

AISTRIKE · THREAT ADVISORY · CRITICAL

php2shell: unauthenticated RCE in PHP core

CVE-2026-109618 and CVE-2026-109619. Memory-safety flaws in PHP request startup, error handling and session upload progress, leaked before coordinated disclosure.

Prepared 23 October 2026

Severity: Critical

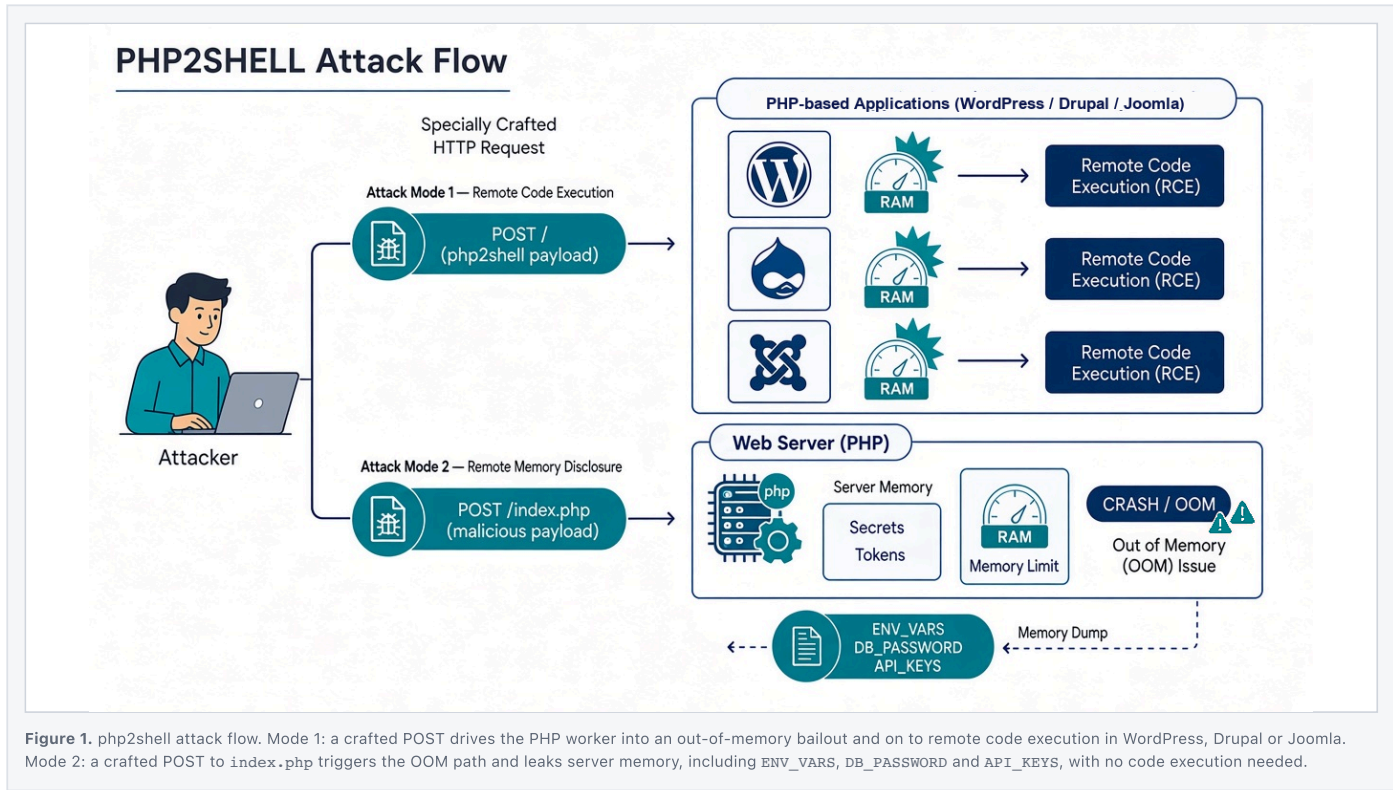
CVSS v4.0 9.2

Platforms: Linux and Windows PHP hosts

Audience: SOC, detection engineering, IR, platform owners

2	9.2	~590	41
Critical CVEs in PHP core (all versions after 5.4)	CVSS v4.0, network vector, no authentication	Requests needed to exploit CVE-2026-109618 in public research	IOCs: 29 IPs, 3 domains, 1 URL, 5 user agents, 4 hashes

