Vulnerabilities in Wordpress

Project By: Mariano Pache, Kyle Martin, Adian Mikulic, Sharith Godamanna, Vivek Cherian

Problem Statement

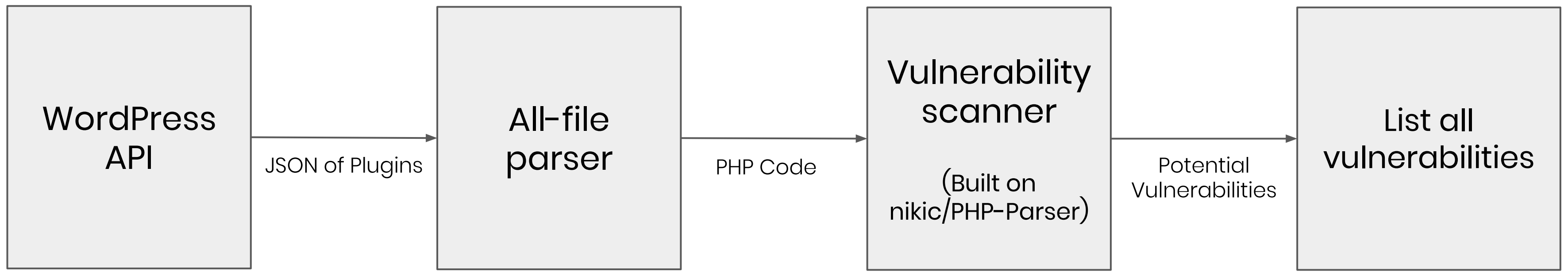
70% of Wordpress Sites Vulnerable

- A study[1] by *WP White Security* found that out of 42,106 Wordpress sites analyzed 73% have a vulnerability.
- This is mostly due to running outdated versions of Wordpress, WP plugins, etc.
- 98% of WordPress vulnerabilities are related to plugins
- Over 455 million websites (>35%) run WordPress

Project Summary

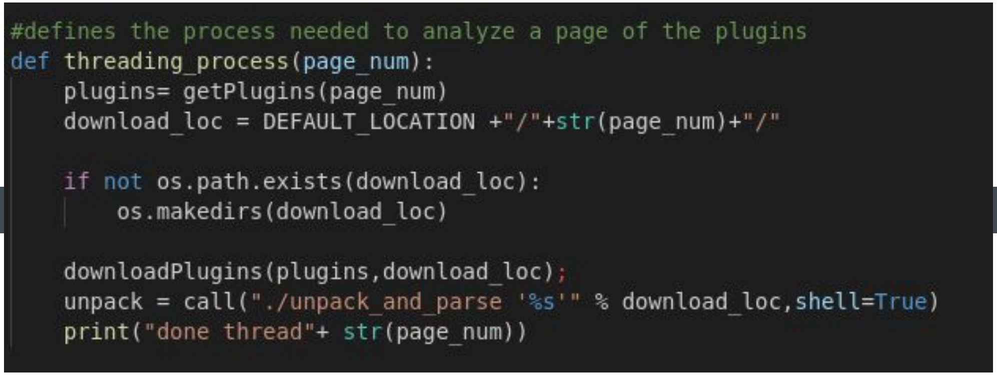
- Researched and determined the most common WordPress vulnerabilities
- Developed a PHP scanner that checks against SQL injection and cross site scripting vulnerabilities
- Listed potentially vulnerable PHP files with line numbers
- Analyzed results

Block Diagram



WordPress API & All-File Parser

- Plugins retrieved using Wordpress API (name, ***download link*****,** etc)
- Plugins downloaded/processed using thread pool to increase throughput
- Each thread downloads one ‘page’ of plugins
- After downloading zips, are extracted (only php files)
- Call to php parser code



