

Enumeration

WordPress Core Version Enumeration

It is always important to know what type of application we are working with. An essential part of the enumeration phase is uncovering the software version number. This is helpful when searching for common misconfigurations such as default passwords that may be set for certain versions of an application and searching for known vulnerabilities for a particular version number. We can use a variety of methods to discover the version number manually. The first and easiest step is reviewing the page source code. We can do this by right-clicking anywhere on the current page and selecting "View page source" from the menu or using the keyboard shortcut `[CTRL + U]`.

We can search for the `meta generator` tag using the shortcut `[CTRL + F]` in the browser or use `cURL` along with `grep` from the command line to filter for this information.

WP Version - Source Code

Code: html

```
...SNIP...
<link rel='https://api.w.org/'
href='http://blog.inlanefreight.com/index.php/wp-json/' />
<link rel="EditURI" type="application/rsd+xml" title="RSD"
href="http://blog.inlanefreight.com/xmlrpc.php?rsd" />
<link rel="wlwmanifest" type="application/wlwmanifest+xml"
href="http://blog.inlanefreight.com/wp-includes/wlwmanifest.xml" />
<meta name="generator" content="WordPress 5.3.3" />
...SNIP...
```

```
mayala@htb[/htb] $ curl -s -X GET http://blog.inlanefreight.com | grep '<meta
name="generator"' <meta name="generator" content="WordPress 5.3.3" />
```

Aside from version information, the source code may also contain comments that may be useful. Links to CSS (style sheets) and JS (JavaScript) can also provide hints about the version number.

WP Version - CSS

Code: html

```
...SNIP...
<link rel='stylesheet' id='bootstrap-css'
href='http://blog.inlanefreight.com/wp-
content/themes/ben_theme/css/bootstrap.css?ver=5.3.3' type='text/css'
media='all' />
<link rel='stylesheet' id='transportex-style-css'
href='http://blog.inlanefreight.com/wp-content/themes/ben_theme/style.css?
ver=5.3.3' type='text/css' media='all' />
<link rel='stylesheet' id='transportex_color-css'
href='http://blog.inlanefreight.com/wp-
content/themes/ben_theme/css/colors/default.css?ver=5.3.3' type='text/css'
media='all' />
<link rel='stylesheet' id='smartmenus-css'
href='http://blog.inlanefreight.com/wp-
content/themes/ben_theme/css/jquery.smartmenus.bootstrap.css?ver=5.3.3'
type='text/css' media='all' />
...SNIP...
```

WP Version - JS

Code: html

```
...SNIP...
<script type='text/javascript' src='http://blog.inlanefreight.com/wp-
includes/js/jquery/jquery.js?ver=1.12.4-wp'></script>
<script type='text/javascript' src='http://blog.inlanefreight.com/wp-
includes/js/jquery/jquery-migrate.min.js?ver=1.4.1'></script>
<script type='text/javascript' src='http://blog.inlanefreight.com/wp-
content/plugins/mail-masta/lib/subscriber.js?ver=5.3.3'></script>
<script type='text/javascript' src='http://blog.inlanefreight.com/wp-
content/plugins/mail-masta/lib/jquery.validationEngine-en.js?ver=5.3.3'>
</script>
<script type='text/javascript' src='http://blog.inlanefreight.com/wp-
content/plugins/mail-masta/lib/jquery.validationEngine.js?ver=5.3.3'>
</script>
...SNIP...
```

In older WordPress versions, another source for uncovering version information is the `readme.html` file in WordPress's root directory.

Plugins and Themes Enumeration

We can also find information about the installed plugins by reviewing the source code manually by inspecting the page source or filtering for the information using `cURL` and other command-line utilities.

Plugins

```
mayala@htb[/htb] $ curl -s -X GET http://blog.inlanefreight.com | sed
's/href=/\n/g' | sed 's/src=/\n/g' | grep 'wp-content/plugins/*' | cut -d'"'"' -f2
http://blog.inlanefreight.com/wp-content/plugins/wp-google-places-review-
slider/public/css/wprev-public_combine.css?ver=6.1
http://blog.inlanefreight.com/wp-content/plugins/mail-masta/lib/subscriber.js?
ver=5.3.3 http://blog.inlanefreight.com/wp-content/plugins/mail-
masta/lib/jquery.validationEngine-en.js?ver=5.3.3
http://blog.inlanefreight.com/wp-content/plugins/mail-
masta/lib/jquery.validationEngine.js?ver=5.3.3 http://blog.inlanefreight.com/wp-
content/plugins/wp-google-places-review-slider/public/js/wprev-public-com-
min.js?ver=6.1 http://blog.inlanefreight.com/wp-content/plugins/mail-
masta/lib/css/mm_frontend.css?ver=5.3.3
```

