INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-04-30 06:30 UTC

AI-Assisted Audit Exposes 38 Vulnerabilities in OpenEMR, Putting 100,000+ Healthcare Providers at Risk

SECURITY ANALYSIS | **HIGH** | CVSS 9.5

SCC Item ID	SCC-STY-2026-0096
Type	Security Analysis
Severity	HIGH
CVSS Base Score	9.5
Affected Products	OpenEMR (open-source electronic health record platform, all deployments, specific versions not confirmed from available source data)
Published	2026-04-29T15:32:42
Discovery Source	Rss

Executive Summary

Researchers from AISLE used AI-assisted analysis to identify 38 security vulnerabilities across OpenEMR, an open-source electronic health record platform deployed by more than 100,000 healthcare providers worldwide. The vulnerability set includes SQL injection, remote code execution, and deserialization flaws, individually serious, and in combination, they create direct ransomware and patient data exfiltration pathways. Patches have been issued, but the disclosure signals a broader truth: open-source healthcare infrastructure carries systemic security debt that AI-assisted auditing is now making visible at scale.

Technical Analysis

AISLE's research team applied AI-assisted static and dynamic analysis to OpenEMR, producing a 38-vulnerability disclosure that spans six CWE categories: SQL injection (CWE-89), OS command injection (CWE-78), code injection (CWE-94), path traversal (CWE-22), deserialization of untrusted data (CWE-502), and information disclosure (CWE-200). The breadth is notable; this is not a single bug class with a single fix surface. Each category represents a distinct attack entry point or escalation mechanism.

The attack chain potential here is significant. An unauthenticated or low-privileged adversary exploiting CWE-89 (SQL injection) can extract patient records or credential tables. CWE-78 and CWE-94, OS command injection and code injection, offer remote code execution if reachable from an exposed interface. CWE-502 deserialization flaws are historically reliable for RCE in Java and PHP applications and are a known ransomware

deployment vector. CWE-22 path traversal enables file read and, in some configurations, write operations outside the web root. Together, these create a viable kill chain: initial access via web-facing interface (MITRE T1190), command execution (T1059), web shell deployment (T1505.003), lateral data collection (T1005, T1213), and exfiltration or encryption (T1048, T1041, T1486).

The threat actor alignment in the item data - Rhysida, BlackCat/ALPHV, and LockBit - reflects the realistic threat population most likely to weaponize these vulnerability classes against healthcare targets. None are attributed to this specific OpenEMR disclosure; this represents threat modeling based on sector targeting history documented by CISA and HHS advisories.

The aggregate CVSS base score of 9.5 represents the highest individual vulnerability in the set, per source reporting, and is not independently verified against NVD or AISLE's full disclosure. Individual CVE identifiers for the 38 findings are referenced by source outlets but were not present in the raw data provided; verification requires checking NVD or the AISLE disclosure report directly.

The AI-methodology angle carries its own analytical weight. AISLE's use of AI-assisted analysis to surface 38 findings in a single audit cycle suggests that the cost curve for comprehensive security review of open-source healthcare software is shifting. This matters for the sector: OpenEMR is free to deploy, which drives adoption among smaller and under-resourced providers who often lack dedicated security teams. The same economics that make OpenEMR attractive also mean those deployments are least likely to be hardened, monitored, or patched quickly. Patches are confirmed issued; the exposure window is now a function of how rapidly that long tail of smaller providers applies them.