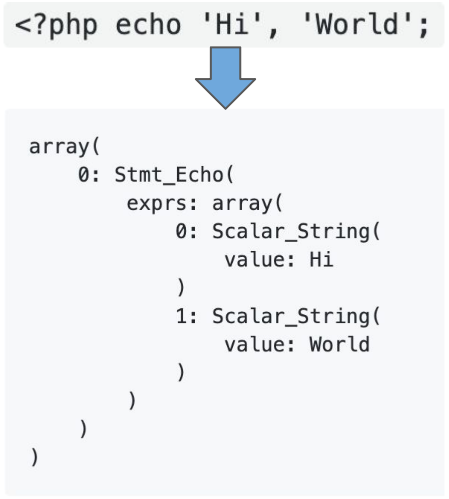
```
#defines the process needed to analyze a page of the plugins
def threading_process(page_num):
    plugins= getPlugins(page_num)
    download_loc = DEFAULT_LOCATION +"/"+str(page_num)+"/"

    if not os.path.exists(download_loc):
        os.makedirs(download_loc)

    downloadPlugins(plugins,download_loc);
    unpack = call("./unpack_and_parse '%s'" % download_loc,shell=True)
    print("done thread"+ str(page_num))
```

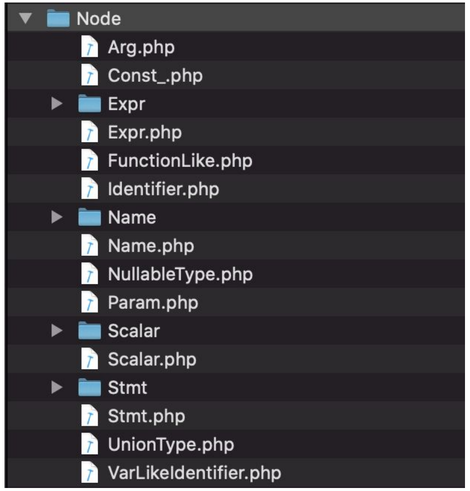
Scanner: Overview

- Converts all php files in a given directory to abstract syntax trees using nikic/PHP-Parser
- Searches AST for vulnerable features
- Outputs vulnerabilities on the command line



```
<?php echo 'Hi', 'World';
```

```
array(
    0: Stmt_Echo(
        exprs: array(
            0: Scalar_String(
                value: Hi
            )
            1: Scalar_String(
                value: World
            )
        )
    )
)
```



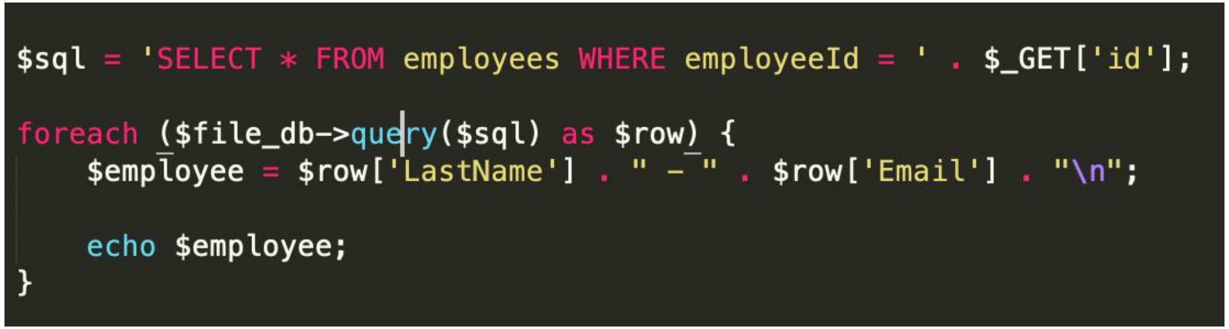
```
▼ Node
    Arg.php
    Const_.php
  ▶ Expr
    Expr.php
    FunctionLike.php
    Identifier.php
  ▶ Name
    Name.php
    NullableType.php
    Param.php
  ▶ Scalar
    Scalar.php
  ▶ Stmt
    Stmt.php
    UnionType.php
    VarLikeIdentifier.php
```

Scanner: SQL Injections

- Scans all php files starting from the root folder of the plugin
- Class extension of NodeVisitor that checks for SQL queries formed by concatenation
- If no args are passed to the constructor, the Class looks for the query function
- Else, it looks for the query strings
- If found, logs file name and line number

Scanner: SQL Injections

Vulnerable code:



```
$sql = 'SELECT * FROM employees WHERE employeeId = ' . $_GET['id'];

foreach ($file_db->query($sql) as $row) {
    $employee = $row['LastName'] . " - " . $row['Email'] . "\n";

    echo $employee;
}
```

