INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-29 15:08 UTC

CVE-2026-55200: Public PoC Released for Critical libssh2 Pre-Auth Heap Overflow Affecting Broad SSH Client Ecosystem

CVE VULNERABILITY | **CRITICAL** | CVSS 9.5

SCC Item ID	SCC-CVE-2026-0371
Type	CVE Vulnerability
CVE ID	CVE-2026-55200, CVE-2026-55199, CVE-2025-15661, CVE-2019-3855
Severity	CRITICAL
CVSS Base Score	9.5
EPSS Score	0.0092 (56th percentile)
Affected Products	libssh2 all versions through 1.11.1; downstream consumers including curl, Git, PHP, backup agents, firmware updaters, and embedded appliances, particularly those statically linking libssh2
Published	2026-06-29T03:06:34
Discovery Source	Rss

Executive Summary

A critical vulnerability in libssh2, a widely embedded SSH client library, allows a malicious SSH server to trigger a heap buffer overflow before any user authentication occurs. Because libssh2 is statically linked into curl, Git, PHP, backup agents, firmware updaters, and embedded appliances, a standard OS-level patch will not protect downstream products; each consumer must be individually identified and updated. A public proof-of-concept has been released, according to The Hacker News and the oss-sec mailing list, lowering the barrier for exploit development and raising urgency for organizations to audit their full software supply chain.

Technical Analysis

CVE-2026-55200 is an integer overflow (CWE-190) leading to a heap buffer write (CWE-122, CWE-680) in libssh2 through version 1.11.1. The flaw is exploitable by a malicious or adversary-controlled SSH server prior to client authentication, requiring no user interaction. CVSS base score is reported at 9.2-9.5; EPSS score is 0.00922 (55.8th percentile) as of the data provided. A public PoC has been released; according to reporting, reliable remote code execution against live targets requires additional work beyond the PoC. No fixed libssh2 release was tagged at the time of reporting. CVE-2026-55199 is a companion vulnerability disclosed in the

same cluster; CVE-2025-15661 and CVE-2019-3855 are referenced as contextual or related findings in the oss-sec thread. The attack surface is broad due to static linking: downstream consumers (curl, Git, PHP, backup agents, firmware updaters, appliances) each carry their own copy of the vulnerable library and will not be remediated by patching the OS-level libssh2 package alone. MITRE ATT&CK techniques associated with this class of exploitation include T1195.002 (Compromise Software Supply Chain), T1203 (Exploitation for Client Execution), T1190 (Exploit Public-Facing Application), T1554 (Compromise Client Software Binary), and T1071.002 (Application Layer Protocol: File Transfer Protocols). Confidence in the core technical claims is MEDIUM: primary reporting is news aggregation; NVD confirmation for CVE-2026-55200 specifically was not present in the supplied data, though the companion CVE-2026-55199 has a cited NVD detail page providing partial authoritative corroboration. Red Hat Customer Portal is cited as a T1 source for CVE-2026-55200. All source URLs require human validation before treating as confirmed.

