



BGD e-GOV CIRT

TLP: CLEAR



CYBER THREAT ADVISORY

**PHP2Shell: Critical
Unauthenticated Remote Code
Execution Vulnerabilities in PHP**

**TLP:** CLEAR**Distribution:** Public**Advisory on:** PHP2Shell: Critical Unauthenticated Remote Code Execution Vulnerabilities in PHP**Severity:** High**Vulnerability Type:** Remote Code Execution (RCE)**CVE IDs:** CVE-2026-109618, CVE-2026-109619**CVSS v4.0:** 9.2**Potential Impact:** Server compromise, unauthorized command execution, application data exposure, credential theft and service disruption**Date:** 22 October 2026

Executive Summary

BGD e-GOV CIRT alerts organizations to two critical vulnerabilities in the PHP runtime, collectively discussed under the name **PHP2Shell**. The vulnerabilities can enable unauthenticated remote code execution in vulnerable PHP deployments, potentially allowing an attacker to execute code on a web server without first authenticating to the application.

The vulnerabilities affect PHP request-processing and internal state-management behavior:

- **CVE-2026-109618:** Remote code execution associated with out-of-memory (OOM) handling and corruption of partially initialized request data.
- **CVE-2026-109619:** Remote code execution associated with stale global state and the use of request-related state after its backing memory has been freed.

The official security notice identifies PHP-FPM, Apache with mod_php, php-cgi, FrankenPHP and LiteSpeed deployments as potentially at risk. It highlights particular concern for mod_php, where an unauthenticated request has been demonstrated to trigger exploitation before application code executes.

Fixed releases are available for **PHP 8.5, 8.4, 8.3 and 8.2**. **Organizations should upgrade the PHP runtime immediately**. Configuration changes and network restrictions can reduce exposure while patching is arranged, but they do not provide a complete fix.

Vulnerability Details

CVE-2026-109618 — Out-of-Memory Handling

The vulnerability concerns PHP's handling of an out-of-memory condition while internal request data is only partially initialized. During request cleanup, the runtime may incorrectly treat invalid or incomplete data as valid.

Under exploitable conditions, this behavior can potentially lead to remote code execution.

CVE-2026-109619 — Stale Global State

This vulnerability concerns request-owned internal state that can remain accessible after the associated memory has been freed. A subsequent request handled by the same worker may use that stale state before normal initialization resets it.

Under exploitable conditions, the resulting invalid state handling can potentially lead to remote code execution.

These descriptions reflect the published vulnerability mechanisms; they should not be interpreted as evidence that a particular Bangladesh-based system has been exploited.