Prevention methods:

Prepared statements:

```
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname, $email);

...

$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
$stmt->execute();
```

DEMO
SQL Injection Detection

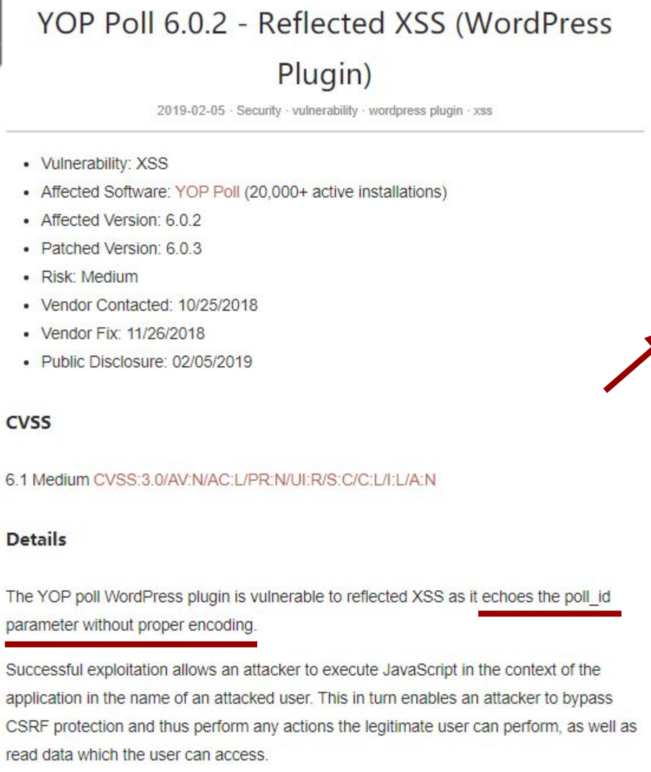
Scanner: Cross Site Scripting

- We look for combinations of dangerous "sinks" and "sources"
- Sinks (i.e. functions)
  - echo, die, print, printf, print_r, var_dump
- Sources (i.e. parameters)
  - $_REQUEST, $_GET, $_POST
- Any instance of above sources meeting above sinks => potential XSS vulnerability

Scanner: Cross Site Scripting

- Traverses AST, anchoring at nodes corresponding to sinks
- Recursively goes down the node to search for a listed source
- Combination found => line is flagged with a XSS warning

Cross Site Scripting Example



YOP Poll 6.0.2 - Reflected XSS (WordPress Plugin)

2019-02-05 · Security · vulnerability · wordpress plugin · xss

- Vulnerability: XSS
- Affected Software: YOP Poll (20,000+ active installations)
- Affected Version: 6.0.2
- Patched Version: 6.0.3
- Risk: Medium
- Vendor Contacted: 10/25/2018
- Vendor Fix: 11/26/2018
- Public Disclosure: 02/05/2019

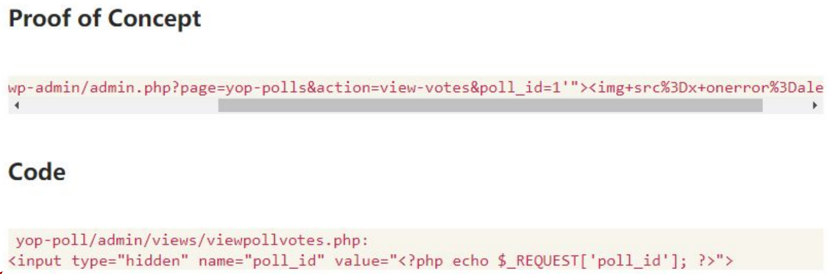
CVSS

6.1 Medium CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N

Details

The YOP poll WordPress plugin is vulnerable to reflected XSS as it echoes the poll_id parameter without proper encoding.

Successful exploitation allows an attacker to execute JavaScript in the context of the application in the name of an attacked user. This in turn enables an attacker to bypass CSRF protection and thus perform any actions the legitimate user can perform, as well as read data which the user can access.