- **What happened.** Details of two critical PHP remote code execution flaws leaked on 20 October 2026, before a patch existed. PHP shipped an emergency release on 21 October at 14:00 UTC. watchTowr reproduced CVE-2026-109618 and has not yet seen exploitation in the wild.
- **Why it matters.** The bug is in PHP itself, not one application, so WordPress, Drupal, Joomla, Magento, shared hosting, control panels and vendor appliances with an embedded PHP runtime are all in scope.
- **What to do now.** Patch to 8.5.12, 8.4.27, 8.3.36 or 8.2.35. The PHP team states there are no complete mitigations. Until every host is patched, alert on web-server processes spawning shells and hunt back to 20 October.



Contents

SECTION	PAGE
1. Overview and timeline	3
2. Vulnerability details	3
3. Attack flow and MITRE ATT&CK mapping	4
4. What attackers can do after exploitation	4
5. Detection strategy	4
6. Threat hunting	5
7. Detection rulesets (Sigma, KQL, EQL)	6
8. Indicators of compromise	7
9. Remediation and exposure reduction	7
10. How AiStrike helps	8
11. References	8

FIGURES	PAGE
Figure 1. php2shell attack flow: remote code execution and remote memory disclosure	1
Figure 2. php2shell kill chain with MITRE ATT&CK techniques and detection points	4

TABLES	PAGE
Table 1. CVE-2026-109618 and CVE-2026-109619 side by side	3
Table 2. MITRE ATT&CK mapping for php2shell exploitation and post-exploitation	4
Table 3. Exploit fingerprints in web server error logs	5
Table 4. Post-exploitation detections (Auditd, Sysmon for Linux, EDR, web logs)	5
Table 5. Hunt hypotheses and data sources	5
Table 6. Indicators of compromise	7
Table 7. Fixed PHP builds	7

How to read this advisory. Sections 1 to 4 explain the vulnerability and attacker behaviour. Sections 5 to 8 are the operational content: detections, hunts, ready-to-deploy rules and IOCs. Each detection, hunt and IOC group carries its MITRE ATT&CK technique IDs, which link to attack.mitre.org. Section 10 shows how AiStrike runs this end to end.

AI STRIKE THREAT ADVISORY · PHP2SHELL

The vulnerability

1. Overview and timeline

On 20 October 2026, details of two critical PHP remote code execution flaws, [CVE-2026-109618](#) and [CVE-2026-109619](#), leaked before coordinated disclosure and before any patch. The PHP security team gave [advance notice of an emergency release](#) for 21 October at 14:00 UTC and rated both critical, with a CVSS v4.0 score of 9.2. [watchTowr](#) confirmed the bugs are real, reproduced CVE-2026-109618, and has not seen exploitation in the wild. The nickname in circulation is **php2shell**.

PHP underpins a large share of the web: WordPress, Drupal, Joomla, Magento, shared hosting, control panels, and vendor software that embeds a PHP runtime. A flaw in PHP core, rather than in a single application, multiplies what has to be inventoried.

The **technical root cause is now public**: a pair of memory-safety bugs in PHP core. There is still no reliable end-to-end proof of concept, so defenders cannot simply run an exploit and watch detections fire. The post-exploitation signal is the same as for any other PHP RCE, and that is what this advisory detects.

DATE (2026)	EVENT
20 October	Details of CVE-2026-109618 and CVE-2026-109619 leak before coordinated disclosure. No patch available.
20 to 21 October	PHP security team announces an emergency release. watchTowr confirms both bugs and reproduces CVE-2026-109618. No in-the-wild exploitation observed.
21 October, 14:00 UTC	Emergency release ships fixed builds 8.5.12, 8.4.27, 8.3.36 and 8.2.35. Root cause published on php.net. CISA issues an alert urging immediate patching.
23 October	This advisory: detections, hunts, rulesets and IOCs for AiStrike customers.

The PHP security team is explicit: there are no complete mitigations, and updating is the only safe choice.