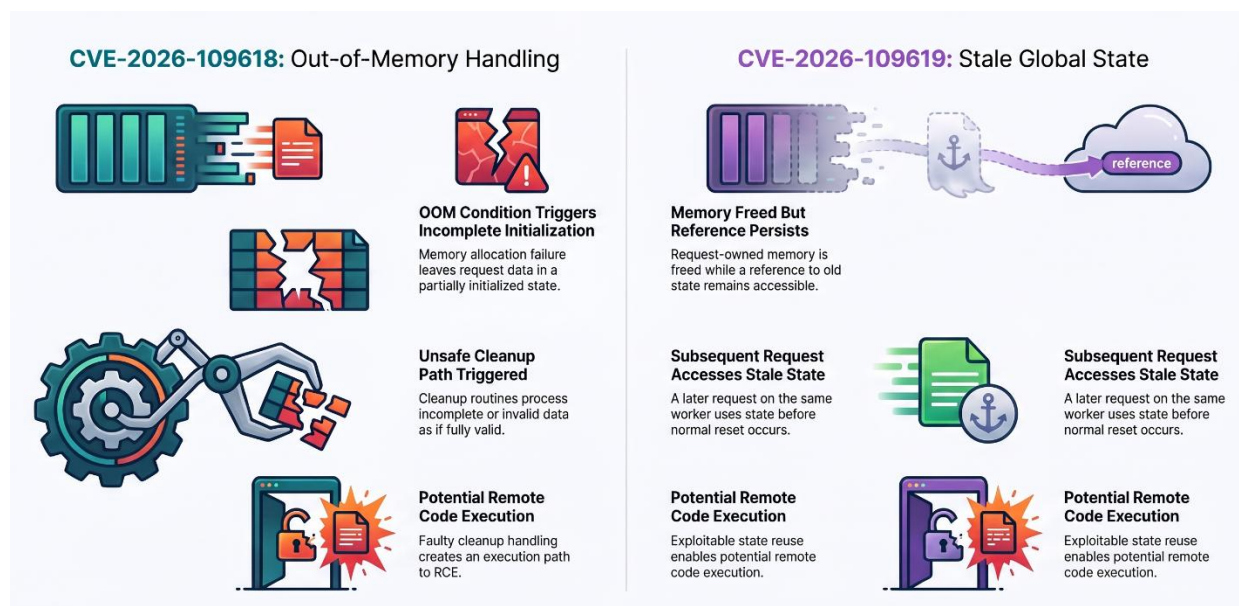


Figure: Comparison of the memory-handling failures associated with CVE-2026-109618 and CVE-2026-109619

Affected Environments

Organizations should identify PHP installations running web-facing applications, including:

- PHP-FPM behind a web server or reverse proxy.
- Apache using mod_php.
- php-cgi deployments.
- FrankenPHP deployments.
- LiteSpeed deployments.
- PHP runtimes inside containers and application images.
- Managed or shared hosting environments.
- Web applications built with PHP frameworks or content-management systems.

Applications using PHP may include government portals, public-facing APIs, administrative systems, e-commerce platforms and internal web applications.

Important: Updating an application or framework does not necessarily update the PHP runtime underneath it. Administrators must verify that the runtime serving web requests has received the security fix.

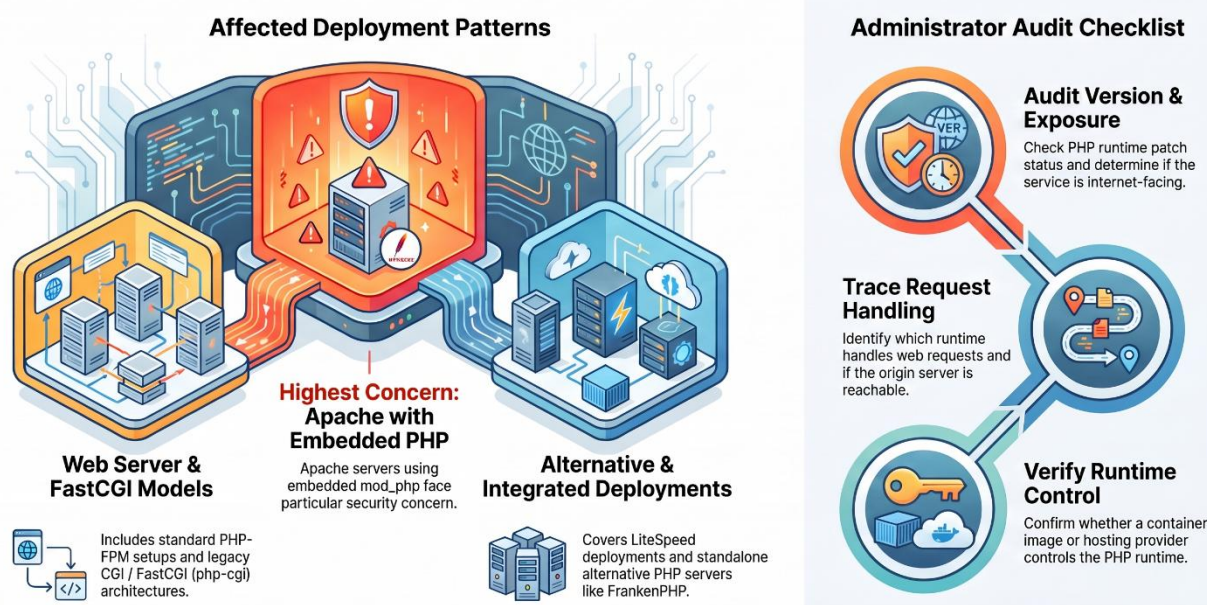


Figure: Common PHP deployment architectures requiring exposure and patch-status verification

Fixed PHP Versions

The following upstream releases contain fixes for both vulnerabilities:

PHP branch	Fixed version
PHP 8.5	8.5.12
PHP 8.4	8.4.27
PHP 8.3	8.3.36
PHP 8.2	8.2.35

Organizations should upgrade to the applicable fixed release or install a verified security-patched package provided by their operating-system distribution or hosting provider.

PHP 8.1 and older branches no longer receive regular upstream security updates. Where such versions remain in use, organizations should upgrade to a supported branch or verify that a trusted distribution has supplied an appropriate backported fix. PHP 8.2 is scheduled to reach end of life on 31 December 2026, so organizations should also plan their migration to a longer-supported branch.