Action Checklist

1. Step 1: Assess exposure, determine if your organization deploys OpenEMR in any environment, including affiliates, subsidiaries, managed service partners, or third-party billing and clinical vendors who may run OpenEMR on your behalf.
2. Step 2: Apply available patches immediately. OpenEMR patches have been confirmed issued; verify your deployed version against the AISLE disclosure report and OpenEMR's official release notes, then apply updates across all instances including non-production environments.
3. Step 3: Review web-facing attack surface, confirm whether your OpenEMR instance is internet-accessible or reachable from untrusted network segments; if so, implement WAF rules covering SQL injection and command injection patterns as a compensating control while patching proceeds.
4. Step 4: Audit for indicators of prior compromise, review web server logs, application logs, and database query logs for anomalous patterns consistent with SQL injection probing (T1190), unexpected file writes or new files in web directories (T1505.003), and unusual outbound data transfers (T1048, T1041) covering the period prior to patch availability.
5. Step 5: Update threat model, add OpenEMR-class open-source EHR software to your third-party risk register; validate that vendor security assessment processes cover open-source dependencies used by clinical partners, not only commercial software.
6. Step 6: Communicate findings, brief clinical operations and compliance leadership on patient data exposure risk with specific reference to the vulnerability classes involved; frame remediation timeline against HIPAA breach risk thresholds.
7. Step 7: Monitor developments - track NVD for individual CVE assignments across the 38 findings, monitor CISA for any KEV additions, and check AISLE's technical disclosure (aisle.com) for IOC and

detailed remediation guidance.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to CISO, Privacy Officer, and legal counsel immediately if Step 4 log analysis reveals confirmed SQLi probe attempts, successful authentication following anomalous queries, new PHP files in the OpenEMR web root, or any outbound data transfer from the database server during the pre-patch exposure window, as any of these conditions constitutes a presumptive HIPAA breach requiring 60-day OCR notification assessment under 45 CFR §164.402.
Recovery Notes	Post-patching, restore OpenEMR to a known-good state by verifying file integrity against the official OpenEMR release checksums and confirming no unauthorized PHP files remain in the web root using `find /var/www/openemr/ -name '*.php' | xargs sha256sum` compared against a clean installation manifest. Maintain elevated logging (MySQL general query log, Apache/nginx combined log with POST body logging enabled) for a minimum of 90 days post-patch to detect any re-exploitation attempts or delayed-action web shells installed prior to patching. If the AISLE full technical disclosure reveals deserialization gadget chains specific to OpenEMR's PHP object handling, conduct a targeted memory analysis of the web server process using Volatility or `/proc//maps` inspection to rule out in-memory persistence that survived patching.
Forensic Artifacts	Apache/nginx access logs containing POST requests to OpenEMR endpoints such as `/interface/login/login.php`, `/portal/account/register.php`, and `/library/ajax/` — SQL injection exploitation against OpenEMR's authentication and patient data endpoints will appear as requests with encoded payloads (`%27`, `UNION+SELECT`, `--+`) in URI parameters or POST bodies; preserve these before log rotation overwrites them. | MySQL general query log (`/var/log/mysql/mysql.log`) and slow query log showing queries executed by the OpenEMR application database user — successful SQLi will produce syntactically anomalous or stacked queries against PHI tables (`patient_data`, `form_encounter`, `documents`, `pnotes`); queries with unusual `OUTFILE` or `LOAD_FILE` clauses indicate data exfiltration or file write attempts. | Filesystem timeline for the OpenEMR web root (`/var/www/openemr/`) generated with `find / -newer -type f -name '*.php'` — RCE exploitation via T1505.003 (Server Software Component: Web Shell) would result in new PHP files written to writable directories such as `sites/default/documents/`, `public_html/`, or `custom/`; file creation timestamps within the vulnerability exposure window are high-priority artifacts. | PHP session files in `/var/lib/php/sessions/` and OpenEMR's own session storage — if SQLi was used to extract credential hashes or session tokens, subsequent authenticated sessions from unexpected source IPs would appear in the PHP session store; correlate session IDs in the web server access logs against the session files to identify unauthorized authenticated access. | Network flow records or `tcpdump` captures showing outbound connections from the OpenEMR server to non-standard destinations — T1048 (Exfiltration Over Alternative Protocol) and T1041 (Exfiltration Over C2 Channel) following RCE would produce outbound traffic to attacker-controlled infrastructure; connections on non-standard ports or to newly registered domains from the OpenEMR server IP during the exposure window are primary exfiltration indicators.

