



PHP Out-of-Memory Request Corruption Remote Code Execution Vulnerability (PHP2Shell)

CVE-2026-109618 - Published: 2026-10-21

CRITICAL

CVSS 9.8

EPSS score: 5.31% | CISA KEV: NO | VulnCheck KEV: YES | Ransomware use: Unknown

CVSS interpretation

The CVSS score of 9.8 (Critical) is driven by several compounding metrics: Network Attack Vector (AV:N) means an attacker can exploit it remotely across a network without physical access; Low Attack Complexity (AC:L) indicates that no specialized conditions or complex bypasses are required; and Privileges Required (PR:N) and User Interaction (UI:N) mean anyone on the internet can launch an attack anonymously without a user clicking a link. Combined with High impacts across Confidentiality, Integrity, and Availability (C:H/I:H/A:H), an attacker who successfully exploits this flaw achieves total control over the affected application worker, allowing them to steal sensitive data, modify clinical records, or disrupt system availability.

Executive summary

A critical remote code execution vulnerability, commonly tracked as part of the 'PHP2Shell' flaws, affects the PHP web programming language. An unauthenticated attacker over the network can send malicious requests that trigger memory exhaustion errors, causing the application to improperly handle internal memory and execute arbitrary code. The vulnerability carries a maximum CVSS base score of 9.8, indicating extreme severity due to the lack of required authentication or user interaction. For healthcare organizations and biomedical engineering teams, this flaw presents severe risks if web-based clinical management portals, medical device middleware, or administrative servers rely on vulnerable PHP runtimes. Immediate patching and web application shielding are essential to prevent complete system compromise.

What this vulnerability is

This vulnerability is a flaw in how the PHP programming language engine manages memory when a web request runs out of memory (an Out-of-Memory or OOM error). PHP uses a low-level error-handling mechanism to abruptly abort requests when they exceed memory limits, but under certain conditions, it leaves data structures in an uninitialized or corrupt state. Think of it like a safety mechanism in a medical pump that abruptly stops an infusion due to an error, but accidentally leaves valves in a half-open state, causing subsequent operations to read dangerous, unintended data. It impacts any application built on vulnerable versions of the PHP runtime environment.

How it works

When a PHP script hits a memory limit, the engine executes an abrupt jump function to escape the current routine and initiate a cleanup procedure. Due to flawed control flow logic, the cleanup routine mistakes uninitialized or half-baked memory pointers for valid data and attempts to read or branch on them. An attacker can systematically trigger these out-of-memory states using carefully crafted network requests, forcing the system to execute attacker-supplied instructions. This grants the attacker unauthorized remote code execution capabilities on the underlying server without needing any login credentials.



Real-world exploitability

Exploiting this vulnerability in the wild is relatively straightforward for attackers once they target an exposed, unpatched PHP endpoint, with active public exploit concepts and tracking repositories emerging rapidly after disclosure. Because the attack requires zero authentication and no user interaction, automated scanning tools can easily scan the internet and bombard endpoints with OOM-triggering payloads. Attackers require baseline networking knowledge to construct the payloads, but no advanced or privileged system access beforehand.

Consequences if exploited

If successfully exploited, an adversary can execute arbitrary code with the permissions of the PHP web worker process. In an operational healthcare technology context, this could result in the theft of Protected Health Information (PHI), modification of medical middleware routing data, complete denial of service for administrative portals, or lateral movement into connected hospital network segments.

Healthcare and medical device relevance

Healthcare environments frequently utilize PHP-based applications for intranet portals, clinical trial management systems, asset tracking interfaces, and vendor-supplied medical device management servers. Security advisories from health entities like NHS England emphasize that unpatched PHP runtimes expose critical infrastructure to remote takeovers. Biomedical engineers and clinical security leads must inventory all standalone servers, virtual machines, and embedded appliance web interfaces that bundle PHP, particularly those exposed to hospital enterprise networks or the broader internet.

Weakness classifications (CWE)

CWE-416 - Use After Free

This weakness occurs when a program continues to use a memory pointer after that memory has been freed or invalidated. In this vulnerability, aborted request handling leaves memory references unstable or mismanaged, leading the cleanup routine to interact with corrupted memory allocations.

CWE-824 - Access of Uninitialized Pointer

This refers to software referencing a pointer before it has been properly assigned a valid memory address or value. Because the OOM error bypasses normal initialization sequences, the system treats raw, uninitialized memory as valid instructions during request teardown.

CWE-755 - Improper Handling of Exceptional Conditions

This weakness happens when software fails to securely handle unexpected runtime errors or exceptions. Here, the emergency exception path used for memory exhaustion fails to safely clean up context states, exposing the application to deep logic corruption.

Remediation and mitigation

1. Upgrade PHP immediately to the official patched versions: 8.5.12, 8.4.27, 8.3.36, 8.2.35, or later.
2. Rebuild and redeploy all container images and virtual machine base templates that package their own internal PHP runtime environments.
3. Deploy temporary compensating controls such as Web Application Firewall (WAF) rules or signature blocks to



intercept malicious out-of-memory exploitation patterns if patching must follow standard clinical change management windows.

Key references - Microsoft MSRC

Not found in MSRC.

NVD references

- <https://github.com/php/php-src/security/advisories/GHSA-w8hw-w9fx-j79w>
- <http://www.openwall.com/lists/oss-security/2026/10/21/6>
- <https://www.wiz.io/blog/php2shell-php-rce>

Generated by VulnBrief | Model: gemini-3.5-flash-lite | 2026-10-22 18:59 UTC