INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-04-22 13:41 UTC

CVE-2026-5752: Cohere Terrarium Sandbox Escape via Pyodide Prototype Chain, No Patch Available

CVE VULNERABILITY | HIGH | CVSS 7.5

SCC Item ID	SCC-CVE-2026-0069
Type	CVE Vulnerability
CVE ID	CVE-2026-5752
Severity	HIGH
CVSS Base Score	7.5
EPSS Score	0.0002 (6th percentile)
Affected Products	Cohere AI Terrarium (Python sandbox), Pyodide (WebAssembly Python runtime), Node.js host process, Docker containers
Published	2026-04-22T03:16:00
Discovery Source	Rss

Executive Summary

A high-severity sandbox escape vulnerability (CVE-2026-5752) in Cohere AI's Terrarium tool allows an attacker to break out of an isolated Python execution environment and run arbitrary commands as root on the underlying host system. Terrarium is no longer maintained by Cohere, meaning no patch is forthcoming. Organizations using it to execute LLM-generated or untrusted Python code must mitigate through architectural controls or decommission the tool. Any AI/ML pipeline relying on Terrarium for code execution is exposed until mitigations are in place.

Technical Analysis

CVE-2026-5752 affects Cohere AI Terrarium, a Docker-deployed sandbox using Pyodide (WebAssembly Python runtime) to execute untrusted or LLM-generated Python code. The attack vector is JavaScript prototype chain traversal (prototype pollution) via Pyodide's Python-to-JavaScript bridge. An attacker-controlled Python payload can manipulate the JS prototype chain within the Pyodide runtime, escaping the WebAssembly sandbox boundary and achieving arbitrary command execution on the host Node.js process with root privileges. Relevant CWEs: CWE-1321 (prototype pollution), CWE-269 (improper privilege management), CWE-693 (protection mechanism failure). MITRE ATT&CK mappings include T1611 (escape to host), T1068 (exploitation for privilege escalation), T1059.006 (Python execution), T1210 (exploitation of remote services), T1083 (file and

directory discovery). CVSS base score is 7.5 (NVD). EPSS score is 0.00024 (6.5th percentile), indicating low current exploitation probability. No patch exists; Terrarium is unmaintained. Primary authoritative source: NVD (https://nvd.nist.gov/vuln/detail/CVE-2026-5752).

Action Checklist

1. Step 1, Identify and isolate: Immediately identify all environments running Cohere AI Terrarium. Isolate Terrarium Docker containers from the host network and from internal networks. If Terrarium accepts untrusted input or is internet-exposed, block inbound code submissions at the network perimeter until architectural mitigations are applied.
2. Step 2, Detect anomalies: Audit Docker container logs and Node.js process logs for anomalous child process spawning, unexpected file system access outside sandbox directories, or privilege escalation indicators. Look for Python payloads containing prototype manipulation patterns (e.g., '__proto__', 'constructor.prototype', 'Object.setPrototypeOf'). Check host process audit logs (auditd) for unexpected root-level command execution from the Node.js process.
3. Step 3, Remediate: No vendor patch is available. Remediation options: (a) Decommission Terrarium entirely and migrate LLM code execution to an actively maintained sandbox alternative; (b) If continued use is required, add a secondary isolation layer, run Terrarium inside a dedicated VM with no host network access, or use gVisor/Kata Containers to harden the container runtime boundary; (c) Apply strict input filtering to block prototype pollution patterns before code reaches the Pyodide runtime.
4. Step 4, Verify recovery: After isolation or decommissioning, verify no unauthorized processes persist on affected hosts. Audit root-level process history on all systems that ran Terrarium. Confirm container network segmentation is enforced. If compromise is suspected, isolate the host, preserve logs, and reimage from clean media.
5. Step 5, Systemic review: Review all AI/ML pipeline components that execute LLM-generated or untrusted code; apply the same isolation standard to each. Document Terrarium's end-of-maintenance status as a control gap in your third-party component inventory. Add unmaintained AI tooling to your software lifecycle review process. Map this incident to NIST CSF Protect function PR.IP-1 (baseline configuration management) and Govern function GV.SC (supply chain risk) for remediation tracking.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to CISO and legal/compliance immediately if auditd or process tree analysis reveals any root-level command execution originating from the Node.js Terrarium process, any new accounts created on the host, or any evidence of lateral movement from a host that ran Terrarium — confirmed compromise of a root-accessible host running an AI/ML pipeline may trigger breach notification obligations if the pipeline processed PII, PHI, or regulated data.