Technical Risk and Potential Attack Path

The principal concern is that exploitation may occur through web requests handled by a vulnerable PHP runtime, without requiring an attacker to compromise the application account first.

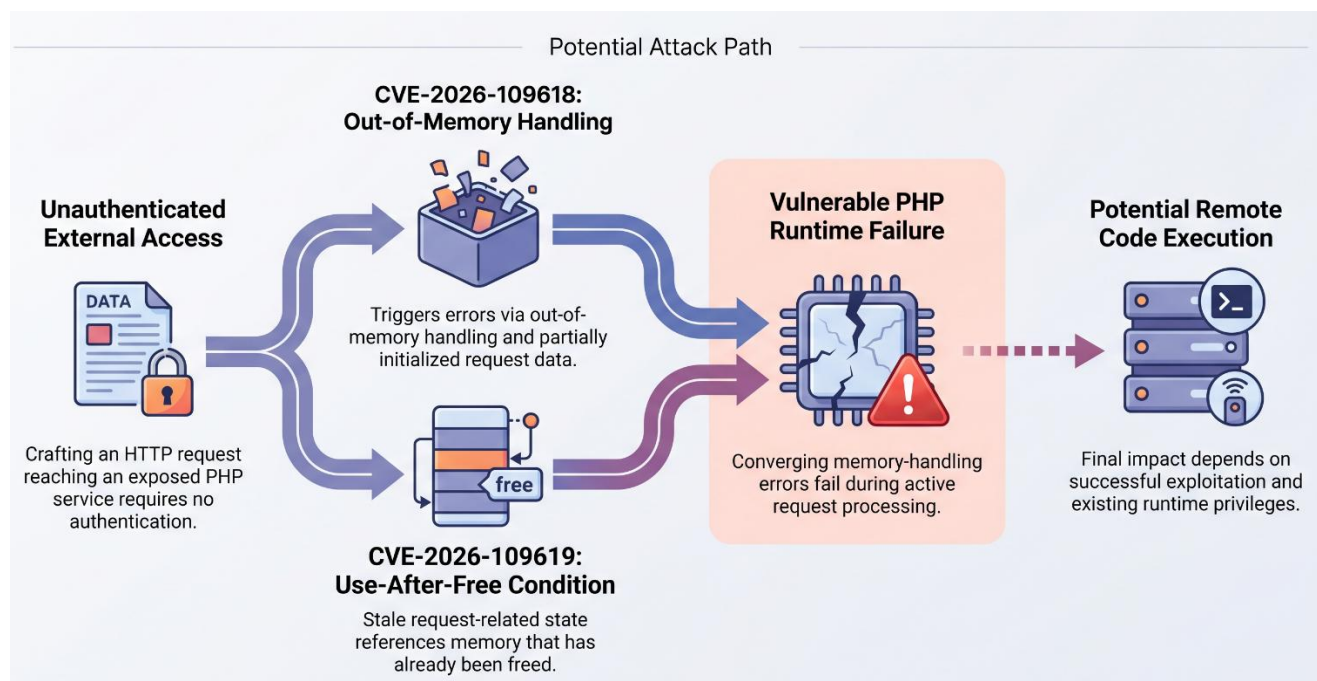


Figure: Overview of the PHP2Shell vulnerabilities and potential path from unauthenticated request processing to remote code execution

A potential attack scenario is:

1. An attacker identifies an Internet-accessible server running a vulnerable PHP runtime.
2. The attacker sends crafted requests that reach the vulnerable request-processing path.
3. The PHP runtime encounters the vulnerable memory-handling or stale-state condition.
4. If exploitation succeeds, the attacker may obtain code execution in the security context of the PHP worker or associated service.
5. The attacker may then attempt to access application files, environment variables, database credentials, stored secrets or other resources permitted to that process.

The final stages are **potential post-exploitation consequences**, not confirmed outcomes for every vulnerable installation.