2. Vulnerability details

Impact Remote code execution	Attack vector Network, unauthenticated	EPSS High likelihood of exploitation	Affected PHP core after v5.4
--	--	--	--

Table 1. CVE-2026-109618 and CVE-2026-109619 side by side

ATTRIBUTE	CVE-2026-109618	CVE-2026-109619
Severity	CRITICAL	CRITICAL
CVSS v4.0	9.2	9.2
EPSS	High likelihood of exploitation	High likelihood of exploitation
Impact	Remote code execution	Remote code execution
Attack vector	Network	Network
Authentication required	No	No
Vulnerable component	longjmp() for out-of-memory errors and zend_hash_str_add_or_update_i()	longjmp() for out-of-memory errors, fatal_error_backtraces, session upload progress, and other state or error handling components
Affected versions	PHP core after v5.4	PHP core after v5.4
Public exploit	Reproduced by watchTowr; about 590 requests in public research	No reliable end-to-end proof of concept
MITRE ATT&CK	T1190 T1212	T1190 T1212
Advisory	GHSA-w8hw-w9fx-j79w	GHSA-6wmf-4rv5-p3rp

3. Attack flow and MITRE ATT&CK mapping

A remote attacker exploiting a PHP web application wants code execution in the context of the web stack, and that context is a predictable set of processes: `php-fpm` and its workers, `php-cgi`, `lsphp`, or a general web server like `apache2`, `httpd`, or `nginx` driving PHP. Once the attacker has execution there, the next actions are almost always the same regardless of how they got in: confirm access and enumerate, stage through an interpreter, pull second-stage tooling or drop a webshell, decode a staged payload inline, harvest credentials and cloud identity, and establish persistence beyond the shell. Figure 1 on the cover shows the two entry modes. Figure 2 shows the chain that follows and where each stage can be detected.

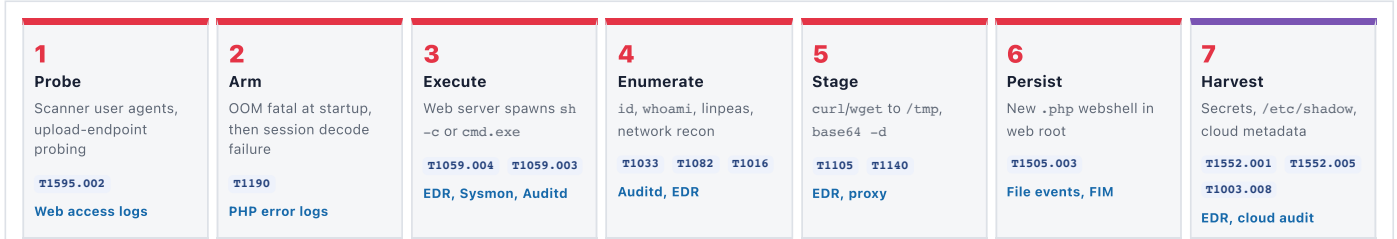


Figure 2. php2shell kill chain with MITRE ATT&CK techniques and the telemetry that sees each stage. Stages 1 and 2 are attempts; stage 3 onwards means the exploit worked.

Table 2. MITRE ATT&CK mapping for php2shell exploitation and post-exploitation

TACTIC	TECHNIQUE	ID	PHP2SHELL BEHAVIOUR
Reconnaissance	Active Scanning: Vulnerability Scanning	T1595.002	Scanners sending <code>php2shell-*</code> user agents and probing upload endpoints
Initial Access	Exploit Public-Facing Application	T1190	Crafted multipart POST with <code>PHP_SESSION_UPLOAD_PROGRESS</code> driving the OOM <code>longjmp()</code> path
Credential Access	Exploitation for Credential Access	T1212	Mode 2 memory disclosure: secrets and tokens of other requests read without RCE
Execution	Command and Scripting Interpreter: Unix Shell / Windows Command Shell / PowerShell	T1059.004 T1059.003 T1059.001	<code>php-fpm</code> , <code>apache2</code> , <code>php-cgi.exe</code> spawning <code>sh -c</code> , <code>cmd.exe</code> , <code>powershell.exe</code>
Discovery	System Owner/User, System Information, Network Configuration Discovery	T1033 T1082 T1016	<code>id</code> , <code>whoami</code> , <code>hostname</code> , <code>ip</code> , <code>ss</code> , <code>netstat</code> , <code>linpeas</code> , <code>pspy</code>
Command and Control	Ingress Tool Transfer; Non-Application Layer Protocol	T1105 T1095	<code>curl/wget</code> to <code>/tmp</code> or <code>/dev/shm</code> ; <code>bash -i >& /dev/tcp/... reverse shells</code>
Defense Evasion	Deobfuscate/Decode Files or Information	T1140	<code>echo ... base64 -d sh</code> inline staged payloads
Persistence	Server Software Component: Web Shell	T1505.003	New <code>.php</code> files written into the web root by the web server user
Credential Access	Unsecured Credentials: Files, Cloud Instance Metadata API; OS Credential Dumping	T1552.001 T1552.005 T1003.008	Reading <code>wp-config.php</code> , <code>.env</code> , <code>/etc/shadow</code> , and the cloud metadata endpoint
Impact	Defacement: External Defacement	T1491.002	Website defacement after server takeover

