WordPress Vulnerabilities

Gabriel Urbaitis
Bennett Poulin
CS544

What is WordPress?

WordPress is “The open source publishing platform of choice for millions of websites worldwide—from creators and small businesses to enterprises.”

It is popular? “It is also the platform of choice for over 43% of all sites across the web.”

It gives content creators with zero code ability the chance to make websites and online content. “People with a limited tech experience can use it “out of the box”, and more tech-savvy folks can customize it in remarkable ways.”

https://wordpress.org

Why Study WordPress Vulnerabilities?

- ➔ The knowledge transfers to other applications – even phone apps
- ➔ Knowing what kinds of attacks are out there you can be more aware as you do your own online activities
- ➔ Be a better coder

Why is it important?

Many State actors and corporations rely on WordPress:

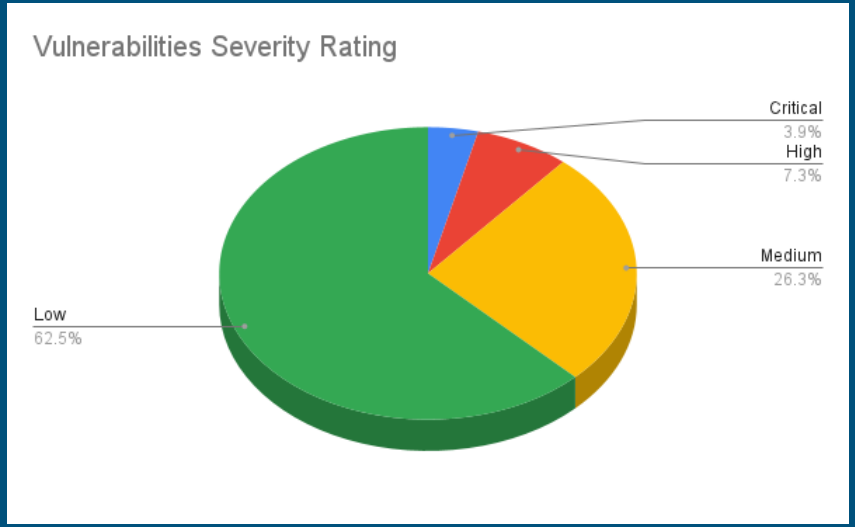
- ➔ https://www.whitehouse.gov/
- ➔ https://www.state.gov/
- ➔ https://www.nasa.gov/
- ➔ https://thewaltdisneycompany.com/
- ➔ https://www.sonymusic.com/

Problem Statement

- WordPress powers over 40% of websites — high-value target
- Vulnerabilities in plugins are a major attack vector
- Goal: Explore real-world exploitability of common WordPress plugin vulnerabilities
- Focus: CSRF, XSS, nonce mismanagement

How BIG is the problem?

- ➔ 9496 vulnerabilities were reported over 2024, averaging to ~183 per week.
- ➔ Over 1000 of these were a Critical or High rating.
- ➔ Each vulnerability can range in number of installs from a few hundred to millions.
- ➔ Patching is slow and up to the plugin maintainers.
- ➔ Core only had 6 vulnerabilities



Statistics compiled from data located at https://solidwp.com

Vulnerabilities by Popularity

1. Cross Site Scripting (XSS)
2. Broken Access Control
3. Cross Site Request Forgery (CSRF)
4. SQL Injection
5. Sensitive Data Exposure
6. Arbitrary File Upload
7. Local File Inclusion
8. PHP Object Injection
9. Privilege Escalation
10. Broken Authentication
11. Server Side Request Forgery (SSRF)
12. Bypass Vulnerability
13. Insecure Direct Object References (IDOR)
14. Remote Code Execution (RCE)
15. Arbitrary File Deletion Backdoor
16. Settings Change
17. Arbitrary File Download
18. Path Traversal
19. Arbitrary Code Execution
20. Open Redirection

Statistics compiled from data located at https://solidwp.com

Method

- Built isolated WordPress instances for safe testing
- Deployed vulnerable versions of selected plugins
- Used manual payload crafting and tools like Burp Suite, DevTools
- Focused on reproducibility and ethical testing

Challenges (Gabe)

- Difficulty sourcing old vulnerable plugin versions
- Complicated CSRF setups requiring valid session and nonce
- WordPress core defenses sometimes blocked naive attacks
- Time-consuming setup across multiple plugin combinations

Challenges (Bennett)

All of the cooler exploits were removed from availability, so in order to analyze them we would have to have been lucky enough to already have them installed. We were not so lucky.