Action Checklist

1. Step 1: Containment. Identify all systems that initiate outbound SSH connections using libssh2, including applications that statically link it (curl, Git clients, PHP SSH extensions, backup agents, firmware update utilities, embedded appliances). Restrict outbound SSH to known, trusted server IPs via firewall ACLs to reduce exposure to malicious server-side exploitation while remediation is underway. Prioritize internet-facing or externally reachable SSH client workflows. Reference: NIST AC-4 (Information Flow Enforcement).
2. Step 2: Detection. Inventory all software packages and appliances for embedded or statically linked libssh2 through version 1.11.1. Standard package manager queries (e.g., rpm -qa libssh2, dpkg -l libssh2) will not surface statically linked copies; use binary scanning tools (e.g., grep for libssh2 version strings in binaries, or use a software composition analysis tool) across application directories, container images, and firmware. Review CIS 1.1 (Enterprise Asset Inventory) and CIS 2.1 (Software Inventory) to confirm scanning coverage. Check Red Hat Customer Portal (cited T1 source) and vendor advisories for affected product lists specific to your environment.
3. Step 3: Eradication. Apply vendor-issued patches for each downstream consumer independently as they become available. Monitor the libssh2 project for a fixed release beyond 1.11.1, as well as downstream advisories from curl, Git, PHP, and appliance vendors. Do not rely on OS-level libssh2 package updates to remediate statically linked consumers. Reference: CIS 7.3 (Automated OS Patch Management) and CIS 7.4 (Automated Application Patch Management) for patch cadence requirements. Reference: NIST SI-2 (Flaw Remediation).
4. Step 4: Recovery. After patching, re-scan affected binaries and container images to confirm the vulnerable libssh2 version string is no longer present. Re-enable outbound SSH flows that were restricted during containment, validated against the updated software inventory. Monitor SSH client connection logs for anomalous server certificate changes or unexpected connection targets that could indicate ongoing adversary-controlled server exposure. Reference: NIST AU-6 (Audit Record Review, Analysis, and Reporting); CIS 8.2 (Collect Audit Logs).
5. Step 5: Post-Incident. Conduct a supply chain dependency review to establish a repeatable process for identifying statically linked third-party libraries across the software portfolio before future disclosures. Establish a software composition analysis (SCA) capability if not already present. Map gaps against CIS 2.1 (Software Inventory) and NIST AC-4 (Information Flow Enforcement). Review SSH client trust policies to ensure clients only connect to known, validated servers, reducing exposure to malicious server-initiated exploitation. Reference: D3FEND D3-CH (Credential Hardening) and D3-CRO (Credential Rotation) for

any SSH keys that may have been used during the exposure window.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to senior IR leadership and legal/compliance if any host from the Step 1 volatile capture shows outbound SSH connections to unrecognized external IPs during the PoC-active exposure window, if crash dumps or memory corruption artifacts consistent with heap overflow exploitation are found in curl/git/backup-agent processes, or if regulated data (PII, PHI, PCI) was accessible from any host running a vulnerable statically linked libssh2 consumer — as this may trigger breach notification obligations.
Recovery Notes	After patching all identified consumers, re-validate using the strings-based binary scan to confirm no version ≤ 1.11.1 remains in any binary, container layer, or firmware image — OS-level package updates alone are insufficient and will produce false confidence. Maintain elevated outbound SSH monitoring for a minimum of 30 days post-remediation, specifically watching for reconnection attempts to server IPs captured during the containment phase, which could indicate an adversary retaining access via a previously compromised SSH server. Rotate all SSH private keys and known_hosts entries on affected systems, as the pre-auth nature of CVE-2026-55200 means exploitation could have occurred without leaving authentication log evidence.
Forensic Artifacts	Core dumps and SIGABRT/SIGSEGV crash records from curl, git, PHP CLI, or backup agent processes — located in `/var/crash/`, `/tmp/core.*`, or the process working directory — indicating a heap overflow triggered during libssh2 pre-auth key exchange packet parsing (SSH_MSG_KEXINIT or SSH_MSG_NEWKEYS phase) against CVE-2026-55200. | Outbound SSH connection logs captured via `ss -tnp dport = :22` or `netstat -ano` showing remote server IPs that do not appear in the organization's approved SSH destination inventory — the pre-auth attack requires client-to-adversary-server connectivity, making novel destination IPs the primary network indicator. | libssh2 version strings extracted via `strings` from statically linked binaries (curl, git, PHP extensions, backup agent executables) confirming the presence of a vulnerable version (≤ 1.11.1) — these serve as the definitive exposure scope record and must be preserved pre-patch. | SSH known_hosts files (`~/.ssh/known_hosts`, `/etc/ssh/ssh_known_hosts`) showing entries for external servers contacted during the exposure window — a changed or newly added host key for a previously trusted server IP may indicate server substitution by an adversary leveraging this server-side exploitation vector. | Memory image heap analysis (via Volatility3 `linux.heap` or `windows.heap` plugin) from processes holding active SSH connections at time of containment — a successful CVE-2026-55200 exploit would produce anomalous heap chunk metadata corruption in the libssh2 memory region of the affected process, distinguishable from normal SSH session state.

