity cannot be confirmed. Monitor reinstated systems for 30 days for recurrence of Ethereum RPC outbound traffic or anomalous JavaScript execution.
5. Post-Incident: Implement a verified software sourcing policy: all administrative tools must be downloaded from official vendor sites with hash verification before execution. Enforce application allowlisting (AppLocker or Windows Defender Application Control) to block unsigned or unrecognized JavaScript and Node.js execution on administrative workstations. Conduct search-result hygiene training for IT and security staff targeting the specific tactic of SEO-poisoned results for tool names. Review GitHub download workflows for any automated pipelines pulling from unverified repositories.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE

Escalation Criteria	Escalate immediately to CISO, legal counsel, and external IR retainer if any compromised administrator account has Active Directory Domain Admin, Azure Global Administrator, or equivalent cloud tenant admin privileges; if evidence exists of lateral movement (Event ID 4648 or 4624 Type 3 logons from the compromised host to additional systems); or if regulated data (PII, PHI, PCI-scoped cardholder data) was accessible to the compromised account, triggering breach notification obligations under GDPR (72-hour), HIPAA, or applicable state breach notification laws.
Recovery Notes	Re-image all confirmed-compromised administrative workstations from a known-good baseline image rather than attempting remediation in place — EtherRAT's JavaScript payload executing under a privileged account context makes it impossible to rule out secondary persistence mechanisms without a clean rebuild. After reissuing all credentials, specifically verify that Azure service principals, AzCopy SAS tokens, and any API keys accessible from the compromised session have been rotated, as these are high-value targets for a RAT specifically impersonating Azure tooling like AzCopy. Monitor all reinstated systems for 30 days using Sysmon Event ID 1 and 3 alerts scoped to node.exe and Ethereum RPC destinations, and query the identified smart contract address on Etherscan weekly to detect any infrastructure updates the threat actor pushes to the same contract.
Forensic Artifacts	Ethereum smart contract address and JSON-RPC call logs — the specific contract address hardcoded in the EtherRAT JavaScript payload, recoverable from the malicious .js file in the download directory and from heap memory of a running node.exe process; query this address on Etherscan to retrieve the C2 IP the contract returned, providing permanent campaign infrastructure attribution | GitHub repository download artifacts — browser history entries (Chrome: C:\Users*\AppData\Local\Google\Chrome\User Data\Default\History; Edge equivalent path) showing navigation to a fake GitHub repository page for one of the 44 impersonated tools, plus the downloaded ZIP or cloned directory in %USERPROFILE%\Downloads\ containing package.json and the malicious JavaScript entry point | Sysmon Event ID 1 (Process Create) and Event ID 3 (Network Connection) records — node.exe process creation events showing parent process (browser or file explorer), full command line, working directory in a non-standard path, and subsequent network connection events to infura.io, cloudflare-eth.com, or eth.llamarpc.com with destination port 443 | Scheduled task and service persistence XML — full XML export of any scheduled tasks or Windows services registered by node.exe or wscript.exe, captured from Task Scheduler operational log (Event ID 106) and the registry at HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tasks\ and HKLM\SYSTEM\CurrentControlSet\Services\ | Active Directory authentication logs for the compromised privileged account — Windows Security Event Log Event ID 4624 (Successful Logon), 4648 (Logon with Explicit Credentials), and 4728/4732/4756 (Group Membership Changes) scoped to the administrator account that executed the trojanized tool, covering the window from first node.exe execution forward, to establish the full lateral movement and privilege escalation timeline

Per-Action IR Details

Containment — Block Ethereum RPC endpoints at the network perimeter. Deny outbound connections to common public Ethereum nodes (infura.io, cloudflare-eth.com, mainnet.infura.io, eth.llamarpc.com) for all endpoints that have no legitimate blockchain development need. Alert on any outbound traffic to TCP port 8545 or 8546 (Ethereum JSON-RPC defaults) from non-developer systems.

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)

Compensating: On Windows endpoints without EDR: run 'netstat -ano | findstr ":8545 :8546"' to identify active JSON-RPC sessions. Deploy a Windows Firewall GPO rule blocking outbound TCP 8545/8546 enterprise-wide immediately. For DNS-layer blocking of infura.io, cloudflare-eth.com, eth.llamarpc.com without a commercial DNS filter, add entries to the enterprise DNS server's response policy zone (RPZ) or, for standalone systems, push a hosts file entry via GPO mapping these domains to 0.0.0.0. Use Wireshark or Windows built-in 'netsh trace start capture=yes' on suspected hosts to capture JSON-RPC POST bodies containing 'eth_call' or 'eth_getLogs' method strings as confirmation of active C2 beacon activity.