Some modules were written in other languages. Translation softwares went far, but not far enough most of the time.

Each module has its own method of interfacing with site administrators. There wasn’t always a lot of rhyme or reason to how one plugin worked compared to the next.

Some plugins cost money.

teachPress Plugin (CSRF Vulnerability)

- teachPress admin endpoint did not validate nonce or referer
- Crafted manual curl POST to create SQL report
- Attack succeeded without using a browser
- Lesson: Nonce + Referer validation critical for admin pages

Thumbnail carousel slider

- ➔ Plugin Version: Thumbnail carousel slider <= 1.0.4
- ➔ Vulnerability: SQL Injection
- ➔ URL: https://wordpress.org/plugins/wp-responsive-thumbnail-slider/
- ➔ Description: Lack of validation on get parameter ID allowed for direct access to SQL query injection.
- ➔ Severity Score: High

https://www.cve.org/CVERecord?id=CVE-2019-25222

TCS Vulnerable Code:

```
1322        <?php if(isset($_GET['id']) and $_GET['id']>0)
1323            {
1324
1325
1326                $id= $_GET['id'];
1327                $query="SELECT * FROM ".$wpdb->prefix."responsive_thumbnail_slider WHERE id=$id";
1328                $myrow  = $wpdb->get_row($query);
1329
1330                if(is_object($myrow)){
1331
1332                    $title=$myrow->title;
1333                    $image_link=$myrow->custom_link;
1334                    $image_name=$myrow->image_name;
1335
1336                }
1337
1338            ?>
```

https://plugins.trac.wordpress.org/browser/wp-responsive-thumbnail-slider/tags/1.0.4/wp-responsive-images-thumbnail-slider.php#L1326

TCS SQLMap Results

```
sqlmap -u "http://192.168.1.4/wp-admin/admin.php?page=responsive_thumbnail_slider_image_management&action=addedit&id=4" --cookie="<cookie_redacted>" --risk=3 --level=5 --batch
less /home/kali/.local/share/sqlmap/output/192.168.1.4/log
#sqlmap identified the following injection point(s) with a total of 23669 HTTP(s) requests:
#---
#Parameter: id (GET)
#    Type: time-based blind
#    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
#    Payload: page=responsive_thumbnail_slider_image_management&action=addedit&id=4 AND (SELECT 5871 FROM (SELECT(SLEEP(5)))cjTs)
#---
#web server operating system: Linux Fedora
#web application technology: Apache 2.4.63, PHP 8.3.19
#back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
```

GlobalPayments WooCommerce Plugin (Stored XSS for Chaining)

- Global Payments plugin did not sanitize input properly
- Injected <script> into payment fields successfully
- XSS succeeded but failed to chain to CSRF
- Lesson: Stored XSS is serious; chaining attacks needs access to secure tokens (nonces)