Recovery Notes	After containment or decommission, monitor all hosts that ran Terrarium for a minimum of 30 days for anomalous root-owned process execution, unexpected outbound connections from Node.js processes, and new cron or systemd timer entries — these are the persistence mechanisms most accessible to an attacker who achieved root via CVE-2026-5752. Before returning any replacement sandbox to production, verify the replacement runtime (gVisor, Kata Containers, or an alternative maintained sandbox) is enrolled in your vulnerability scanning cadence and explicitly tracked in the third-party component inventory. If host reimaging was required, validate post-reimage integrity by comparing system binary hashes against a trusted baseline before reconnecting the host to internal networks.
Forensic Artifacts	Docker container filesystem diff ('docker diff '): captures all files written, modified, or deleted inside the Terrarium container during its lifetime — the CVE-2026-5752 sandbox escape would produce writes outside the expected Pyodide scratch directory ('/tmp/pyodide/' or equivalent), which appear as anomalous 'A' (added) or 'C' (changed) entries in paths like '/etc/', '/root/', or '/usr/bin/' within the container layer. | Host auditd execve records filtered on Node.js parent PID: the prototype chain escape in Pyodide triggers arbitrary command execution on the host as root via the Node.js worker process — auditd records with 'syscall=execve', 'uid=0', and 'ppid=' that reference non-Node.js binaries (e.g., '/bin/bash', '/usr/bin/curl', '/bin/sh') are the primary host-side indicator of a successful escape. | Node.js application logs and container stdout for Pyodide prototype pollution payload patterns: successful exploitation of CVE-2026-5752 requires input containing JavaScript prototype chain manipulation ('__proto__', 'constructor.prototype', 'Object.setPrototypeOf') passed through Terrarium's code submission API — these strings will appear in the Node.js request handling logs or container stdout immediately before any anomalous child process activity. | '/proc//maps' and '/proc//fd' snapshot from the live Node.js process: Pyodide runs a WebAssembly binary in the Node.js process heap — a sandbox escape may result in unexpected executable memory regions or open file descriptors pointing to host filesystem paths outside the container mount namespace, visible in the process maps before the process is terminated. | Container network connection state captured via 'ss -tulnp' and 'nsenter -t -n -- ss -tulnp' before isolation: if CVE-2026-5752 was exploited for command-and-control establishment, the exploit payload executing as root on the host may have opened a reverse shell or beacon connection that is visible as an unexpected ESTABLISHED TCP connection from the Node.js process to an external IP, distinct from Terrarium's expected API listener port.

Per-Action IR Details

Step 1: Containment — Immediately identify all environments running Cohere AI Terrarium. Isolate Terrarium Docker containers from the host network and from internal networks. If Terrarium is internet-facing or accepts input from external systems (including LLM-generated code), block inbound code submission paths at the network perimeter until architectural mitigations are applied.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST SC-7 (Boundary Protection), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 12.2 (Establish and Maintain a Secure Network Architecture)

Compensating: Run 'docker ps --filter ancestor=cohere-terrarium --format "{{.ID}} {{.Names}} {{.Ports}}"' to enumerate all running Terrarium containers. Immediately apply 'docker network disconnect ' to detach each from host and internal bridge networks. At the host firewall (iptables or nftables), insert a DROP rule on the port Terrarium's Node.js HTTP listener binds to (commonly 8080 or 3000): 'iptables -I INPUT 1 -p tcp --dport 3000 -j DROP'. For perimeter blocking, add an ACL on the upstream router or cloud security group to deny inbound traffic to the Terrarium API endpoint IP/port immediately.