Evidence: Before implementing blocks, capture full packet captures (PCAPs) of outbound HTTPS traffic to Ethereum RPC providers — specifically look for HTTP POST bodies to /v3/ endpoints on infura.io or mainnet.infura.io containing JSON-RPC payloads with 'method':'eth_call' or 'method':'eth_getLogs'; these calls are how EtherRAT retrieves its C2 server address from a smart contract. Export DNS query logs showing resolution of infura.io, cloudflare-eth.com, or eth.llamarpc.com from administrative workstations. Preserve Windows Firewall logs (C:\Windows\System32\LogFiles\Firewall\pfirewall.log) showing destination IPs and ports prior to rule enforcement. Timestamp all captures — the blockchain query sequence immediately precedes C2 callback and establishes the causal chain for forensic reporting.

Detection — Search endpoint logs and EDR telemetry for: (1) Node.js processes spawned from user download directories or temp folders, (2) GitHub repository clones or ZIP downloads of tools matching the 44 impersonated names followed within minutes by JavaScript execution, (3) outbound HTTPS to Ethereum RPC providers from endpoints not running blockchain software, (4) new or unsigned scheduled tasks and services created by node.exe or wscript.exe. Review proxy logs for HTTP POST requests containing JSON-RPC method calls (eth_call, eth_getLogs) from non-developer hosts.

NIST Phase: Detection Analysis

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

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

Compensating: Without EDR, deploy Sysmon with a config that captures Event ID 1 (Process Create) filtering on Image paths containing 'node.exe' with ParentImage in '%USERPROFILE%\Downloads', '%TEMP%', or '%APPDATA%'; and Event ID 11 (File Create) for .js files created in those directories. Use the following PowerShell one-liner to scan all endpoints for suspicious scheduled tasks created by node.exe or wscript.exe: 'Get-ScheduledTask | Where-Object {$_.Actions.Execute -match "node|wscript|cscript"} | Select-Object TaskName, TaskPath, @{N="Execute";E={$_.Actions.Execute}} | Export-Csv tasks_audit.csv'. For GitHub ZIP download correlation, query proxy logs (Squid, BlueCoat, or Windows IIS proxy) for GET requests to codeload.github.com/*/zip/* where the repository name matches any of the 44 impersonated tool names (PsExec, AzCopy, Sysmon, LAPS, etc.) followed within a 5-minute window by outbound HTTPS to an Ethereum RPC provider from the same source IP. Write a Sigma rule detecting this exact sequence: parent_process=browser OR download_manager → child_process=node.exe → network_destination=infura.io|cloudflare-eth.com.

Evidence: Pull Windows Security Event Log Event ID 4688 (Process Creation) with command-line logging enabled, filtering on 'node.exe' spawned from non-standard paths such as C:\Users*\Downloads*, C:\Users*\AppData\Local\Temp*, or any path containing a tool name from the 44 impersonated list (e.g., 'PsExec', 'AzCopy', 'Sysmon'). Collect Sysmon Event ID 3 (Network Connection) records where the Image field is node.exe and the DestinationHostname resolves to any Ethereum RPC provider. Export browser download history (Chrome: C:\Users*\AppData\Local\Google\Chrome\User Data\Default\History; Edge: C:\Users*\AppData\Local\Microsoft\Edge\User Data\Default\History) to correlate SEO-poisoned search result clicks with subsequent downloads. Collect Windows Task Scheduler operational log (Microsoft-Windows-TaskScheduler/Operational, Event ID 106 — Task Registered, Event ID 200 — Task Started) and filter for tasks registered within the same time window as node.exe execution. Capture the contents of %APPDATA%\npm, %APPDATA%\node_modules, and any package.json files found in download directories, as these will reveal the trojanized package structure.

