



PHP Runtime Memory Corruption and Remote Code Execution Vulnerability (PHP2Shell)

CVE-2026-109619 - Published: 2026-10-21

CRITICAL

CVSS 9.8

EPSS score: 5.76% | CISA KEV: NO | VulnCheck KEV: YES | Ransomware use: Unknown

CVSS interpretation

The CVSS base score of 9.8 (Critical) is driven by several compounding factors: Attack Vector (AV:N) means it can be exploited remotely across a network without physical access, and Privileges Required (PR:N) alongside User Interaction (UI:N) indicates that an attacker needs zero authentication and zero user assistance to succeed. Combined with an Unchanged Scope (S:U) and High impacts across Confidentiality, Integrity, and Availability (C:H/I:H/A:H), a successful exploit gives an adversary complete control over the affected application worker and its underlying data.

Executive summary

A critical vulnerability known as PHP2Shell has been discovered in the core PHP programming language runtime, carrying a maximum CVSS score of 9.8. The flaw stems from improper handling of global system states and memory during request execution and error handling, allowing unauthenticated remote attackers to execute arbitrary code on target servers. Although active exploitation frameworks and threat intel trackers confirm public proof-of-concept exploits exist, it is not currently listed in CISA's Known Exploited Vulnerabilities catalog. Healthcare security leads and biomedical engineers must evaluate clinical web applications, middleware, and biomedical data servers running vulnerable PHP versions to prevent potential system compromise and operational disruption.

What this vulnerability is

This vulnerability affects the core PHP language runtime engine, which powers millions of web applications, administrative portals, and backend biomedical data dashboards. Think of PHP as an engine processing incoming data requests; normally, it cleanly allocates memory for a task and erases it when done. In this flaw, certain error-handling pathways or states fail to clear pointers properly, leaving behind 'ghost' references to memory slots that have already been deleted or freed.

How it works

Mechanically, when an attacker sends a carefully crafted web request that triggers a fatal error or specific state change (such as via upload progress tracking), PHP's internal cleanup routines fail to reset global variables. These variables continue pointing to memory addresses that are later reallocated or read during subsequent requests. An unauthenticated attacker can exploit this stale memory state to corrupt application memory and achieve remote code execution over the network without needing any credentials.

Real-world exploitability

Exploiting this vulnerability requires low attack complexity and no prior authentication, meaning script kiddies or automated scanner bots can easily target exposed endpoints. Threat intelligence sources note that exploit code and detection artifacts have spread rapidly across security repositories since disclosure, yielding an elevated EPSS



exploitation probability. However, successful exploitation depends on targeting specific vulnerable PHP branch versions (such as unpatched 8.2, 8.3, 8.4, or 8.5 releases) where input handling triggers the memory reuse bug before application code even executes.

Consequences if exploited

If successfully exploited, an attacker can achieve arbitrary remote code execution within the context of the web server user account. Operationally, this can result in full data theft or tampering of electronic health records, unauthorized modification of medical device middleware databases, or complete denial-of-service affecting clinical portals and patient management workflows.

Healthcare and medical device relevance

Healthcare environments frequently utilize custom or commercial PHP-based applications for hospital intranet portals, asset tracking, biomedical equipment data aggregation, and electronic medical record (EMR) auxiliary interfaces. Because this flaw bypasses standard application authentication and executes early in the request lifecycle (such as during upload progress routines), standard web application firewalls (WAFs) relying on large request body inspection may fail to block the small ~450-byte exploit payload. Biomedical engineers and hospital IT security teams must urgently audit web servers hosting medical software stacks to inventory exposed PHP runtimes, as an operational disruption could directly delay clinical decision-making or patient care delivery.

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 deleted. In CVE-2026-109619, global PHP state pointers reference memory regions that have already been released, allowing attackers to manipulate newly allocated data in those same slots.

CWE-824 - Access of Uninitialized Pointer

This weakness happens when software tries to use a pointer before it has been properly assigned a valid memory address. In this scenario, subsequent requests read or branch on stale, uninitialized state left behind in memory, enabling execution flow hijacking.

CWE-755 - Improper Handling of Exceptional Conditions

This refers to flaws in how software reacts to unexpected errors or runtime failures. During fatal error handling (such as longjmp events), PHP fails to safely reset global states, paving the way for memory corruption.

Remediation and mitigation

1. Immediately upgrade PHP to the official patched versions released by the vendor (e.g., PHP 8.2.35, 8.3.36, 8.4.27, 8.5.12, or later depending on your branch).
2. For unsupported legacy PHP branches (versions 5.4 through 8.1 which lack upstream vendor patches), isolate applications behind strict network segmentation or migrate workloads immediately to a supported runtime.
3. As a temporary compensating control if patching is delayed, disable vulnerable upload progress callbacks or session upload tracking vectors (such as setting `session.upload_progress.enabled = Off`) where applicable, and restrict network access to clinical web servers via strict internal firewalls and zero-trust proxies.



Key references - Microsoft MSRC

Not found in MSRC.

NVD references

- <https://github.com/php/php-src/security/advisories/GHSA-6wmf-4rv5-p3rp>
- <http://www.openwall.com/lists/oss-security/2026/10/21/16>
- <http://www.openwall.com/lists/oss-security/2026/10/21/17>
- <http://www.openwall.com/lists/oss-security/2026/10/21/23>
- <http://www.openwall.com/lists/oss-security/2026/10/21/7>
- <http://www.openwall.com/lists/oss-security/2026/10/22/2>
- <https://www.wiz.io/blog/php2shell-php-rce>

Generated by VulnBrief | Model: gemini-3.5-flash-lite | 2026-10-22 19:04 UTC