Evidence: Before isolating, capture a full snapshot of active container state: 'docker inspect > terrarium_inspect_$(date +%s).json' and 'docker diff > terrarium_fs_diff_$(date +%s).txt' to record any filesystem mutations the exploit may have already caused inside the container. Also run 'docker top ' and 'ps auxf' on the host to capture any anomalous child processes already spawned from the Node.js Terrarium process prior to network isolation. Preserve these outputs as timestamped forensic artifacts before any containment action modifies container state.

Step 2: Detection — Audit Docker container logs and Node.js process logs for anomalous child process spawning, unexpected file system access outside sandbox directories, or privilege escalation indicators. Look for Python payloads containing prototype manipulation patterns (e.g., references to '__proto__', 'constructor.prototype', or 'Object.setPrototypeOf'). Check host process audit logs (auditd or equivalent) for unexpected root-level command execution originating from the Node.js process.

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-3 (Content of Audit Records), CIS 8.2 (Collect Audit Logs), CIS 8.11 (Collect Service Provider Logs)

Compensating: Enable auditd on the Node.js host with a rule targeting process execution from the Terrarium service user: 'auditctl -a always,exit -F arch=b64 -S execve -F uid=0 -F ppid=$(pgrep -f terrarium) -k terrarium_escape'. Pull container stdout/stderr logs via 'docker logs --timestamps 2>&1 | grep -Ei "proto__|constructor\.prototype|setPrototypeOf|child_process|exec|spawn|eval"' to surface Pyodide prototype chain manipulation attempts. On the host, search auditd logs with 'ausearch -k terrarium_escape --interpret | grep -v "node\|npm"' to find root-level commands not originating from expected Node.js internals. Use osquery with the query 'SELECT pid, ppid, name, cmdline FROM processes WHERE parent IN (SELECT pid FROM processes WHERE name="node")' to map the full Node.js process tree and identify unauthorized child processes spawned by the Terrarium worker.

Evidence: Collect Docker container logs in full before any restart or cleanup: 'docker logs --timestamps > terrarium_container_logs_$(date +%s).txt'. Extract Node.js application logs from the container at '/app/logs/' or equivalent path via 'docker cp :/app/logs ./terrarium_app_logs/'. On the host, archive current auditd logs: 'cp /var/log/audit/audit.log ./audit_terrarium_$(date +%s).log'. Specifically search for Pyodide WebAssembly runtime activity — look for '.wasm' file access events and any writes outside the expected Terrarium sandbox scratch directory (typically '/tmp/terrarium/' or a volume mount). Capture '/proc//fd' and '/proc//maps' on the live Node.js process to identify any unexpected file descriptors or memory-mapped regions indicative of a sandbox escape payload still resident in memory.

Step 3: Eradication — No vendor patch is available. Eradication options: (a) Decommission Terrarium entirely and migrate LLM code execution to an actively maintained, isolated sandbox alternative; (b) If continued use is required, add a secondary isolation layer — run Terrarium inside a dedicated VM with no host network access, or use gVisor/Kata Containers to harden the container runtime boundary; (c) Apply strict input filtering to block prototype pollution patterns before code reaches the Pyodide runtime.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), NIST CM-7 (Least Functionality), NIST SA-22 (Unsupported System Components), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: If decommissioning: run 'docker stop && docker rm && docker rmi cohere/terrarium' and remove any bind-mounted volumes. Document decommission timestamp and retain the final container image as evidence. If continued use is required pending migration: deploy gVisor by installing 'runsc' and reconfiguring the Docker daemon to use 'runsc' as the runtime ('docker run --runtime=runsc ...'), which interposes a user-space kernel between the Pyodide WebAssembly runtime and the host kernel, breaking the prototype chain escape path at the syscall boundary. For input filtering as a compensating control, deploy a pre-processing Node.js middleware that applies a regex blocklist for '__proto__', 'constructor.prototype', and 'Object.setPrototypeOf' before any string reaches the Pyodide 'runPython()'

call — this is a defense-in-depth measure only and should not be treated as a substitute for architectural isolation given that no patch is available.