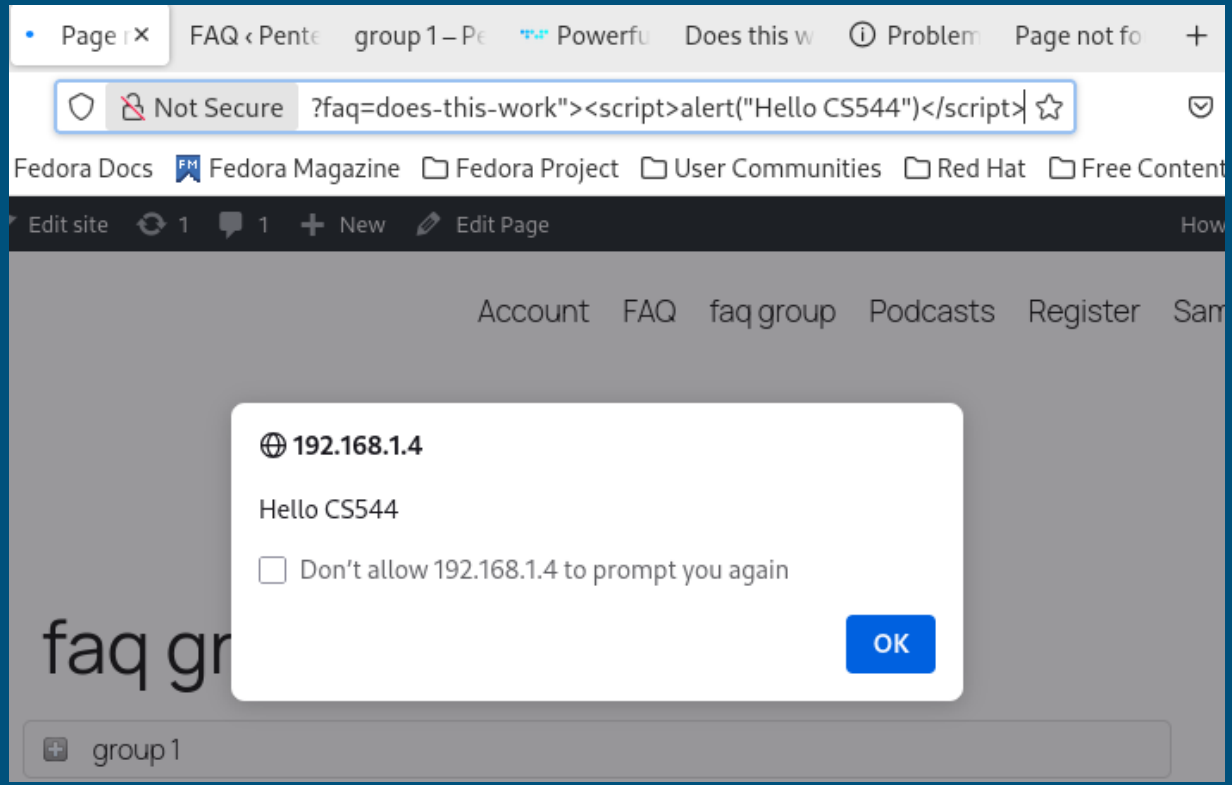
Arconix FAQ

- ➔ Plugin Version: arconix-faq <= 1.9.5
- ➔ Vulnerability: XSS
- ➔ URL: https://wordpress.org/plugins/arconix-faq/
- ➔ Description: Lack of validation on get parameter FAQ allowed for direct JavaScript injection.
- ➔ Severity Score: High

Omitted in report for lack of reproducibility.

https://www.cve.org/CVERecord?id=CVE-2025-32531

Arconix FAQ: XSS



EZ SQL Reports “Save Report” CSRF Attempt

- Vulnerability: Critical Cross-Site Request Forgery (no built-in nonce on action=save).
- Attack Path:

Malicious HTML page to hidden POST to

/wp-admin/admin.php?page=elisqlreports&action=save with SQL-report fields.

- Did It Work?

Browser issued the POST, server replied 302, wp-login.php (not authorized), no report saved.

EZ SQL Reports “Save Report” CSRF Attempt

- What We Learned

Nonce and session check on server and X-Frame-Options: sameorigin stop CSRF.

Need valid nonce or XSS chain for a true “no-click” exploit.

- Outcome

Attack failed - demonstrating real-world CSRF defenses.

Future Work

WooComerce and few others seem to commonly repeat

We may see something we have installed show up with a with something we have installed already for a Broken Access Control, PHP Object Injection, or backdoor

Finding live in the wild vulnerabilities would be fun, but ethics and legality would then come in to play

Conclusion

If you stick with WordPress Core you're probably pretty safe. Especially if you update often.

If you branch out in plugins or themes, vet them first.

Read the documentation.

Stay safe out there.