Per-Action IR Details

Step 1: Assess exposure — determine if your organization deploys OpenEMR in any environment, including affiliates, subsidiaries, managed service partners, or third-party billing and clinical vendors who may run

OpenEMR on your behalf.

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: establishing asset visibility and scope before incident declaration

Controls: NIST IR-4 (Incident Handling) — preparation component requires knowing which systems are in scope, NIST SI-5 (Security Alerts, Advisories, and Directives) — requires tracking vendor advisories for deployed software including open-source, CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory) — OpenEMR instances must appear in the asset inventory, including third-party-hosted instances, CIS 7.1 (Establish and Maintain a Vulnerability Management Process) — scope assessment is the first gate in any vulnerability response workflow

Compensating: Run a network sweep using nmap targeting default OpenEMR ports (80, 443, 8080) combined with HTTP banner grabbing: `nmap -p 80,443,8080 --script http-title,http-server-header ` and grep output for 'OpenEMR'. Supplement with a query to your DNS zone files or firewall NAT rules for hostnames containing 'emr', 'openemr', or 'ehr'. For third-party vendors, issue a one-page questionnaire requiring attestation of OpenEMR use within 48 hours.

Evidence: Before conducting the sweep, snapshot your current firewall ACLs, DNS records, and any existing CMDB exports to establish a baseline scope record. If OpenEMR instances are found that were not in your asset inventory, document the discovery gap — this is a finding independent of the vulnerability itself and may constitute a HIPAA Security Rule gap (45 CFR §164.310(a)(2)(iv) asset inventory requirement).

Step 2: Apply available patches immediately — OpenEMR patches have been confirmed issued; verify your deployed version against the AISLE disclosure report and OpenEMR's official release notes, then apply updates across all instances including non-production environments.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication: eliminating the vulnerability from affected systems after confirming scope

Controls: NIST SI-2 (Flaw Remediation) — requires identifying, testing, and applying patches to correct system flaws, NIST CM-3 (Configuration Change Control) — patch application is a configuration change requiring documented approval and testing, even under emergency timelines, CIS 7.3 (Perform Automated Operating System Patch Management) — applies to OS-layer dependencies bundled with OpenEMR (PHP, Apache/nginx, MySQL/MariaDB), CIS 7.4 (Perform Automated Application Patch Management) — directly governs OpenEMR application-layer patch deployment

Compensating: Before patching, capture the current OpenEMR version by reading `/var/www/openemr/version.php` or checking Admin > About within the UI. Take a filesystem snapshot or VM snapshot before applying updates. Apply patches via the OpenEMR upgrade script documented at openemr.org/wiki and validate success by re-checking the version string post-upgrade. For non-production environments with no snapshot capability, use `sha256sum` on key application files (e.g., `library/sqlquery.inc.php`, `interface/main/tabs/main.php`) before and after to confirm file replacement.

Evidence: Capture the pre-patch OpenEMR version string, PHP version (`php -v`), web server version (`apache2 -v` or `nginx -v`), and database version (`mysql --version`) before patching. Archive the OpenEMR `openemr.log` and web server access logs from the 30 days prior to patching — these are your pre-remediation forensic baseline. If SQL injection or RCE exploitation occurred before patching, the patched codebase will not show exploitation artifacts; you need the pre-patch log archive.

Step 3: Review web-facing attack surface — confirm whether your OpenEMR instance is internet-accessible or reachable from untrusted network segments; if so, implement WAF rules covering SQL injection and command injection patterns as a compensating control while patching proceeds.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy: limiting exposure while eradication is in progress

Controls: NIST SC-7 (Boundary Protection) — requires monitoring and controlling communications at external boundaries; internet-exposed OpenEMR with unpatched SQLi/RCE is a direct boundary protection failure, NIST SI-3 (Malicious Code Protection) — WAF rules blocking SQLi and command injection patterns function as entry-point malicious input filters, CIS 4.4 (Implement and Manage a Firewall on Servers) — restrict inbound access to OpenEMR

ports to authorized source IPs only; if internet access is not operationally required, block it entirely at the host firewall, CIS 4.5 (Implement and Manage a Firewall on End-User Devices) — ensure clinical workstations accessing OpenEMR are not on segments reachable from untrusted networks

