

WordPress Security

Hunting security bugs in a supermarket

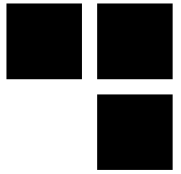


Presented 09th of February, 2017

To Security Day 2017, Lille

By Thomas Chauchefoin





```
() { x;}; echo Content-type :: whoami
```

- Security ninja @Synacktiv
- What we do:
 - Internal / external security assessments
 - Red Team
 - Code review
 - Exploit development
 - Formations
 - Acrobatic juggling





() { x;}; echo Content-type :: groups

- They are too numerous... We need more ninjas!
- Internship positions:
 - Security assessments framework developer
 - 0-days hunter
 - Automated testing on Android applications
- Pentester positions as well
- Ping us at contact@synaktiv.com



WordWhat?

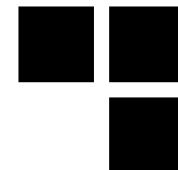
- Content Management System (CMS) by Automatic
- Written in PHP
 - With 5.2 support enforced (EOL: 6 years ago!)
- 179519 lines of code right now (counted by hand)
- Runs 27% of all websites (source: Wikipedia)
 - 53,4 % are not using a CMS
 - Easy to detect (wp-includes, wp-content, ...)

Security of the core

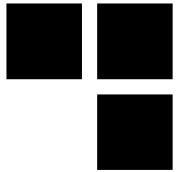


- Auto-updates are enabled if the permissions on the folders are correctly set
 - Leaks PHP version, MySQL version, blogs count, users count...
- Fetches the last release from `api.wordpress.com`
 - You compromise it, you win, nothing's signed, but maybe one day... ([#39309](#), [#25052](#))
 - Maximal mayhem: block future auto-updates
 - Potential RCE on this host was silently patched: "Add support [...] documentation."

Security of the core



- “Content spoofing” in REST API (< 4.7.2)
 - “As part of a vulnerability research project [...] on WordPress, **we discovered was a severe content injection (privilege escalation) vulnerability** affecting the REST API.”
 - “We disclosed the vulnerability to the WordPress Security Team **who handled it extremely well.** They worked closely [...] security providers aware and **patched before this became public.**”
 - “**A fix for this was silently included** on version 4.7.2 along with other less severe issues.”



Security of the core

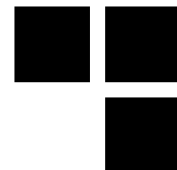
- MySQL's utf8 \neq utf8mb4
- Without the strict mode, it'll truncate the value before insertion...
- ...but your server-side check will be performed on the whole string
- Insert two comments to form a new tag:
 - `<q cite='xx 🍌`
 - `' onmouseover='... '>`
- 14 months to fix the vulnerability (4.1.2)



Extending WordPress

- Core can be extended with themes and plugins
- More than 48k plugins, manually reviewed (??)
- Some statistics for each target plugin
 - Active installs: 100k+, 200k+, 2M+...
 - Download history with real statistics
 - Active versions repartition
- WordPress <3 monorepos:
 - <https://plugins.svn.wordpress.org/>
 - 1.6M ~ revisions and counting, you can't just clone it

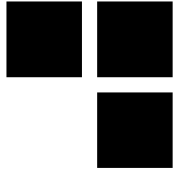
So what?



■ The facts

- More than one million source code files
- Written in PHP, with 5.2 support in mind
- Mostly developed by individuals, small agencies
- They ~~will~~ can do things wrong, grep it!





A10: Open redirects

- `wp_redirect()` VS `wp_safe_redirect()`
 - Host checking
 - Always prevents response splitting
 - Works with `data://`, for all your phishing fantasies
- Mostly useful when chained with other vulnerabilities
- Not always vulnerable, more especially when getting prefixed
 - `get_bloginfo('url')`
- `exit()` and `let die()`



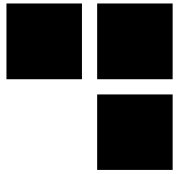
A09: Vulnerable components

- PHPMailer
 - 84 occurrences of the class in all the plugins
 - Not directly exploitable
 - Already bundled by WordPress
- php-jwt
 - 5 occurrences of the class
- Core dependencies are not handled with composer

A08: Cross-Site Request Forgeries

- Per-request nonces
 - Not one-time use (even if it's called a nonce)
 - Tied to one user, action, session, window of times
 - Depends of `NONCE_SALT`, `NONCE_KEY`
 - `wp_nonce_field()`, `wp_verify_nonce()`
- Check the referrer too!
- Hard to grep for, need a better idea

A07: Missing Function Level Access Control



- What's the purpose of `is_admin()`?
- What's the purpose of `is_user_admin()`?
- What's the purpose of `is_super_admin()`?
- `current_user_can(cap1, cap2...)`
- AJAX endpoints are often missed:
 - Call it at `/wp-admin/admin-ajax.php?action=`
 - `wp_ajax_*` / `wp_ajax_nopriv_*`
 - `add_action()`

A06: Sensitive data exposure



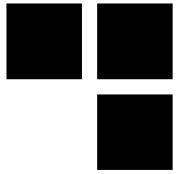
- A lot of administrative plugins are “doing the things wrong. Sad!”.
 - Wrong permissions / extensions on the files
 - Predictable paths / names
 - LFI / AFD
- Directory listing on the download folder may help
- Be restrictive with your exotic parsers

A05: Security misconfiguration



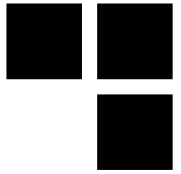
- “put your unique phrase here”
 - It may call <https://api.wordpress.org/secret-key/1.1/salt/>—not funny.
 - CA bundle: ## Includes a WordPress Modification - We include the 'legacy' 1024bit certificates for backward compatibility. See <https://core.trac.wordpress.org/ticket/34935#comment:10> Wed Sep 16 08:58:11 2015
 - Still includes *WoSign* and *Startcom*, now removed from Mozilla’s list
- Bake smelly authentication cookies