Eradication — Remove any identified trojanized tool installations. Verify the cryptographic hash of all 44 impersonated tools against official vendor sources (Microsoft Sysinternals, Microsoft Azure, vendor download pages). Revoke and rotate credentials for any administrator account that downloaded or executed unverified tools. Disable and remove persistence mechanisms (scheduled tasks, services) created by the malware. Quarantine affected systems before returning them to production.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST IR-4 (Incident Handling), NIST SI-2 (Flaw Remediation), NIST SI-7 (Software, Firmware, and Information Integrity), NIST AC-2 (Account Management), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.3 (Address Unauthorized Software), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: For hash verification of all 44 impersonated tools without a dedicated file integrity tool: use PowerShell 'Get-FileHash -Algorithm SHA256 ' against every installed instance and compare against hashes published on Microsoft Sysinternals (https://learn.microsoft.com/en-us/sysinternals/), Microsoft Azure documentation, and individual vendor download pages. Script this at scale with: 'Get-ChildItem -Path C:\ -Recurse -Include psexec.exe,azcopy.exe,sysmon.exe,procdump.exe,autoruns.exe -ErrorAction SilentlyContinue | ForEach-Object { [PSCustomObject]@{Path=$_.FullName; Hash=(Get-FileHash $_.FullName -Algorithm SHA256).Hash} } | Export-Csv hash_audit.csv'. For credential revocation without a PAM tool: use Active Directory PowerShell module to force password reset ('Set-ADAccountPassword -Reset') and disable accounts pending reissuance for all admin accounts confirmed on affected systems. Remove trojanized scheduled tasks with 'schtasks /delete /tn /f' and audit residual registry run keys at HKCU\Software\Microsoft\Windows\CurrentVersion\Run and HKLM\Software\Microsoft\Windows\CurrentVersion\Run for node.exe or wscript.exe entries.

Evidence: Before removing any persistence mechanisms, capture complete registry exports of HKCU\Software\Microsoft\Windows\CurrentVersion\Run, HKLM\Software\Microsoft\Windows\CurrentVersion\Run, HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce, and HKLM\SYSTEM\CurrentControlSet\Services for any service entries pointing to node.exe or JavaScript files. Export the full scheduled task XML for every task identified as malicious using 'schtasks /query /fo XML /tn '. Image or copy the trojanized tool directory (the full folder as downloaded from the fake GitHub repository) before deletion — this preserves the package.json, the malicious .js payload file, and any embedded Ethereum smart contract ABI or wallet address hardcoded for C2 resolution. Collect Windows Security Event Log Event ID 4720 (User Account Created), 4728/4732/4756 (Member Added to Security Group), and 4648 (Logon with Explicit Credentials) for the period following confirmed node.exe execution to identify any accounts created or elevated by the RAT during its access window.

Recovery — Re-image compromised endpoints rather than attempting in-place cleanup, given the privileged account targeting and lateral movement risk. Reissue credentials for all accounts confirmed or suspected to have run on the compromised system. Restore from known-good backups if file system integrity cannot be confirmed. Monitor reinstated systems for 30 days for recurrence of Ethereum RPC outbound traffic or anomalous JavaScript execution.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST CP-10 (System Recovery and Reconstitution), NIST SI-7 (Software, Firmware, and Information Integrity), NIST AC-2 (Account Management), CIS 5.1 (Establish and Maintain an Inventory of Accounts), CIS 5.2 (Use Unique Passwords), CIS 7.3 (Perform Automated Operating System Patch Management)

Compensating: For a 2-person team without automated imaging infrastructure: prioritize re-imaging administrative workstations first given EtherRAT's explicit targeting of administrator, DevOps, and security analyst accounts — a compromised privileged endpoint represents the highest lateral movement risk. Use Windows Deployment Services (WDS) or a pre-staged WinPE USB with a known-good base image. Before reinstating any account, verify Active Directory group memberships were not modified during the compromise window by running 'Get-ADGroupMember -Identity "Domain Admins" -Recursive' and comparing against a pre-incident baseline. For 30-day post-recovery monitoring without EDR: deploy Sysmon on reinstated systems and forward Event ID 1 (Process Create for node.exe) and Event ID 3 (Network Connection to Ethereum RPC destinations) to a central Windows Event Forwarding (WEF)

collector using a free subscription. Use osquery with a scheduled query checking for node.exe in non-standard paths every 15 minutes: 'SELECT pid, name, path, cmdline FROM processes WHERE name = "node.exe" AND path NOT LIKE "C:\\Program Files\\nodejs\\%"'.