Per-Action IR Details

Step 1: Containment — Identify all systems that initiate outbound SSH connections using libssh2, including applications that statically link it (curl, Git clients, PHP SSH extensions, backup agents, firmware update utilities, embedded appliances). Restrict outbound SSH to known, trusted server IPs via firewall ACLs to reduce exposure to malicious server-side exploitation while remediation is underway. Prioritize internet-facing or externally reachable SSH client workflows. Reference: NIST AC-4 (Information Flow Enforcement).

NIST Phase: Containment

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

Controls: NIST AC-4 (Information Flow Enforcement), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 4.5 (Implement and Manage a Firewall on End-User Devices)

Compensating: On Linux hosts, use `iptables -A OUTPUT -p tcp --dport 22 -d -j ACCEPT && iptables -A OUTPUT -p tcp --dport 22 -j DROP` to allowlist known SSH server IPs. On Windows, use Windows Firewall with Advanced Security (netsh advfirewall) to block outbound TCP/22 except to approved destinations. Generate a prior netstat baseline: `netstat -ano | findstr ':22'` (Windows) or `ss -tnp dport = :22` (Linux) to document active SSH client flows before applying restrictions.

Evidence: Before applying ACL changes, capture active outbound SSH connection state: run `ss -tnp dport = :22` or `netstat -ano | findstr ':22'` to record all live SSH client sessions and the PIDs/process names holding them. Log the remote server IPs — any unrecognized destination is a candidate for adversary-controlled server involvement in a pre-auth heap overflow attempt against CVE-2026-55200. On Linux, also run `lsof -i :22 -n -P` to map file descriptors to owning binaries (e.g., curl, git, backup agent) before flows are disrupted.

Step 2: Detection — Inventory all software packages and appliances for embedded or statically linked libssh2 through version 1.11.1. Standard package manager queries (e.g., rpm -qa libssh2, dpkg -l libssh2) will not surface statically linked copies; use binary scanning tools (e.g., grep for libssh2 version strings in binaries, or use a software composition analysis tool) across application directories, container images, and firmware. Review CIS 1.1 (Enterprise Asset Inventory) and CIS 2.1 (Software Inventory) to confirm scanning coverage. Check Red Hat Customer Portal (cited T1 source) and vendor advisories for affected product lists specific to your environment.

NIST Phase: Detection Analysis

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

Controls: CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.2 (Ensure Authorized Software is Currently Supported), NIST AU-2 (Event Logging)

Compensating: Run `find / -type f \( -name '*.so*' -o -perm /111 \) -exec strings {} \; 2>/dev/null | grep -i 'libssh2' | grep -E '1\.(([0-9]\.)|(1[01]\.))'` to surface version strings from both shared objects and statically linked binaries. For container images, use `docker run --rm -v /:/host alpine find /host -type f -exec strings {} \; 2>/dev/null | grep 'libssh2_version'`. For firmware blobs, extract with `binwalk -e ` and repeat the strings grep. A 2-person team can automate this across a host list using parallel-ssh or Ansible's raw module.

Evidence: This step does not alter live state; no volatile pre-capture is required. However, document the exact binary paths, version strings found, and owning process names before any patching occurs — this establishes the pre-remediation exposure baseline. Preserve package manager output (`rpm -qa libssh2; dpkg -l '*libssh2*'`) and strings-scan results as forensic records of scope. If any binary shows a libssh2 version string ≤ 1.11.1 AND that binary has active outbound SSH connections per Step 1 capture, escalate that host immediately as potentially exploited.

Step 3: Eradication — Apply vendor-issued patches for each downstream consumer independently as they become available. Monitor the libssh2 project for a fixed release beyond 1.11.1, as well as downstream advisories from curl, Git, PHP, and appliance vendors. Do not rely on OS-level libssh2 package updates to remediate statically linked consumers. Reference: CIS 7.3 (Automated OS Patch Management) and CIS 7.4 (Automated Application Patch Management) for patch cadence requirements. Reference: NIST SI-2 (Flaw Remediation) — note this control is not in the provided knowledge base; cite only if your organization's policy references it independently.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: Track patch availability per consumer using a shared spreadsheet mapping binary path → owning product → vendor advisory URL → patch status. For curl specifically, monitor https://curl.se/docs/security.html; for Git, monitor https://git-scm.com/downloads/; for PHP, monitor https://www.php.net/ChangeLog-8.php. Until vendor patches are available, maintain the ACL restrictions from Step 1 as a compensating control. For embedded appliances with no patch yet available, physically isolate or power down units that initiate outbound SSH to untrusted servers.