A05: Security misconfiguration

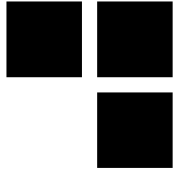


- But wait, there is a plugin for it!!!
- “Salt Shaker enhances WordPress security by changing WordPress security keys and salts manually and automatically.”
- It’s just using `file_get_contents` on the API
 - > PHP5.6: “All encrypted client streams now enable peer verification by default.”
- It’ll also create a `wp-config.php.tmp` :^)

A05: Security misconfiguration

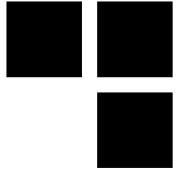


- A lot of HTTP calls, everywhere
 - Credits
 - Importers plugins
 - Browser needs update?
- Others are HTTPS, “if supported”
- The WordPress development team made assumptions like
 - Your usernames are public, so their enumeration is OK
 - Full path disclosures are a configuration issue, don't you run your instance on a dedicated server?



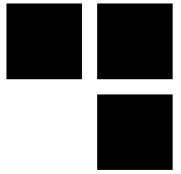
A04: Direct Object Reference

- Don't circumvent core mechanisms
 - `get_post()`
 - `get_user_data()`
 - ...



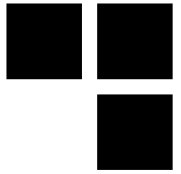
A03: Cross-Site Scripting

- It's a problem of output encoding, not of sanitization
- Don't forget the context:
 - JavaScript code,
 - HTML attribute,
 - Inline content,
 - etc.



A03: Cross-Site Scripting

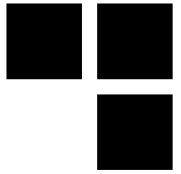
- It's a problem of validation and output encoding
- `sanitize_*()` functions family
- Don't forget the context
 - JavaScript code: `esc_js()`,
 - HTML attribute: `esc_attr()`,
 - Inline content: `esc_html()`,
 - etc.



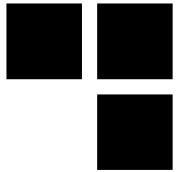
A03: Cross-Site Scripting

- Sounds lame but It'll easily lead to server compromise
- You can bypass nonces and edit files
 - Make a request via XHR,
 - Extract `_wpnonce`, `_wp_http_referer`,
 - Send the `action=update` request to `/wp-admin/theme-editor.php`.
- You can also install a malicious plugins, if the editor is disabled

A02: Broken Authentication and Session Management



- Hashes are stored in the PHPass format
 - 14000 hashes/s ~ on my laptop
 - Future-proof?
- Everything can be overloaded by plugins, authentication too
- Cool target functions
 - `wp_set_auth_cookie()`
 - `wp_login()`
 - `wp_signon()`



A01: Injection

- You name it, SQL injections
- Core functions *should* be safe
 - CVE-2017-5611, “Ensure that queries work correctly with post type names with special characters”. Yep, that was silently patched too.
- People will still misuse \$wpdb
 - Common miscomprehension of prepared statements
 - Or even mysql_*()!



A01: Injection

- PHP Object Injections are in da place too
- Serialization: creating a string representation of the state of the instance of an object
- unserialize(), maybe_unserialize()
- Forget class whitelisting, thanks PHP 5.2
- It much more common than you may think



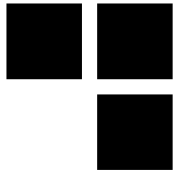
A01: Injection

- Crafting a popchain

- Find an entrypoint

- `__wakeup()`
 - `__destruct()`
 - `__toString()`
 - `__call()`
 - `__set()`
 - `__get()`

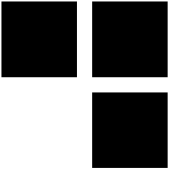
- No autoloader in Wordpress, but put a breakpoint and list available classes and methods



A01: Injection

- Crafting a popchain
 - Define an objective
 - Read the configuration file?
 - Delete a file?
 - Execute code or commands?
 - Identify the needed function, depending of the objective
 - Find a path between two!
 - A popchain was presented by Sam Thomas in 2015, abusing translations

A01: Injection



- translations.php

```
function make_plural_form_function($nplurals, $expression) {  
    $expression = str_replace('n', '$n', $expression);  
    $func_body = "  
        \$index = (int)($expression);  
        return (\$index < $nplurals)?  
        \$index : $nplurals - 1;";  
    return create_function('$n', $func_body);  
}
```



A01: Injection

- Craft the right PO file

- msgid ""

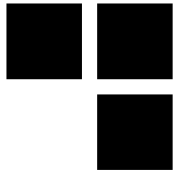
- msgstr ""

- "Content-Type: text/plain; charset=UTF-8\n"

- "Plural-Forms: nplurals = 2; plural =
die(eval(\$_GET['x']));"

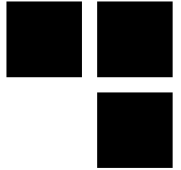
- When unserializing a WP_Theme object, you can force it to fetch a .mo file over the network

- Not all schemes are supported due to is_readable(), but FTP is



Conclusion

- Huge attack surface—don't miss that!
- Monitor new commits on the core for juicy 1days
- Automate everything
 - Reporting is the less fun part
- Audit private plugins?
- Do bug bounties :-)
 - pluginvulnerabilities.com (if > 100k+ active installs)
 - HackerOne, Bugcrowd... you name it



THANKS FOR YOUR ATTENTION!