4. What attackers can do after exploitation

MODE 1 · RCE T1190

Take over the server

Execute code on affected servers without authentication, potentially taking control of the server, accessing sensitive data, moving laterally, or defacing its websites.

MODE 2 · MEMORY READ T1212

Read process memory

Read sensitive PHP process memory, including passwords, authentication tokens, API keys and certificates belonging to other requests or the server itself.

STEALTH · NO RCE NEEDED

Steal data quietly

Steal sensitive data through memory disclosure without first achieving remote code execution. Tools that only watch for malicious code execution will miss this, which is why the error-log fingerprints in section 5 matter.

5. Detection strategy

Why behaviour, not signatures. There is no reliable public proof of concept, and memory-corruption payloads are easy to vary, so a payload signature will age quickly. What does not change is how the exploit stresses PHP (an out-of-memory bailout during request startup) and what an attacker does once the web server is theirs (spawn a shell). AiStrike recommends four detection layers, ordered from weakest to strongest signal:

1. Exposure

An unpatched PHP binary is running. Raises the priority of every other alert on that host.

SIGMA RULE 1 · T1190

2. Attempt

OOM fatals at startup and session decode failures in error logs. Scans or tries, not compromise.

TABLE 3 · T1190

3. Success

A PHP or web-server process spawns a shell. On an unpatched internet-facing host, treat as compromise.

SIGMA RULE 2 · EQL · T1059.004

4. Post-exploitation

Downloads, reverse shells, webshell files, recon and credential access.

TABLE 4 · KQL · T1505.003

Option 1: exploit fingerprints in web server error logs

Look for `PHP Fatal error` (memory exhaustion) or `Failed to decode session object` messages in Apache and Nginx `error.log` files. A single error is common in healthy applications; a burst from one client is the brute-force fingerprint. The `in Unknown on line 0` suffix matters: it means PHP failed during request startup,

before any script ran.

Table 3. Exploit fingerprints in web server error logs

DETECTION	WHAT IT CATCHES	MITRE	SEVERITY
PHP Memory Exhaustion Before Script Execution	A single Allowed memory size of N bytes exhausted ... in Unknown on line 0 fatal. This means an attempt or a scan, not a compromise.	T1190	MEDIUM
PHP Session Decode Failure	The Failed to decode session object. Session has been destroyed warning that follows the memory-exhaustion bailout.	T1190	MEDIUM
PHP Memory Exhaustion Before Script Execution, Burst	Three or more of those fatals from one host within 15 minutes. CVE-2026-109618 exploitation needed about 590 requests in the public research.	T1190 T1595.002	HIGH
PHP Memory Exhaustion Followed By Session Decode Failure	A memory-exhaustion fatal and a session decode failure on the same host within 15 minutes, the <i>arm</i> stage of the exploit.	T1190 T1212	HIGH

Option 2: post-exploitation activity by the web server

Detect post-exploitation commands whose parent is `apache2` or `php-fpm`, or that run as the `www-data` or `apache` user. Also watch for webshell files created in web server directories and other signs of hands-on-keyboard activity. These rules are not `php2shell`-specific, which is the point: they fire whatever entry bug was used.