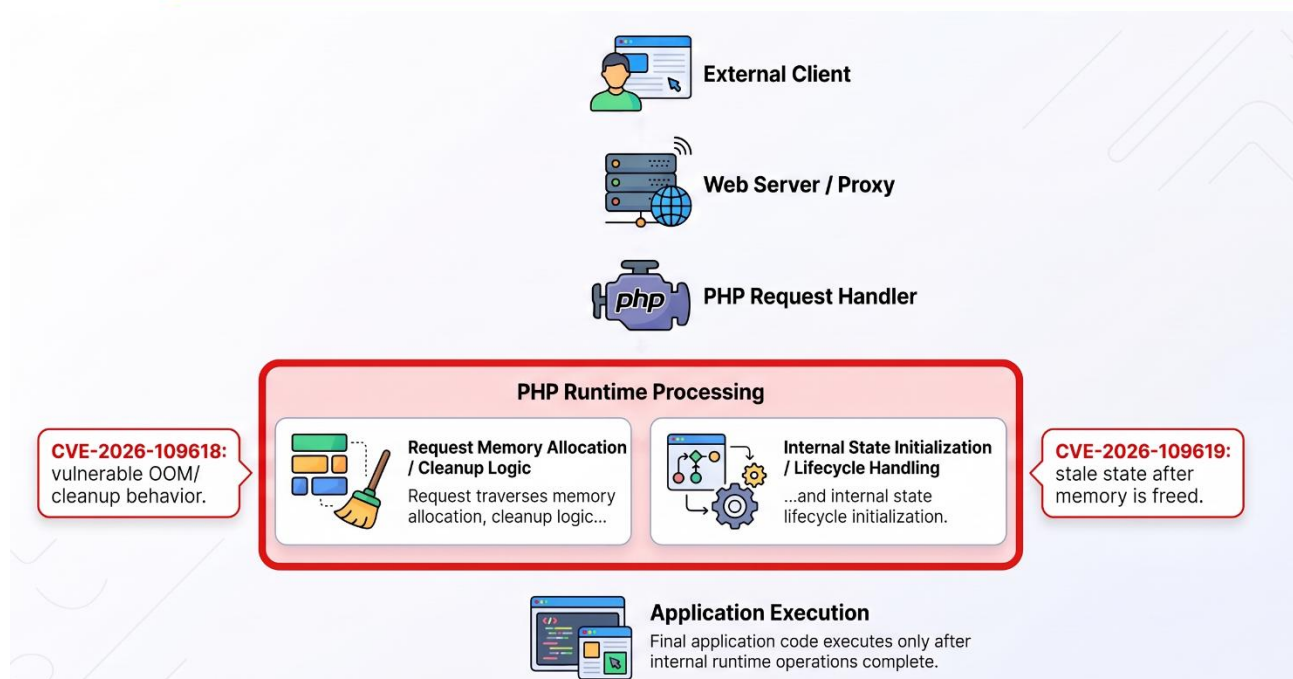


Figure: Conceptual PHP request-processing boundary illustrating where the documented memory-handling vulnerabilities arise

Security Impact

Successful exploitation could potentially result in:

- **Server compromise:** Unauthorized execution of code in the PHP service's security context.
- **Confidentiality loss:** Access to application data, files, configuration or credentials available to the process.
- **Credential exposure:** Theft of database passwords, API keys, connection strings or other secrets stored in accessible configuration.
- **Lateral movement:** Attempts to access other systems using credentials or permissions obtained from the compromised server.
- **Service disruption:** Interference with application availability or server operations.
- **Further compromise:** Installation of malicious scripts or tools, depending on the attacker's privileges and access.

The actual impact depends on the PHP deployment model, process privileges, network reachability, application architecture and available security controls.

Immediate Mitigation

Upgrade PHP

Upgrade the runtime to the applicable fixed release as soon as possible. Confirm the patch status of the runtime that handles production traffic, not only the command-line PHP installation.

Restrict Exposure Until Patched

Where immediate patching is not possible:

- Temporarily remove vulnerable services from public access when operationally feasible.
- Place affected applications behind VPN access, authentication controls or strict source-IP allowlisting.
- Ensure attackers cannot bypass a reverse proxy or access the origin server directly.
- Restrict inbound traffic to required services and authorized sources.