Evidence: Before applying any patch to a host suspected of prior exploitation (identified via Step 1 and Step 2 findings), acquire a full memory image using `avml` (Linux) or WinPMEM (Windows) and capture running process list (`ps auxf` / `Get-Process`), open network connections (`ss -tnp` / `Get-NetTCPConnection`), and heap-related crash artifacts (`/var/crash/`, `/var/log/apport.log`, Windows Event Log Application errors from the vulnerable binary). A heap overflow in libssh2's pre-auth key exchange (specifically in packet parsing prior to NEWKEYS) may produce SIGABRT/SIGSEGV crash dumps or silent memory corruption; check for core dumps in the working directory of curl/git/backup-agent processes.

Step 4: Recovery — After patching, re-scan affected binaries and container images to confirm the vulnerable libssh2 version string is no longer present. Re-enable outbound SSH flows that were restricted during containment, validated against the updated software inventory. Monitor SSH client connection logs for anomalous server certificate changes or unexpected connection targets that could indicate ongoing adversary-controlled server exposure. Reference: NIST AU-6 (Audit Record Review, Analysis, and Reporting); CIS 8.2 (Collect Audit Logs).

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AU-6 (Audit Record Review, Analysis, And Reporting), CIS 8.2 (Collect Audit Logs), NIST AU-3 (Content Of Audit Records), NIST AU-11 (Audit Record Retention)

Compensating: Post-patch, re-run the strings-based version scan from Step 2 against all previously affected binaries and confirm no `libssh2` version string ≤ 1.11.1 remains. To monitor for adversary-controlled SSH server re-engagement after ACL restoration, deploy a Sigma rule on sshd/client logs detecting new outbound connections to IPs outside the approved allowlist: filter on `sshd` or `ssh` process names in `/var/log/auth.log` or Windows Event ID 4688 (process creation) with command lines invoking ssh/curl/git to non-baseline destinations. Capture and compare SSH server host keys on each reconnection using `ssh-keyscan` to detect server substitution.

Evidence: This step re-enables network flows altered during containment; before lifting ACL restrictions, confirm the patched binary hashes match vendor-published values (`sha256sum `). Log the exact timestamp of ACL restoration and begin a 30-day elevated monitoring window on outbound SSH from all previously affected hosts. Watch for `/var/log/auth.log` or Windows Security Event ID 4625 patterns suggesting reattempted exploitation from the same adversary-controlled server IPs documented in Step 1 volatile capture.

Step 5: Post-Incident — Conduct a supply chain dependency review to establish a repeatable process for identifying statically linked third-party libraries across the software portfolio before future disclosures. Establish a software composition analysis (SCA) capability if not already present. Map gaps against CIS 2.1 (Software Inventory) and NIST AC-4 (Information Flow Enforcement). Review SSH client trust policies to ensure clients only connect to known, validated servers, reducing exposure to malicious server-initiated exploitation (D3-CH: Credential Hardening; D3-CRO: Credential Rotation for any SSH keys that may have been used during the exposure window).

NIST Phase: Post Incident

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

Controls: CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.2 (Ensure Authorized Software is Currently Supported), NIST AC-4 (Information Flow Enforcement), NIST AU-6 (Audit Record Review, Analysis, And Reporting)

Compensating: Integrate a free SCA tool such as Syft (`syft -o cyclonedx-json`) to generate a Software Bill of Materials (SBOM) for all container images and application directories; pipe output to Grype (`grype sbom:`) to flag future statically linked vulnerable libraries. Schedule this scan as a weekly cron job across build pipelines. For SSH key rotation, enumerate all SSH private keys that were resident on affected hosts during the exposure window using `find /

-name 'id_rsa' -o -name 'id_ed25519' 2>/dev/null` and revoke/replace any exposed to a host that had active connections to untrusted SSH servers.