Evidence: Before re-imaging, acquire a full forensic disk image using FTK Imager (free) or 'dd' via WinPE to preserve all file system artifacts for post-incident analysis — this is critical because EtherRAT's JavaScript payload and the Ethereum smart contract address used for C2 resolution are forensic evidence needed to track campaign infrastructure. Capture a memory dump using ProcDump ('procdump.exe -ma -o node.exe ') for any running node.exe processes before shutdown, as the decrypted C2 address resolved from the blockchain will be present in heap memory and is not recoverable post-process-termination. Document all account SIDs present in the NTUSER.DAT hive of the compromised profile (accessible via Registry Editor or 'reg export HKEY_USERS') to ensure complete coverage of accounts requiring credential rotation, including cached credentials for cloud services like Azure (AzCopy tokens) that may have been accessible to the RAT.

Post-Incident — Implement a verified software sourcing policy: all administrative tools must be downloaded from official vendor sites with hash verification before execution. Enforce application allowlisting (AppLocker or Windows Defender Application Control) to block unsigned or unrecognized JavaScript and Node.js execution on administrative workstations. Conduct search-result hygiene training for IT and security staff targeting the specific tactic of SEO-poisoned results for tool names. Review GitHub download workflows for any automated pipelines pulling from unverified repositories.

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-2 (Flaw Remediation), NIST SI-7 (Software, Firmware, and Information Integrity), NIST CM-7 (Least Functionality), 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 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: For AppLocker enforcement targeting EtherRAT's specific delivery mechanism: create an AppLocker Executable Rule blocking node.exe execution from %USERPROFILE%\Downloads*, %TEMP%*, and %APPDATA%* while allowing it only from C:\Program Files\nodejs\ if Node.js is a legitimate business tool; separately, create a Script Rule blocking .js file execution by wscript.exe or cscript.exe system-wide on administrative workstations via GPO (Computer Configuration → Windows Settings → Security Settings → Application Control Policies). For GitHub workflow auditing without a commercial SAST tool: run 'grep -r "github.com" .github/workflows/ Makefile Jenkinsfile *.sh *.ps1' across all CI/CD repositories to identify any pipeline steps cloning or downloading from GitHub without pinned commit SHAs or hash verification. For SEO-poisoning awareness training, build a tabletop exercise specifically using the 44 tool names from this campaign — have staff demonstrate how they would download PsExec or AzCopy and verify it is authentic before running it, using only the Microsoft Sysinternals page or official Azure documentation as the source.

Evidence: For the lessons-learned record, document the exact GitHub repository URLs and account names used in this campaign (if identified during investigation) so they can be submitted to GitHub Trust & Safety for takedown and added to threat intelligence platforms as indicators. Preserve the Ethereum smart contract address used for C2 resolution — this is a permanent, immutable blockchain record that can be queried indefinitely via Etherscan to track future campaign activity; record the contract address and the specific function call or event log the malware used to retrieve the C2 IP. Compile a diff of all scheduled tasks, services, and registry run keys present on affected systems versus a clean baseline image — this delta is the definitive persistence artifact inventory for this campaign and should be converted into a YARA rule targeting the task names or service descriptions used by EtherRAT for use in future threat hunting.

Detection Guidance

Primary behavioral indicators: Node.js (node.exe) execution originating from user download directories, Temp, or AppData folders on systems where Node.js has no approved business purpose. Outbound HTTPS

connections to Ethereum JSON-RPC providers (infura.io, cloudflare-eth.com, llamarpc.com, ankr.com, or similar) from non-developer endpoints, particularly containing JSON bodies with eth_call or eth_getLogs methods. Scheduled task or service creation by node.exe, wscript.exe, or cscript.exe. Process lineage showing a browser spawning a downloaded executable that then spawns Node.js. Secondary indicators: GitHub repository clones of tool names matching the 44 impersonated utilities from accounts with no commit history or recent creation dates. Files named identically to Sysinternals or Microsoft admin tools but located outside expected installation paths (e.g., C:\Users*\Downloads\PsExec.exe). Windows Event ID 4698 (scheduled task created) or 7045 (new service installed) generated by unusual parent processes. SIEM query starting point: index=* (process_name='node.exe' OR process_name='wscript.exe') parent_path IN ('*\Downloads*','*\AppData\Temp*') | join src_ip [search index=network dest_host IN ('infura.io','cloudflare-eth.com','llamarpc.com')]. Adjust for your SIEM syntax. Confidence in Ethereum RPC traffic as an indicator is high for non-developer environments; treat any such traffic as a priority alert.