Compensating: If no commercial WAF is available, deploy ModSecurity (free, Apache/nginx module) with the OWASP Core Rule Set (CRS) — specifically enable rules in the `REQUEST-942-APPLICATION-ATTACK-SQLI.conf` and `REQUEST-932-APPLICATION-ATTACK-RCE.conf` files, which directly cover the SQL injection and command injection classes identified in this disclosure. As an immediate stopgap, restrict OpenEMR access to a specific allowlist of clinical IP ranges using Apache `` directives or nginx `allow`/`deny` blocks. Verify the restriction is effective with `curl -I https:///` from an unauthorized IP — a 403 confirms the block.

Evidence: Before implementing WAF rules or IP restrictions, export the current Apache/nginx virtual host configuration and any existing `.htaccess` files in the OpenEMR web root (`/var/www/openemr/`) — these establish what protections (if any) were in place prior to this response. Document the network path to OpenEMR using `traceroute` or firewall rule exports to confirm whether the instance was reachable from the internet or untrusted VLANs during the window of vulnerability.

Step 4: Audit for indicators of prior compromise — review web server logs, application logs, and database query logs for anomalous patterns consistent with SQL injection probing (T1190), unexpected file writes or new files in web directories (T1505.003), and unusual outbound data transfers (T1048, T1041) covering the period prior to patch availability.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis: analyzing available evidence to determine whether exploitation occurred before containment

Controls: NIST AU-6 (Audit Record Review, Analysis, and Reporting) — requires reviewing logs for indicators of compromise at a frequency commensurate with risk; this disclosure triggers an immediate unscheduled review, NIST AU-12 (Audit Record Generation) — verify that OpenEMR application logging, web server access logging, and MySQL general/slow query logging were enabled during the exposure window, NIST SI-4 (System Monitoring) — requires monitoring for indicators consistent with attacks; SQLi probing against OpenEMR endpoints and web shell drops are specific to this disclosure, CIS 8.2 (Collect Audit Logs) — confirms log collection was active; if logs are absent for the exposure window, treat the gap as a presumptive compromise indicator

Compensating: Query Apache/nginx access logs for SQLi patterns specific to OpenEMR endpoints: `grep -E "(UNION|SELECT|INSERT|DROP|--|\'|%27|%3B)" /var/log/apache2/access.log | grep -i openemr`. Check for web shell artifacts dropped via T1505.003 by running `find /var/www/openemr/ -name '*.php' -newer /var/www/openemr/version.php -ls` — any PHP files newer than the known installation date are suspicious. For outbound data exfiltration (T1048/T1041), review MySQL general query log at `/var/log/mysql/mysql.log` for large `SELECT *` queries against patient tables (`patient_data`, `form_encounter`, `openemr_postcalendar_events`) executed by the web application user. Use `tcpdump -i eth0 -w capture.pcap 'dst port not 80 and dst port not 443'` to capture anomalous outbound traffic if the system is still running.

Evidence: Preserve the following before any remediation actions that could overwrite artifacts: (1) Apache/nginx access logs in `/var/log/apache2/` or `/var/log/nginx/` for the full exposure window — these will contain SQLi probe URIs and POST body evidence if logging is verbose; (2) MySQL general query log and slow query log showing queries executed by the OpenEMR database user (`openemr` or equivalent) — SQLi exploitation will appear as syntactically anomalous queries or stacked queries; (3) OpenEMR application log at `/logs/` for PHP errors triggered by malformed injection payloads; (4) filesystem metadata — run `find /var/www/openemr/ -newer -type f` to identify files written during the exposure window, which may indicate web shell installation via T1505.003; (5) PHP session files in `/var/lib/php/sessions/` or `/tmp/` for evidence of authenticated session hijacking following credential theft via SQLi.

