PROJECT INFO

PROJECT TYPE

Cybersecurity Incident Response

ENVIRONMENT

WordPress / DigitalOcean / Cloudflare

DURATION

September 2026

MY ROLE

Incident Response / Security Engineering

TOOLS & PLATFORMS

Cloudflare WAF, DigitalOcean, Postfix, auditd

PROFILE

View on LinkedIn ↗

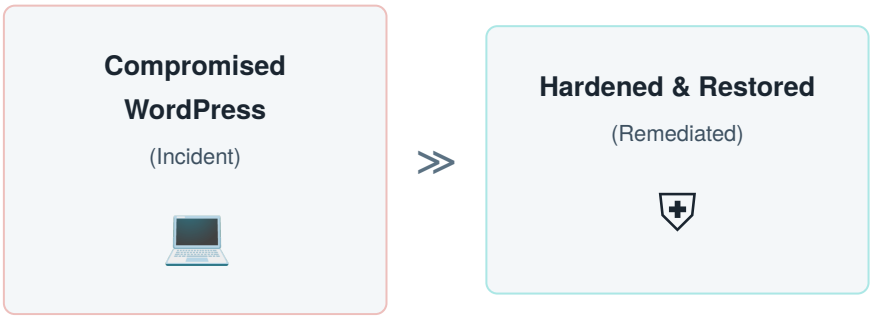
CASE STUDY

WordPress Web Compromise & Incident Response

Web Application Compromise, Malware Persistence, Outbound Spam & Cloud Security Hardening

`Incident Response` `WordPress Security` `Cloudflare WAF`

Investigated and contained a compromise of a production WordPress environment — malicious PHP persistence, abuse of the local mail service for outbound spam, and search-index poisoning — then restored and hardened the environment while keeping strict evidence discipline throughout.



01. Project Overview

My Role

Led triage, forensic file analysis, WAF containment and post-incident hardening.

Environment

Single production WordPress site behind Cloudflare, hosted on a DigitalOcean droplet.

Scope

Malicious file investigation, mail-abuse tracing, WAF hardening, SEO remediation.

Focus

Contain active compromise and reduce attack surface without overstating unproven claims.

Project Objective: Contain active compromise activity, recover a clean environment from a known-good snapshot, and harden Cloudflare/WordPress/infrastructure controls — while explicitly separating confirmed evidence from suspected attack vectors.

02. The Incident

01

Malicious File Persistence

Obfuscated PHP webshells and loader scripts placed under core WordPress paths.

02

Outbound Spam Abuse

Apache's service account used the local Postfix service to send phishing email.

03

Search-Index Poisoning

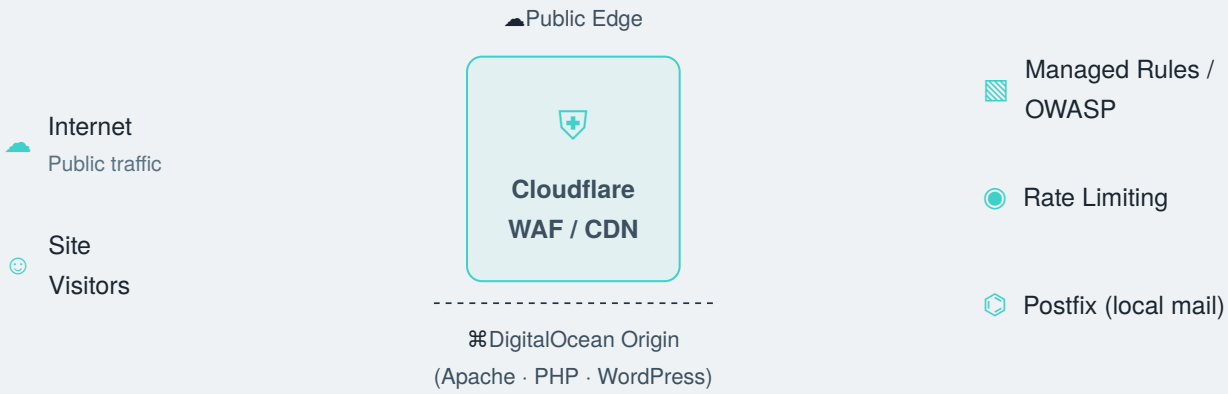
Spam-style content appeared in Google's index for the domain.