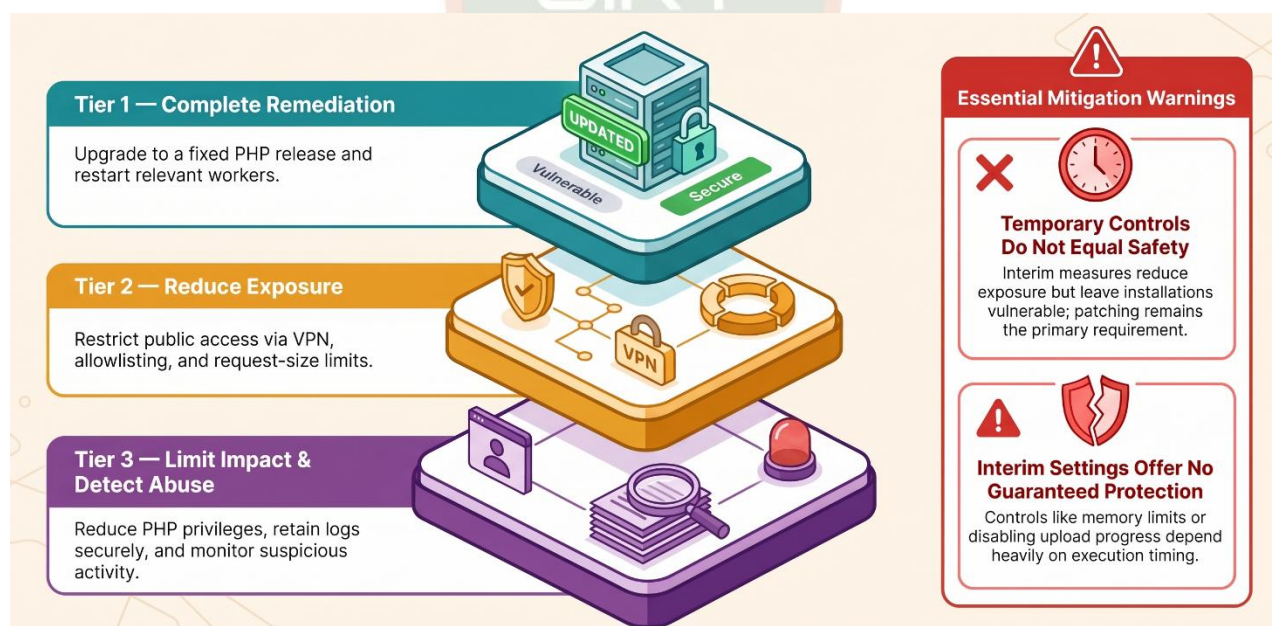


Figure: Recommended mitigation hierarchy for vulnerable PHP deployments

Reduce Potential Impact

- Run PHP workers with the minimum required operating-system permissions.
- Restrict access to application secrets and cloud credentials.
- Remove unnecessary credentials from application environments.
- Limit access from web servers to databases and internal services.
- Ensure logs are enabled and stored in a protected location.

These measures reduce exposure or limit the consequences of a compromise; **they do not replace the security update.**

Configuration-Based Risk Reduction

The official security notice describes temporary settings that may reduce exposure to specific paths associated with these vulnerabilities. Administrators should apply them only after assessing application compatibility and operational risk.

For **CVE-2026-109618**, published risk-reduction measures include:

- Setting `memory_limit=-1` in an appropriate administrator-controlled PHP configuration where feasible.
- Restricting `post_max_size` according to application requirements.
- Enforcing request-size, multipart-part, header-count and nesting-depth limits at the web server or proxy.
- Using suitable container or cgroup limits to constrain resource consumption.

For **CVE-2026-109619**, published measures include:

- Setting `session.upload_progress.enabled=0` in a configuration that takes effect before request parsing, followed by restarting all relevant workers.
- On PHP 8.5, setting `fatal_error_backtraces=0` to block a specific confirmed path.
- Removing invalid or deprecated per-request configuration settings that may emit diagnostics before initialization.
- Correcting attacker-reachable fatal errors and execution timeouts.

These settings address specific attack paths and must not be treated as a complete mitigation. Configuration behavior varies by PHP deployment model; administrators must verify the effective settings in the actual production runtime.

Detection and Threat Hunting

BGD e-GOV CIRT recommends reviewing web-server, PHP and endpoint telemetry for unusual request patterns and suspicious activity around vulnerable PHP workers.

Investigate:

- Unexpected child processes spawned by web-server or PHP worker processes.

- PHP processes launching shells, interpreters, download utilities or other unexpected executables.
- Unexpected outbound connections from web servers.
- New or modified files in web roots, temporary directories and application directories.
- Unauthorized changes to PHP configuration or application files.
- Access to environment files, configuration files or credentials outside normal application behavior.
- Unexpected database queries, account creation or privilege changes.
- Abnormal CPU or memory usage, worker crashes and repeated request-processing errors.
- Unexpected activity following suspicious requests to public-facing PHP services.