Step 5: Update threat model — add OpenEMR-class open-source EHR software to your third-party risk register; validate that vendor security assessment processes cover open-source dependencies used by clinical partners, not only commercial software.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: using lessons learned to update risk posture and improve detection for future incidents

Controls: NIST RA-3 (Risk Assessment) — requires assessing risk from third-party systems including open-source software used by clinical partners; OpenEMR's deployment profile (100,000+ providers, open-source, internet-exposed) must now appear in the organizational risk register, NIST SA-9 (External System Services) — requires monitoring and assessing third-party service providers who process organizational data, including clinical vendors running OpenEMR on your behalf, NIST IR-8 (Incident Response Plan) — post-incident updates to the IR plan must incorporate open-source EHR software as a named threat surface, CIS 2.1 (Establish and Maintain a Software Inventory) — the software inventory must now include open-source EHR platforms deployed by clinical partners, not only internally managed software

Compensating: Create a simple CSV-based third-party risk register entry for OpenEMR with fields: vendor/partner name, OpenEMR version deployed, internet exposure status, last patched date, and data types processed (PHI categories). Review existing Business Associate Agreements (BAAs) with clinical partners to confirm they require timely notification of software vulnerabilities affecting PHI systems — if BAAs are silent on open-source software, flag for legal review. Add OpenEMR to a recurring quarterly check against the NVD CVE feed using a free RSS alert for CPE `cpe:2.3:a:open-emr:openemr`.

Evidence: Document the current state of your third-party risk register before updating it — this creates an audit trail showing that OpenEMR-class software was not previously assessed, which is itself a finding for your HIPAA Security Rule gap analysis. Collect copies of existing vendor contracts and BAAs for clinical partners identified in Step 1 as running OpenEMR; these are required evidence for any subsequent HIPAA breach investigation or OCR audit.

Step 6: Communicate findings — brief clinical operations and compliance leadership on patient data exposure risk with specific reference to the vulnerability classes involved; frame remediation timeline against HIPAA breach risk thresholds.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment: internal communication and coordination with stakeholders during active response

Controls: NIST IR-6 (Incident Reporting) — requires reporting suspected incidents to organizational leadership within defined timeframes; SQL injection and RCE vulnerabilities in a PHI-processing system meet the threshold for escalation to compliance leadership, NIST IR-4 (Incident Handling) — incident handling capability must include communication procedures that reach clinical operations stakeholders, not only IT, NIST AU-3 (Content of Audit Records) — briefing materials must reference specific evidence: which vulnerability classes (SQLi, RCE, deserialization), which data stores were accessible (patient_data table, document storage), and what the exploitation window was

Compensating: Prepare a one-page breach risk assessment memo using the HHS four-factor test (45 CFR §164.402) as the structure: (1) nature and extent of PHI involved — OpenEMR stores demographics, diagnoses, medications, and clinical notes; SQLi and RCE provide direct database and filesystem access to all of these; (2) who accessed or could have accessed PHI — unknown during the exposure window, which defaults toward presumptive breach under HIPAA; (3) whether PHI was actually acquired or viewed — log analysis from Step 4 is the primary evidence; (4) extent to which risk has been mitigated — patching status and WAF controls from Steps 2 and 3. Deliver this memo to the Privacy Officer and Compliance Officer within 24 hours of completing Steps 1-4, as the 60-day HIPAA breach notification clock may have already started.

Evidence: Before the compliance briefing, compile: the OpenEMR version confirmation from Step 2, the log analysis findings from Step 4 (including any confirmed SQLi probe attempts or absence of logs), the network exposure determination from Step 3, and the third-party vendor list from Step 1. These form the factual basis of the briefing and must be preserved as records in the event of an OCR investigation. If log analysis is incomplete at briefing time, state that explicitly — do not brief leadership with a false 'no evidence of compromise' conclusion when log coverage gaps exist.