04

Unresolved Entry Point

The exact initial attack vector could not be conclusively established.

03. Architecture (High Level)



Architecture Highlights

- Public traffic expected to traverse Cloudflare before reaching the origin
- Origin later restricted to Cloudflare's published IP ranges
- Local Postfix used for outbound mail — a path attackers reused for spam
- Application path: `/var/www/[app-root]/`
- WordPress 6.8.2 on PHP 7.4.3 (end-of-life) at time of compromise

04. Attack Chain & Timeline

Reconstructed from confirmed post-exploitation evidence. The initial-access step is intentionally left unresolved — no single vector was proven.

```
Potential Initial Access  NOT DETERMINED
  ↓
WordPress / Application → Malicious PHP Execution → Apache (UID 48) → Local Postfix → Outbound Spam

PARALLEL — PERSISTENCE
Compromised WordPress → Malicious Files / Webshell → SEO Poisoning
```

DETECTION LAYER

`Cloudflare WAF → LFI / File Inclusion Detection → Managed Challenge`

01 Aug 2026

Known-good DigitalOcean snapshot captured — later used as the recovery baseline.

06 Aug 2026

WordPress discloses CVE-2026-64638, affecting core 6.8.0–6.8.6 (fixed in 6.8.7). The production install was later found on 6.8.2 — inside the affected range.

~02 Sep 2026

Believed start of the compromise window based on surrounding file-modification evidence. UNCONFIRMED

08 Sep 2026

File modifications observed:

- memberfuns.php, t00l.php
- wp-blog-header.php, wp-config.php, php.ini

09 Sep 2026

Malicious PHP files identified under `/wp-admin/js/js/` (`cache.php` , `index.php`), owned by `apache:apache` . CONFIRMED

09 Sep 2026, ~21:53–21:54

Apache (UID 48) observed submitting local mail to Postfix from `support@[client-domain]` — first direct link between the compromised app and outbound spam. CONFIRMED

09 Sep 2026

A suspicious PHP process executed under Apache from within `wp-includes/` ; the invoked script had disappeared by the time of inspection, consistent with delete-after-execution.

10 Sep 2026

Cloudflare logged a request against `admin-ajax.php` attempting to read `wp-config.php` , classified under "Broken Access Control, File Inclusion" and Managed Challenged. Occurred *after* the compromise indicators above — treated as later probing, not initial entry.

05. Detection & Investigation

Malicious file evidence

CONFIRMED

/var/www/[app-root]/wp-includes/ad4b6460/ — an out-of-place directory containing `edit.php`, `doge.gif`, a decoy zero-byte "license" file, and other artifacts.

edit.php used obfuscated PHP: dynamic string construction, `pack("H*")` decoding, and remote-payload/eval behavior consistent with a webshell.

CONFIRMED

/var/www/[app-root]/wp-admin/js/js/cache.php (375 bytes) and **index.php** (423 bytes) — owned by `apache:apache`, modified 09 Sep ~19:28, obfuscated archive-loader style payloads.

> Not every modified file was malicious — `index.php`, `.htaccess`, `php.ini`, and `wp-config.php` also changed and needed individual contextual review, since legitimate files change too.

Post-exploitation mail abuse

The strongest single piece of evidence: Apache's service account submitted mail directly to the local Postfix queue.

```
postfix/pickup ... uid=48 from=<support@[client-domain]> Received: by [origin-hostname].localdomain (Postfix, from userid 48)
```

The queued message was a fraudulent traffic-fine phishing email. Outbound attempts were rejected/blocklisted by Microsoft, Gmail, and FortiGuard.