Table 4. Post-exploitation detections (Auditd, Sysmon for Linux, EDR, web logs)

CATEGORY	SOURCE	RULE	WHAT IT CATCHES	MITRE
Webshell	Auditd	Sigma: Webshell Remote Command Execution	Any program run by the web server user	T1505.003 T1059.004
Hacktool	Auditd	Sigma: Linux HackTool Execution	Known attacker tools: <code>linpeas</code> , <code>pspy</code> , <code>nuclei</code> , <code>ffuf</code> , <code>sliver</code> , <code>hydra</code> ...	T1082 T1057 T1110
Reverse shell	Auditd	Sigma: Suspicious Use of /dev/tcp	<code>bash -i >& /dev/tcp/...-style</code> reverse shells	T1059.004 T1095
Download and execute	Auditd	Sigma: Program Executions in Suspicious Folders	A payload run from <code>/tmp</code> , <code>www</code> or <code>public_html</code> after <code>curl/wget</code> fetches it	T1105 T1204.002
Discovery	Auditd	ESCU: Linux Auditd System Network Configuration Discovery	4+ distinct recon commands (<code>ip</code> , <code>ss</code> , <code>netstat</code> , <code>arp</code> ...) in 15 minutes	T1016 T1049
Webshell	Sysmon for Linux	ESCU: Web or Application Server Spawning a Shell	A web server starting a shell. PHP's <code>system()</code> / <code>exec()</code> always go through <code>sh -c</code> , so this catches almost every webshell command.	T1505.003 T1059.004
Curl / wget	Sysmon for Linux	Sigma: Suspicious Download and Execute Pattern via Curl/Wget	<code>sh -c</code> that downloads into <code>/tmp</code> or <code>/dev/shm</code> , the classic webshell dropper	T1105 T1059.004
Command	EDR	Suspicious Command Execution via Web Server	Recon or credential commands with a web-server parent	T1059 T1033
Child process	EDR	Suspicious Child Execution via Web Server	Shells, LOLBins or interpreters launched by <code>php-fpm</code> , <code>httpd</code> , <code>php-cgi.exe</code>	T1059.004 T1059.003
Rare child	EDR	Unusual Child Execution via Web Server	A child binary never seen before under this web server (baseline deviation)	T1059
File drop	PHP web access logs	Unusual File Creation via Web Server	New <code>.php</code> files in web roots; new ELF binaries in <code>/tmp</code> or <code>/dev/shm</code>	T1505.003 T1105
Script child	PHP web access logs	Web Shell Detection: Script Process Child of Common Web Processes	Script interpreters spawned under common web processes	T1505.003

Tuning. Expected children of PHP include `sendmail`, `ImageMagick` or other image tools, and `node` in some build pipelines. Allow-list them per host and parent command line, not globally, so an attacker cannot hide behind the same binary name.

6. Threat hunting

Scope and window. Hunt every host that runs PHP, not only the known web estate: control panels, monitoring tools and vendor appliances often embed their own PHP runtime. Start the window on **20 October 2026**, the day of the leak, and keep it open until the host is patched. Prioritise internet-facing hosts that also match Sigma rule 1 (unpatched build).

Reading the results. Probe and OOM hits without a shell spawn mean the host was targeted. A shell spawn from a PHP parent on an unpatched host means treat it as compromised: isolate, preserve memory and the web root, then rotate every secret that the PHP process could read, because Mode 2 can leak them without leaving process evidence.

Table 5. Hunt hypotheses and data sources

HUNT	WHAT TO LOOK FOR	DATA SOURCE	MITRE
php2shell probing	Multipart POST requests carrying <code>PHP_SESSION_UPLOAD_PROGRESS</code> , unusually large <code>PHPSESSID</code> cookies, and bursts of HTTP 500 errors on upload endpoints.	Web access logs, WAF, load balancer	T1595.002 T1190
Scanner user agents	User agents containing <code>php2shell</code> (see the IOC list in section 8).	Web access logs, WAF, proxy	T1595.002
Startup OOM bailouts	PHP error log lines with <code>Allowed memory size</code> and <code>exhausted</code> together with <code>Unknown on line 0</code> , which indicate a failure during request startup rather than inside a script.	PHP and web server error logs	T1190

8. Indicators of compromise

Validate before blocking. Scanner IPs rotate, and research honeypots (MadPot) appear in the same data. Block the exploitation and malware infrastructure, monitor MadPot, and do not block the decoy IP. All values are defanged.