Evidence: Before stopping containers, use 'docker export | gzip > terrarium_final_image_$(date +%s).tar.gz' to preserve the container filesystem for post-mortem analysis. Capture the full list of files modified inside the container during its lifetime via 'docker diff ' and note any files written outside expected Pyodide scratch paths. If compromise is confirmed or suspected, preserve a memory dump of the Node.js process before termination using 'gcore ' — this may capture in-memory WebAssembly heap state from Pyodide and any shellcode or post-exploitation payloads injected during the sandbox escape.

Step 4: Recovery — After isolation or decommissioning, verify no unauthorized processes persist on affected hosts. Audit root-level process history on all systems that ran Terrarium. Confirm container network segmentation is enforced. If compromise is suspected, treat the host Node.js environment as potentially compromised and follow full incident response procedures including host reimaging.

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), CIS 4.2 (Establish and Maintain a Secure Configuration Process for Network Infrastructure), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory)

Compensating: Run 'ps auxf | grep -v "\[" | awk "{print $1,$2,$3,$11}" | sort -k1' on all hosts that ran Terrarium to identify any root-owned processes without a clear legitimate parent. Cross-reference against a known-good process baseline if one exists. Use 'last -F' and 'lastlog' to check for any new interactive logins to the host that coincide with the window Terrarium was running. Verify Docker network segmentation post-recovery by running 'docker network inspect bridge' and confirming Terrarium containers (if any remain for analysis) have no routes to internal subnets. For host integrity verification before returning to production, use AIDE or Tripwire if deployed — if not, generate a fresh file hash baseline of critical Node.js binary paths and system binaries ('sha256sum /usr/bin/node /usr/local/bin/npm /bin/bash /usr/bin/python3 > post_recovery_baseline.txt') and compare against a trusted reference.

Evidence: Before reimaging any potentially compromised host, collect: full auditd log archive ('tar czf auditd_final_$(hostname)_$(date +%s).tar.gz /var/log/audit/'), Node.js npm package list ('npm list -g --depth=0 > npm_global_packages.txt' and per-project 'npm list --depth=0'), and a snapshot of all cron jobs and systemd timers ('crontab -l -u root > root_crontab.txt; systemctl list-timers --all > systemd_timers.txt') to detect any persistence mechanisms planted via the Terrarium escape. These artifacts are critical because the CVE-2026-5752 escape path grants root on the host — an attacker with root can install cron-based backdoors, modify Node.js binaries, or plant SUID binaries that survive container removal.

Step 5: Post-Incident — Review all AI/ML pipeline components that execute LLM-generated or untrusted code; apply the same isolation-depth standard to each. Document Terrarium's end-of-maintenance status as a control gap in your third-party component inventory. Add unmaintained AI tooling to your software lifecycle review process. Map this incident to NIST CSF Protect function PR.IP-1 (baseline configuration management) and Govern function GV.SC (supply chain risk) for remediation tracking.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity (Lessons Learned)

Controls: NIST IR-5 (Incident Monitoring), NIST IR-8 (Incident Response Plan), NIST SA-22 (Unsupported System Components), NIST RA-3 (Risk Assessment), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Build a dedicated inventory entry for Terrarium in your CMDB or a simple CSV asset register, flagging it with 'maintenance_status: EOL', 'last_vendor_commit: [date]', 'risk_acceptance_required: true', and 'sandbox_escape_cve: CVE-2026-5752'. Extend this review to all other AI/ML sandbox components (e.g., any other Pyodide-based runners, Jupyter kernel execution environments, LangChain tool execution wrappers) using a one-time 'pip show' and 'npm list' audit against a list of known unmaintained AI execution packages. Schedule a quarterly review of AI/ML pipeline components against the CISA Known Exploited Vulnerabilities catalog and GitHub's advisory

database using 'gh api /repos///vulnerability-alerts' or OSV Scanner ('osv-scanner --lockfile package-lock.json') to catch future EOL or newly disclosed issues before they reach production.