> **Interpretation:** local Postfix abuse by the compromised application — not evidence that the `support@[client-domain]` mailbox itself was compromised.

Cloudflare security events (7-day window)

RULE	EVENTS
Inbound Anomaly Score Exceeded (949110)	382
WordPress – Broken Access Control, File Inclusion	35
Version Control – Information Disclosure	32

Information Disclosure – Common Files	6
WordPress/Drupal Code Injection / Deserialization	3

STRONG INDICATOR

Notable LFI attempt — source 139.9.35.81 (China, AS55990 Huawei Cloud), GET against:

```
/wp-admin/admin-ajax.php?action=kbsilder_show_image&img=../wp-config.php
```

Managed Challenged by Cloudflare, likely a file-read attempt. This IP is **not** identified as the original attacker — timeline evidence does not support that attribution.

WordPress role review

A custom "Web Designer" role held a disproportionate capability set: 118/130 total capabilities, including full plugin, user, and custom-code access.

> This role did not cause the compromise — it's flagged because excessive privilege increases the potential impact of any future account compromise, and is treated as a hardening finding, not a root cause.

06. Response Approach

☰

Detection

Identified malicious files, execution, and outbound spam via log and file review.

▤

Containment

Deployed WAF rules, restricted origin exposure, verified outbound SMTP blocking.

⚙

Eradication

Restored from the 01 Aug known-good snapshot to remove persistence.

Recovery

Reviewed application state, users/roles, and vulnerable core version.

Hardening

Cloudflare rules, least-privilege review, TLS and patch-gap fixes.

Monitoring

Scoped host-level auditd + centralized logging as the next phase.

07. Technology Stack

Cloudflare
WAF / CDN / DNS

DigitalOcean
Origin Hosting

WordPress
CMS

Apache / PHP
Application Runtime

Postfix
Local Mail

Linux auditd
Recommended Monitoring

08. My Contribution

- ✓ Led incident triage, log analysis and timeline reconstruction
- ✓ Performed malicious file discovery and PHP webshell analysis
- ✓ Investigated the Apache/Postfix mail-abuse process chain
- ✓ Designed and deployed Cloudflare WAF and rate-limit rules
- ✓ Audited WordPress users, roles and privilege assignments
- ✓ Executed snapshot-based infrastructure restoration
- ✓ Remediated SEO poisoning via Search Console
- ✓ Produced the remediation plan and residual-risk assessment



---

09. Outcome & Impact

⛨

Persistence Removed

Malicious PHP eliminated via restoration.

▧

Spam Contained

Outbound spam stopped; TCP/25 egress verified.

◉

WAF Hardened

New rules deployed; XML-RPC abuse blocked.

⌘

Origin Restricted

Direct access limited to Cloudflare IP ranges.

Index Cleaned

SEO poisoning remediated on the live site.

10. Key Takeaways

- → WAF visibility is not equivalent to host-level filesystem visibility
- → A compromised WordPress app can pivot into abusing local services like Postfix
- → Least privilege matters for WordPress custom roles, not just OS accounts
- → A snapshot restore is sound containment but not proof of root cause
- → SEO poisoning can outlive the malicious content that caused it
- → Confirmed evidence and suspected vectors must stay clearly separated



11. Residual Risk & Incident Status

FINAL STATUS

Remediated / Contained

INITIAL ATTACK VECTOR

Not Conclusively Identified

Known post-exploitation activity — malicious PHP execution, mail-service abuse, and SEO poisoning — was investigated and contained, and the environment was restored and hardened. Residual risk remains because initial access was never proven, the site was running a vulnerable WordPress core during the compromise window, several components were outdated, and one custom role held excessive privileges. The exact entry point could not be reconstructed with certainty from the available evidence.