Table 6. Indicators of compromise

Exploitation / scanners T1595.002 T1190 · 15 entries 176[.]65[.]132[.]128 146[.]70[.]202[.]189 185[.]5[.]70[.]233 152[.]233[.]51[.]47 77[.]237[.]246[.]73 122[.]97[.]242[.]167 192[.]140[.]154[.]1 47[.]238[.]232[.]123 31[.]182[.]51[.]55 98[.]80[.]11[.]91 200[.]119[.]44[.]50 45[.]136[.]71[.]89 34[.]136[.]122[.]211 146[.]70[.]184[.]216 38[.]207[.]136[.]217	Malware infrastructure T1105 T1071.001 · 6 entries 134[.]199[.]154[.]64 117[.]55[.]202[.]115 92[.]118[.]112[.]100 150[.]251[.]112[.]208 15[.]235[.]206[.]16:8443 45[.]198[.]224[.]169	MadPot T1595.002 · 6 entries 103[.]216[.]220[.]92 31[.]223[.]184[.]166 37[.]19[.]200[.]5 134[.]209[.]159[.]64 103[.]114[.]163[.]188 95[.]217[.]98[.]37 Decoy IP T1595.002 · 1 entry 134[.]154[.]64[.]199	Additional reported IP T1595.002 · 1 entry 221[.]14[.]159[.]134 Domains T1071.001 · 3 entries thunder[.]godofthunder[.]st hackattackkaboom[.]botlesscucks[.]st hik[.]gettingheadfromduckysmom[.]st URL T1105 · 1 entry hxxp://92[.]118[.]112[.]100/nfs
IOAs · HTTP user-agent strings T1595.002 · 5 entries php2shell-scan/109618 php2shell-109618 php2shell-verify/1[.]0 php2shell-test/2[.]0[.]0 Mozilla/5[.]0 (php-session-startup-rce-poc)	SHA-256 hashes T1105 T1505.003 · 4 entries 0c5486f06620e26e316f4b33f318b0d646d5e1133789331aa2fed012360d1486 7133612558a5d0f257ca6bdb4f765453af9465bc5298acdcf33b5761b8c59aac dd98bf48899aa385035352688357742c73ddc5e47ec4ecc7d7b74c53c2e3ca3d 17758cf05e3376f8448dff926564e038b763b4a46668455e2ccb5ac0839e629c		

117[.]55[.]202[.]115, 134[.]199[.]154[.]64 and 146[.]70[.]202[.]189 were also reported separately in the source and are listed once above. 221[.]14[.]159[.]134 was reported without a category, so validate it before blocking.

9. Remediation and exposure reduction

Table 7. Fixed PHP builds

PHP BRANCH	FIXED BUILD	ACTION
8.5	8.5.12	Upgrade now
8.4	8.4.27	Upgrade now
8.3	8.3.36	Upgrade now
8.2	8.2.35	Upgrade now
Older branches after 5.4	No fixed build listed	Affected and end of life. Migrate to a supported branch, or take the host off the internet until you can.

Patch and verify - Inventory every PHP runtime, including containers, control panels and vendor appliances. Sigma rule 1 and EDR software inventory find the ones nobody owns. - Patch internet-facing hosts first, then internal ones. Restart <code>php-fpm</code>, Apache and any long-running PHP workers, because the old binary stays in memory until restart. - Confirm the running version (<code>php -v</code>, <code>phpinfo()</code> on a protected page, or EDR file version), not only the package version.	Until patched, and after - There are no complete mitigations. WAF rules for oversized <code>PHPSESSID</code> cookies, <code>PHP_SESSION_UPLOAD_PROGRESS</code> abuse and <code>php2shell</code> user agents reduce noise but are not a fix. - Run the section 6 hunts back to 20 October. If any host shows a shell spawn or a startup OOM burst, rotate database passwords, API keys, cloud credentials and TLS keys that the PHP process could read. - Remove unknown <code>.php</code> files from web roots and binaries from <code>/tmp</code> and <code>/dev/shm</code>, then rebuild the host from a known-good image if a webshell was found.
--	--

10. How AiStrike helps

php2shell is a good example of why detection has to be stitched together: each signal on its own is weak or noisy, but together they tell a clear story. AiStrike does that stitching and runs the response, so the SOC sees one case per host with the evidence already attached.