Evidence: Retain the complete incident record including: the original Terrarium container image hash ('docker inspect --format="{{.Id}}" '), all log archives collected during detection and recovery phases, the auditd rule configuration applied during investigation, and a timestamped record of when Cohere AI's last Terrarium commit occurred (verifiable via the public GitHub repository commit history). This evidence package supports post-incident review per NIST 800-61r3 §4, satisfies NIST IR-5 (Incident Monitoring) documentation requirements, and provides the factual record needed for a GRC control gap report linking CVE-2026-5752 to the absence of a third-party component lifecycle management process.

Detection Guidance

Primary detection focus is anomalous behavior in the Node.js host process and Docker container environment. Key indicators: (1) Unexpected child process creation from the Node.js PID, monitor with auditd rule '-a always,exit -F arch=b64 -S execve' and filter for processes whose parent is the Terrarium Node.js process. (2) File system access outside expected sandbox directories by the Node.js process, particularly writes to /etc, /bin, /usr, or /root. (3) Python code submissions containing prototype pollution strings: '__proto__', 'constructor.prototype', 'Object.defineProperty', or 'Object.setPrototypeOf'. (4) Outbound network connections from the Terrarium container to unexpected destinations, baseline normal egress and alert on deviations. (5) Docker escape indicators: container processes accessing host PID namespace or host filesystem mounts. SIEM query guidance: correlate Node.js process audit events with container runtime logs (docker logs, containerd logs) for the same time window. As of CVE publication, no public proof-of-concept code or CVE-specific SIEM signatures have been disclosed. Behavioral detection is the primary available method.

Framework Mappings

MITRE-ATTACK

- **T1611** — Escape to Host
- **T1210** — Exploitation of Remote Services
- **T1083** — File and Directory Discovery
- **T1059.006** — Python
- **T1068** — Exploitation for Privilege Escalation

NIST-800-53R5

- **AC-6** — Least Privilege
- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation
- **AT-2** — Literacy Training and Awareness

OWASP-TOP10-2021

- **A01:2021** — Broken Access Control

CIS-V8

- **5.4** — Restrict Administrator Privileges to Dedicated Administrator Accounts

- **6.8** — Define and Maintain Role-Based Access Control
- **7.3** — Perform Automated Operating System Patch Management
- **7.4** — Perform Automated Application Patch Management
- **14.2** — Train Workforce Members to Recognize Social Engineering Attacks

ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1611	Escape to Host	Privilege-Escalation
T1210	Exploitation of Remote Services	Lateral-Movement
T1083	File and Directory Discovery	Discovery
T1059.006	Python	Execution
T1068	Exploitation for Privilege Escalation	Privilege-Escalation

Sources

Source	URL	Tier
Security News	https://thehackernews.com/2026/04/cohere-ai-terrarium-sandbox-flaw....	T3
CVE-2026-5752 Detail - NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-5752	T1
CVE-2026-5752 | Mondoo Vulnerability Intelligence	https://mondoo.com/vulnerability-intelligence/vulnerability/CVE-202...	T3
CVE-2026-5752: Terrarium Sandbox Escape RCE Vulnerability	https://www.sentinelone.com/vulnerability-database/cve-2026-5752/	T3
CVE-2026-5752 Security Vulnerability Analysis & Exploit Details	https://cve.akaoma.com/cve-2026-5752	T3

Solutions is not liable for any damages resulting from the use of this information.

Generated 2026-04-22 13:41 UTC by TJS Security Command Center