Step 7: Monitor developments — track NVD for individual CVE assignments across the 38 findings, monitor CISA for any KEV additions, and watch for AISLE's full technical disclosure for IOC publication.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: sustained monitoring and intelligence integration following initial response

Controls: NIST SI-5 (Security Alerts, Advisories, and Directives) — requires receiving and acting on security advisories from external organizations on an ongoing basis; NVD CVE assignments and CISA KEV additions for these 38 findings are covered advisories, NIST IR-5 (Incident Monitoring) — requires tracking and documenting incidents including their evolving status; as CVEs are assigned and exploitation activity is reported, the incident record must be updated, NIST AU-6 (Audit Record Review, Analysis, and Reporting) — if AISLE publishes IOCs (file hashes, URI patterns, network indicators), trigger a retroactive log review against those specific indicators in addition to the pattern-based review in Step 4, CIS 7.1 (Establish and Maintain a Vulnerability Management Process) — the vulnerability management process must include a mechanism for consuming NVD updates and CISA KEV additions for software in the asset inventory

Compensating: Set up free NVD RSS feed monitoring for OpenEMR CPE `cpe:2.3:a:open-emr:openemr` using a feed reader or a cron job that curls `https://services.nvd.nist.gov/rest/json/cves/2.0?cpeName=cpe:2.3:a:open-emr:openemr:*` and diffs the output daily. Subscribe to the CISA KEV catalog change feed (`https://www.cisa.gov/sites/default/files/feeds/known_exploited_vulnerabilities.json`) with a script that alerts on new entries matching 'OpenEMR' or 'open-emr'. When AISLE publishes full technical disclosure, extract any YARA rules or Sigma rules provided and run them against the log archives preserved in Step 4 using `yara /var/www/openemr/` and `sigma convert` against your log files.

Evidence: Maintain a living incident ticket that is updated each time a new CVE is assigned from the 38 findings — record the CVE ID, CVSS score, CWE class, and whether it maps to an exploit technique already covered in your Step 4 log review. If any of the 38 CVEs receive a CISA KEV entry, that is a mandatory trigger to re-execute the compromise audit from Step 4 with the specific IOCs published in the KEV entry, as KEV addition indicates confirmed active exploitation in the wild and elevates the presumptive breach risk under HIPAA.

Detection Guidance

Focus detection efforts on three surfaces: the web application layer, the database tier, and outbound data movement.

Web application logs: Hunt for SQL injection pattern strings in HTTP request parameters and URI paths, single quotes, comment sequences (-- and #), UNION SELECT constructs, and encoded variants. Flag requests returning anomalous data volumes or unexpected HTTP 500 responses from database errors. Review for path traversal sequences (../, %2e%2e%2f) in file path parameters.

Application and OS logs: Look for unexpected process spawning from the web server process (e.g., Apache or PHP spawning shell processes), which is a strong indicator of OS command injection (CWE-78) or code injection (CWE-94) exploitation. On Linux hosts, audit /var/log/auth.log and command history for unusual commands executed under the web server service account.

Database logs: Enable and review query logs for anomalous SELECT patterns against credential tables, patient record tables, or schema enumeration queries (INFORMATION_SCHEMA access). Unexpected bulk reads of patient data warrant immediate investigation.

File integrity monitoring: Monitor the OpenEMR web root and adjacent directories for new or modified PHP files; web shell deployment (T1505.003) is a reliable post-exploitation step following RCE.

Network monitoring: Watch for unusual outbound connections from OpenEMR application servers, particularly to external IPs on non-standard ports, large data transfers during off-hours, or connections to cloud storage endpoints not in your baseline (T1048, T1041). Deserialization exploitation (CWE-502) may also trigger outbound callback connections during payload delivery.

SIEM correlation: Build detection rules correlating web application firewall alerts on injection patterns with database query anomalies and process execution events on the same host within a short time window; the combination reduces false positives while catching multi-stage exploitation.