Indicators of Compromise

Type	Value	Context	Confidence
DOMAIN	`infura.io`	Legitimate Ethereum RPC provider abused for C2 resolution via smart contract queries — flag anomalous outbound connections from non-developer endpoints, not the domain itself	MEDIUM
DOMAIN	`cloudflare-eth.com`	Legitimate Ethereum RPC endpoint potentially abused for blockchain-based C2 resolution — treat outbound connections from administrative workstations as suspicious	MEDIUM
URL	`https://github.com/[spoofed-repo]`	Campaign uses cloned or typosquatted GitHub repositories impersonating legitimate tool distributors — specific repository URLs not publicly confirmed in available sources; flag GitHub downloads of tool names matching the 44 impersonated utilities from repos with no commit history	LOW

Framework Mappings

MITRE-ATTACK

- **T1102.002** — Bidirectional Communication
- **T1204.002** — Malicious File
- **T1036.005** — Match Legitimate Resource Name or Location
- **T1021** — Remote Services
- **T1078.002** — Domain Accounts

- **T1608.001** — Upload Malware
- **T1543** — Create or Modify System Process
- **T1568** — Dynamic Resolution
- **T1027** — Obfuscated Files or Information
- **T1583.001** — Domains
- **T1059.007** — JavaScript
- **T1102** — Web Service
- **T1071** — Application Layer Protocol
- **T1195** — Supply Chain Compromise
- **T1036** — Masquerading
- **T1608.004** — Drive-by Target
- **T1195.002** — Compromise Software Supply Chain
- **T1071.001** — Web Protocols
- **T1566** — Phishing

NIST-800-53R5

- **AC-17** — Remote Access
- **AC-3** — Access Enforcement
- **CM-7** — Least Functionality
- **IA-2** — Identification and Authentication (Organizational Users)
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **SA-9** — External System Services
- **SR-2** — Supply Chain Risk Management Plan
- **SR-3** — Supply Chain Controls and Processes
- **SI-7** — Software, Firmware, and Information Integrity
- **AT-2** — Literacy Training and Awareness
- **SI-8** — Spam Protection
- **CM-3** — Configuration Change Control

OWASP-TOP10-2021

- **A08:2021** — Software and Data Integrity Failures

CIS-V8

- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1102.002	Bidirectional Communication	Command-And-Control
T1204.002	Malicious File	Execution
T1036.005	Match Legitimate Resource Name or Location	Defense-Evasion
T1021	Remote Services	Lateral-Movement
T1078.002	Domain Accounts	Defense-Evasion
T1608.001	Upload Malware	Resource-Development
T1543	Create or Modify System Process	Persistence
T1568	Dynamic Resolution	Command-And-Control
T1027	Obfuscated Files or Information	Defense-Evasion
T1583.001	Domains	Resource-Development
T1059.007	JavaScript	Execution
T1102	Web Service	Command-And-Control
T1071	Application Layer Protocol	Command-And-Control
T1195	Supply Chain Compromise	Initial-Access
T1036	Masquerading	Defense-Evasion
T1608.004	Drive-by Target	Resource-Development
T1195.002	Compromise Software Supply Chain	Initial-Access
T1071.001	Web Protocols	Command-And-Control
T1566	Phishing	Initial-Access

Sources

Source	URL	Tier
Security News	https://thehackernews.com/2026/04/etherrat-distribution-spoofing.html	T3
Everything You Need to Know About Windows Administrative Tools ...	https://www.serverwatch.com/guides/windows-server-administrative-to...	T3

Source	URL	Tier
Geo-Joy/Hunting-the-Ethereum-Smart-Contract-Color-inspired ...	https://github.com/Geo-Joy/Hunting-the-Ethereum-Smart-Contract-Colo...	T3
EP23 Immutable C2: How EtherHiding and Frontend Attacks are ...	https://www.youtube.com/watch?v=2Hc8bYX53DI	T3
Demystifying Exploitable Bugs in Smart Contracts · GitHub	https://github.com/ZhangZhuoSJTU/Web3Bugs	T3

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