Evidence: No volatile capture is required for this phase; the host is no longer under active compromise. Preserve the complete incident record: pre-patch binary scan results (Step 2), volatile network state captures (Step 1), memory images and crash dumps (Step 3), and patched binary hashes (Step 4) for at least the organization's defined audit retention period per NIST AU-11 (Audit Record Retention). The exposure window for CVE-2026-55200 begins at the date of public PoC release; document all SSH client connections made to external servers during that window as potential evidence of exploitation attempts against this pre-auth heap overflow.

Detection Guidance

Primary detection focus is inventory, not network signatures, because exploitation is server-initiated pre-authentication and may not produce distinctive client-side log entries. Actions: (1) Run software composition analysis or binary string scanning across all application directories, container images, CI/CD build artifacts, and firmware packages to identify libssh2 version strings of 1.11.1 or earlier; package manager queries alone are insufficient for statically linked copies. (2) Query asset inventory and CMDB for applications known to use curl, Git, PHP SSH extensions, backup agents, or firmware update utilities, and treat each as potentially affected until individually verified. (3) Review outbound SSH connection logs (typically syslog or endpoint EDR telemetry) for connections to external or unexpected SSH servers; adversary-controlled servers are the delivery mechanism. Alert on SSH client connections to newly observed external IPs or hostnames. (4) Monitor vendor advisory feeds for Red Hat, curl, Git, PHP, and appliance manufacturers for updated affected-version lists and patch availability. (5) No IOCs (IPs, domains, hashes) were present in the supplied source data; detection at this stage is posture-based, not indicator-based. Reference: NIST AU-2 (Event Logging), NIST AU-6 (Audit Record Review), CIS 8.2 (Collect Audit Logs). D3FEND countermeasures: D3-SFA (System File Analysis) for binary scanning; D3-LAM (Local Account Monitoring) to detect post-exploitation lateral movement if exploitation does occur.

Framework Mappings

MITRE-ATTACK

- **T1195.002** — Compromise Software Supply Chain
- **T1059** — Command and Scripting Interpreter
- **T1203** — Exploitation for Client Execution
- **T1190** — Exploit Public-Facing Application
- **T1554** — Compromise Host Software Binary
- **T1071.002** — File Transfer Protocols

NIST-800-53R5

- **CM-7** — Least Functionality
- **SA-9** — External System Services
- **SR-3** — Supply Chain Controls and Processes
- **SI-7** — Software, Firmware, and Information Integrity
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring

- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation
- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning
- **SI-16** — Memory Protection
- **AT-2** — Literacy Training and Awareness

CIS-V8

- **16.10** — Apply Secure Design Principles in Application Architectures
- **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
- **A.5.23** — Information security for use of cloud services

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1195.002	Compromise Software Supply Chain	Initial-Access
T1059	Command and Scripting Interpreter	Execution
T1203	Exploitation for Client Execution	Execution
T1190	Exploit Public-Facing Application	Initial-Access
T1554	Compromise Host Software Binary	Persistence
T1071.002	File Transfer Protocols	Command-And-Control

Sources

Source	URL	Tier
Security News	https://thehackernews.com/2026/06/public-poc-released-for-critical....	T2
oss-sec: Re: libssh2: CVE-2026-55200 (critical), CVE ... - Seclists.org	https://seclists.org/oss-sec/2026/q2/1017	T3
CVE-2026-55199 Detail - NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-55199	T1

Source	URL	Tier
libssh2 and alt-libssh2 CVEs (CVE-2026-55200, CVE-2026-55199 ...	https://cloudlinux.zendesk.com/hc/en-us/articles/28390225608092-lib...	T3
CVE-2026-55200 - Red Hat Customer Portal	https://access.redhat.com/security/cve/cve-2026-55200	T1

Generated 2026-06-29 15:08 UTC by TJS Security Command Center