Indicators of Compromise

Type	Value	Context	Confidence
TOOL	`Pending — refer to AISLE blog (aisle.com) and individual CVE NVD entries for published technical indicators`	AISLE's disclosure references 38 CVEs with associated technical findings; specific exploit code, payload hashes, or network indicators were not published in the source articles reviewed. Check the AISLE disclosure report and NVD entries once individual CVE identifiers are confirmed for any associated proof-of-concept or IOC publication.	LOW

Framework Mappings

MITRE-ATTACK

- **T1048** — Exfiltration Over Alternative Protocol
- **T1486** — Data Encrypted for Impact
- **T1078** — Valid Accounts
- **T1041** — Exfiltration Over C2 Channel
- **T1190** — Exploit Public-Facing Application
- **T1505.003** — Web Shell
- **T1005** — Data from Local System
- **T1059** — Command and Scripting Interpreter
- **T1213** — Data from Information Repositories
- **T1565.001** — Stored Data Manipulation

NIST-800-53R5

- **CP-9** — System Backup
- **CP-10** — System Recovery and Reconstitution
- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **SI-4** — System Monitoring
- **CA-8** — Penetration Testing

- **RA-5** — Vulnerability Monitoring and Scanning
- **SI-2** — Flaw Remediation
- **SI-7** — Software, Firmware, and Information Integrity
- **CM-2** — Baseline Configuration
- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-10** — Information Input Validation
- **AC-3** — Access Enforcement
- **SC-28** — Protection of Information at Rest
- **IR-4** — Incident Handling

OWASP-TOP10-2021

- **A03:2021** — Injection
- **A01:2021** — Broken Access Control
- **A08:2021** — Software and Data Integrity Failures

CIS-V8

- **2.5** — Allowlist Authorized Software
- **16.10** — Apply Secure Design Principles in Application Architectures
- **16.12** — Implement Code-Level Security Checks
- **7.3** — Perform Automated Operating System Patch Management
- **7.4** — Perform Automated Application Patch Management

HIPAA-SECURITY

- **164.312(a)(1)** — Access Control
- **164.308(a)(7)(ii)(A)** — Data Backup Plan
- **164.308(a)(6)(ii)** — Response and Reporting

ISO-27001-2022

- **A.8.28** — Secure coding
- **A.5.29** — Information security during disruption
- **A.8.8** — Management of technical vulnerabilities
- **A.5.34** — Privacy and protection of personal information
- **A.5.23** — Information security for use of cloud services

NIST-CSF-2

- **RS.MI-01** — Incidents are contained

SOC2-TSC

- **CC7.4** — Responds to identified security incidents

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1048	Exfiltration Over Alternative Protocol	Exfiltration
T1486	Data Encrypted for Impact	Impact
T1078	Valid Accounts	Defense-Evasion
T1041	Exfiltration Over C2 Channel	Exfiltration
T1190	Exploit Public-Facing Application	Initial-Access
T1505.003	Web Shell	Persistence
T1005	Data from Local System	Collection
T1059	Command and Scripting Interpreter	Execution
T1213	Data from Information Repositories	Collection
T1565.001	Stored Data Manipulation	Impact

Sources

Source	URL	Tier
Security News	https://www.darkreading.com/vulnerabilities-threats/ai-finds-38-sec...	T3
38 Vulnerabilities Found in OpenEMR Medical Software	https://www.securityweek.com/38-vulnerabilities-found-in-openemr-me...	T3
AISLE Discovers 38 CVEs in Healthcare Software Used by 100000 ...	https://aisle.com/blog/aisle-discovers-38-critical-security-vulnera...	T3
Researchers Find 38 Flaws in OpenEMR. They've Been Fixed	https://www.govinfosecurity.com/researchers-find-38-flaws-in-openem...	T3
AISLE Discovers 38 CVEs in OpenEMR Healthcare Software	https://news.ycombinator.com/item?id=47936347	T3

Generated 2026-04-30 06:30 UTC by TJS Security Command Center