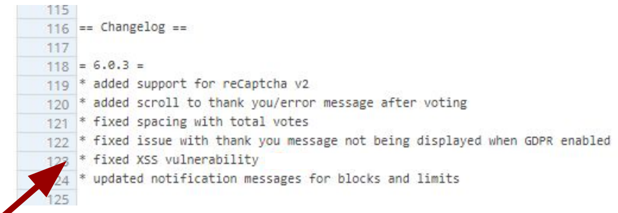


Proof of Concept

```
wp-admin/admin.php?page=yop-polls&action=view-votes&poll_id=1'"><img+src%3Dx+onerror%3Dale
```

Code

```
 yop-poll/admin/views/viewpollvotes.php:
<input type="hidden" name="poll_id" value="<?php echo $_REQUEST['poll_id']; ?>">
```



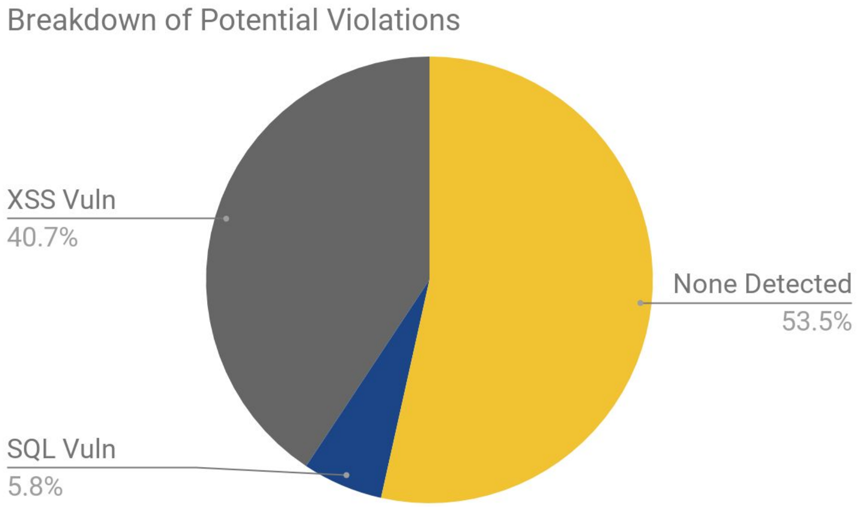
```
== Changelog ==

= 6.0.3 =
* added support for reCaptcha v2
* added scroll to thank you/error message after voting
* fixed spacing with total votes
* fixed issue with thank you message not being displayed when GDPR enabled
* fixed XSS vulnerability
* updated notification messages for blocks and limits
```

DEMO
XSS Vulnerability Detection

Results

Plugins Scanned	Potential SQL Vulnerabilities	Potential XSS Vulnerabilities
50,100 (Whole DB)	2,921	20,386



What we Learned

- Organizing php source code from a website can be very efficiently done with an Abstract Syntax Tree
- SQL injection vulnerabilities often occur when being lazy and concatenating, instead of using PHP's built in sanitizing tools
- XSS vulnerabilities are difficult to detect, but can be prevented with tools like Anti-Samy
- If you're downloading >50,000 WordPress plugins, do so with a 16-core server and 32GB of RAM ;-)

Future Improvements

- Eliminate false positives by checking for sanitization
- Expand SQL detection to pick up many types of query functions
- Expand the list of sinks and sources for XSS detection

Thanks!

Questions?

References

- PHP Parser: https://github.com/nikic/PHP-Parser
- WP API: http://api.wordpress.org/plugins/info/1.1/
- YOP Poll: https://plugins.trac.wordpress.org/browser/yop-poll?order=name
- https://wordpress.org/support/article/brute-force-attacks/
- https://www.exploit-db.com/exploits/44340
- http://dd32.id.au/projects/wordpressorg-plugin-information-api-docs/
- https://wpvulndb.com
- https://www.vice.com/en_us/article/wnjwb4/the-myspace-worm-that-changed-the-internet-forever
- https://www.vice.com/en_us/article/d73j8x/hackers-could-take-over-wordpress-blogs-with-a-single-comment
- http://securityaffairs.co/wordpress/36528/hacking/anonymous-breached-wto-db.html
- https://www.darkreading.com/attacks-and-breaches/fbi-blames-federal-hacks-on-anonymous-campaign/d/d-id/1112650
- https://api.wordpress.org/plugins/info/1.0/
- Icons: [1],[2],[3],[4],[5]