Incident Response

If exploitation is suspected:

- **Contain:** Restrict public access or isolate the affected server where operationally feasible
- **Preserve evidence:** Secure web-server, PHP, authentication, process and network logs before they expire.
- **Investigate:** Review suspicious requests, process trees, outbound connections, file changes and access to application secrets.
- **Protect credentials:** Rotate potentially exposed database credentials, API keys, application secrets and cloud credentials.
- **Eradicate:** Remove unauthorized files, processes, accounts and persistence mechanisms; rebuild the server if its integrity cannot be established.
- **Patch and recover:** Install the fixed PHP runtime, restore trusted application components and validate the server before returning it to service.
- **Monitor:** Increase monitoring for repeated exploitation attempts, lateral movement and renewed access.

Do not assume that installing the patch removes an attacker's existing access or reverses a compromise that occurred before remediation.

Recommendations

BGD e-GOV CIRT recommends that government agencies, financial institutions, telecommunications providers, critical-infrastructure operators, educational institutions and private-sector organizations:

- Inventory all PHP runtimes, including those in containers and managed hosting.
- Identify public-facing systems and prioritize them for immediate patching.

- Coordinate with hosting providers where the organization does not control the underlying runtime.
- Verify patched package status rather than relying solely on a displayed version string, particularly where operating-system distributions backport security fixes.
- Review privileged access and credentials available to web applications.
- Ensure that application servers cannot access unnecessary internal systems.
- Monitor for exploitation attempts and investigate suspicious activity preceding patch installation.
- Maintain tested backups and an incident-response plan for web-server compromise.

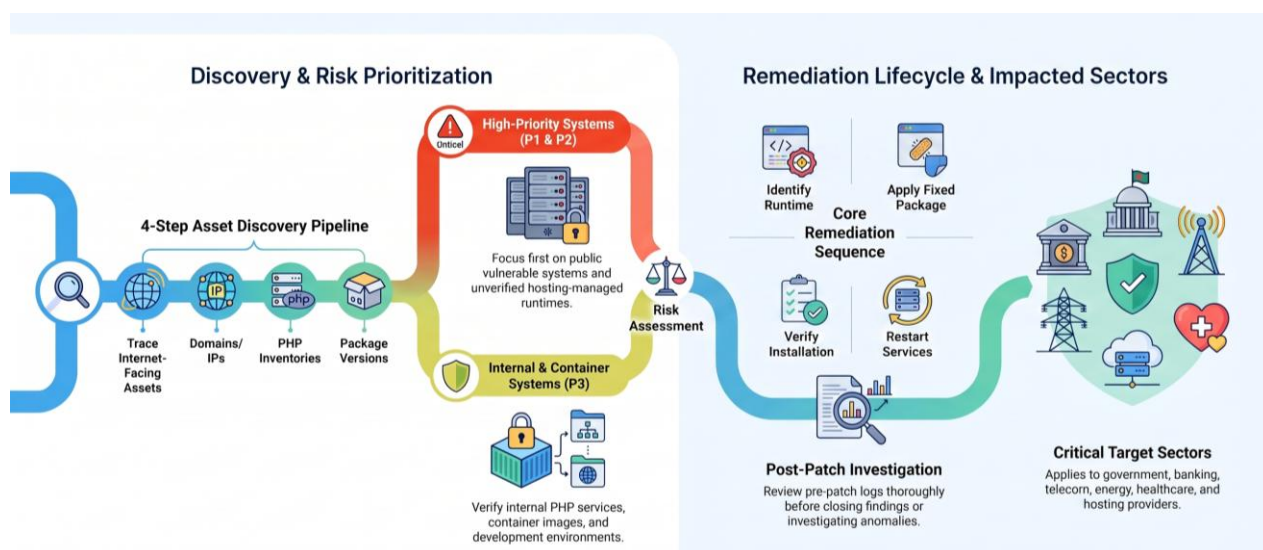


Figure: Proposed national-level PHP exposure assessment and remediation workflow for organizations

The PHP2Shell vulnerabilities represent a critical risk to PHP-based web services because successful exploitation may allow unauthenticated remote code execution. Both vulnerabilities have fixed upstream releases, and organizations should prioritize immediate patching.

Configuration adjustments, request filtering and network restrictions can reduce exposure while updates are prepared, but they do not provide equivalent protection. Organizations should also investigate suspicious activity and rotate credentials if compromise or unauthorized access is suspected.

BGD e-GOV CIRT urges all organizations operating PHP-based services to verify their exposure, apply the appropriate security update immediately and review relevant logs for suspicious activity.

References

- <https://www.php.net/archive/2026.php>
- <https://thehackernews.com/2026/10/php2shell-what-actually-protects-you.html>



BGD e-GOV CIRT