INGEST	DETECT	CORRELATE	INVESTIGATE	RESPOND	HUNT AND REPORT
Web, PHP error, EDR, Auditd, Sysmon, WAF and cloud logs	Tables 3 and 4 plus the section 7 rules, mapped to MITRE	Attempt, success and post-exploitation on one host become one case	AI triage adds asset exposure, PHP version, IOC hits, process tree	Playbooks block, isolate, quarantine and open the patch ticket	Retro-hunts to 20 October and MITRE coverage evidence

Detection content, deployed in your SIEM

The error-log fingerprints, the two AiStrike Sigma rules and the KQL and EQL queries are delivered as managed content and translated for Sentinel, Defender, Elastic, Splunk or Datadog, so coverage goes live without each team rewriting rules.

[T1190](#) [T1059.004](#) [T1505.003](#)

Correlation instead of alert noise

A single memory-exhaustion fatal is medium and common. AiStrike groups the burst, the session decode failure, the shell spawn and the webshell write on the same host into one composite case and raises severity as the chain progresses.

[T1190](#) [T1059.004](#) [T1105](#)

AI-led triage with exposure context

Each case is enriched with whether the host is internet-facing, whether its PHP build is unpatched (Sigma rule 1), IOC and user-agent matches, and the parent-child process tree. The analyst gets a verdict and a reason, not a raw event.

[T1595.002](#) [T1033](#)

Threat hunting packs and IOC sweeps

The section 6 hypotheses run as a hunt pack across the estate with the window pinned to 20 October. The IOC set in section 8 is swept across proxy, firewall, DNS and EDR, with the decoy IP excluded and MadPot set to monitor only.

[T1595.002](#) [T1071.001](#)

Automated response playbooks

Block the exploitation and malware infrastructure, isolate the host, kill the spawned shell, quarantine the webshell, open the patch ticket for the asset owner, and trigger secret rotation when the memory-disclosure pattern is seen.

[T1505.003](#) [T1212](#) [T1552.001](#)

Patch tracking and coverage evidence

Hosts flagged by Sigma rule 1 are tracked until the fixed build is running. A MITRE coverage view shows which php2shell techniques you can see, which data sources are missing, and closure evidence for audit.

[T1190](#) [T1059](#)

11. References

- <https://www.php2shell.com/>
- <https://www.cisa.gov/news-events/alerts/2026/10/21/critical-vulnerabilities-php-require-immediate-patching>
- <https://www.elastic.co/security-labs/threat-command/php2shell-php-rce-detection#the-php2shell-post-exploitation-flow:-what-a-compromised-web-server-does-next>
- <https://www.wiz.io/blog/php2shell-php-rce>
- <https://detectionflow.com/blog/php2shell-detection-engineering-strategy/>
- <https://www.php.net/archive/2026.php#2026-10-21-5>
- <https://news-web.php.net/php.announce/509>
- <https://github.com/php/php-src/security/advisories/GHSA-w8hw-w9fx-j79w>
- <https://github.com/php/php-src/security/advisories/GHSA-6wmf-4rv5-p3rp>
- <https://watchtower.com/intelligence/php-cve-2026-109618-faq/>
- <https://gist.github.com/Blevenne/472692937d05145d29d17dc3c6edc2a0>
- https://github.com/SigmaHQ/sigma/blob/master/rules/linux/auditd/syscall/lnx_auditd_web_rce.yml
- https://github.com/SigmaHQ/sigma/blob/master/rules/linux/process_creation/proc_creation_lnx_susp_hkti_execution.yml
- https://github.com/SigmaHQ/sigma/blob/master/rules/linux/builtin/lnx_susp_dev_tcp.yml
- https://github.com/SigmaHQ/sigma/blob/master/rules/linux/auditd/syscall/lnx_auditd_susp_exe_folders.yml
- https://github.com/splunk/security_content/blob/develop/detections/endpoint/linux_auditd_system_network_configuration_discovery.yml
- <https://attack.mitre.org/>