Themes

```
mayala@htb[/htb] $ curl -s -X GET http://blog.inlanefreight.com | sed
's/href=/\n/g' | sed 's/src=/\n/g' | grep 'themes' | cut -d'"'"' -f2
http://blog.inlanefreight.com/wp-content/themes/ben_theme/css/bootstrap.css?
ver=5.3.3 http://blog.inlanefreight.com/wp-content/themes/ben_theme/style.css?
ver=5.3.3 http://blog.inlanefreight.com/wp-
content/themes/ben_theme/css/colors/default.css?ver=5.3.3
http://blog.inlanefreight.com/wp-
content/themes/ben_theme/css/jquery.smartmenus.bootstrap.css?ver=5.3.3
http://blog.inlanefreight.com/wp-content/themes/ben_theme/css/owl.carousel.css?
ver=5.3.3 http://blog.inlanefreight.com/wp-
content/themes/ben_theme/css/owl.transitions.css?ver=5.3.3
http://blog.inlanefreight.com/wp-content/themes/ben_theme/css/font-awesome.css?
ver=5.3.3 http://blog.inlanefreight.com/wp-
content/themes/ben_theme/css/animate.css?ver=5.3.3
http://blog.inlanefreight.com/wp-content/themes/ben_theme/css/magnific-
popup.css?ver=5.3.3 http://blog.inlanefreight.com/wp-
```

```
content/themes/ben_theme/css/bootstrap-progressbar.min.css?ver=5.3.3
http://blog.inlanefreight.com/wp-content/themes/ben_theme/js/navigation.js?
ver=5.3.3 http://blog.inlanefreight.com/wp-
content/themes/ben_theme/js/bootstrap.min.js?ver=5.3.3
http://blog.inlanefreight.com/wp-
content/themes/ben_theme/js/jquery.smartmenus.js?ver=5.3.3
http://blog.inlanefreight.com/wp-
content/themes/ben_theme/js/jquery.smartmenus.bootstrap.js?ver=5.3.3
http://blog.inlanefreight.com/wp-
content/themes/ben_theme/js/owl.carousel.min.js?ver=5.3.3 background:
url("http://blog.inlanefreight.com/wp-
content/themes/ben_theme/images/breadcrumb-back.jpg") #50b9ce;
```

The response headers may also contain version numbers for specific plugins.

However, not all installed plugins and themes can be discovered passively. In this case, we have to send requests to the server actively to enumerate them. We can do this by sending a GET request that points to a directory or file that may exist on the server. If the directory or file does exist, we will either gain access to the directory or file or will receive a redirect response from the webserver, indicating that the content does exist. However, we do not have direct access to it.

Plugins Active Enumeration

```
mayala@htb[/htb] $ curl -I -X GET http://blog.inlanefreight.com/wp-
content/plugins/mail-masta HTTP/1.1 301 Moved Permanently Date: Wed, 13 May 2020
20:08:23 GMT Server: Apache/2.4.29 (Ubuntu) Location:
http://blog.inlanefreight.com/wp-content/plugins/mail-masta/ Content-Length: 356
Content-Type: text/html; charset=iso-8859-1
```

If the content does not exist, we will receive a 404 Not Found error.

```
mayala@htb[/htb] $ curl -I -X GET http://blog.inlanefreight.com/wp-
content/plugins/someplugin HTTP/1.1 404 Not Found Date: Wed, 13 May 2020
20:08:18 GMT Server: Apache/2.4.29 (Ubuntu) Expires: Wed, 11 Jan 1984 05:00:00
GMT Cache-Control: no-cache, must-revalidate, max-age=0 Link:
<http://blog.inlanefreight.com/index.php/wp-json/>; rel="https://api.w.org/"
Transfer-Encoding: chunked Content-Type: text/html; charset=UTF-8
```

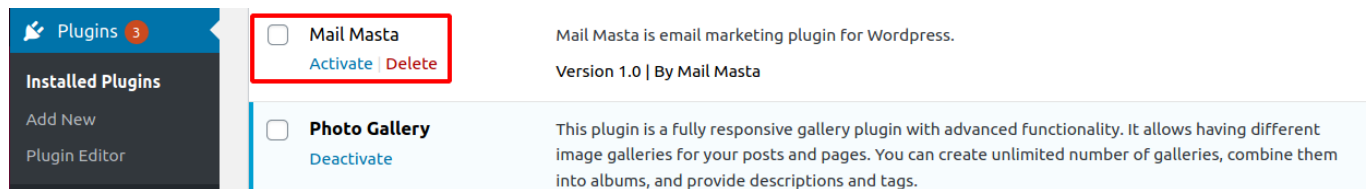
The same applies to installed themes.

To speed up enumeration, we could also write a simple bash script or use a tool such as `wfuzz` or `WPScan`, which automate the process.

Directory Indexing







Active plugins should not be our only area of focus when assessing a WordPress website. Even if a plugin is deactivated, it may still be accessible, and therefore we can gain access to its associated scripts and functions. Deactivating a vulnerable plugin does not improve the WordPress site's security. It is best practice to either remove or keep up-to-date any unused plugins.

The following example shows a disabled plugin.



If we browse to the plugins directory, we can see that we still have access to the `Mail Masta` plugin.

Index of /wp-content/plugins/mail-masta

Name	Last modified	Size	Description
 Parent Directory		-	
 amazon_api/	2020-05-13 18:01	-	
 inc/	2020-05-13 18:01	-	
 lib/	2020-05-13 18:01	-	
 plugin-interface.php	2020-05-13 18:01	88K	
 readme.txt	2020-05-13 18:01	2.2K	

Apache/2.4.29 (Ubuntu) Server at blog.inlanefreight.com Port 80

We can also view the directory listing using `cURL` and convert the HTML output to a nice readable format using `html2text`.

```
mayala@htb[htb] $ curl -s -X GET http://blog.inlanefreight.com/wp-
content/plugins/mail-masta/ | html2text ***** Index of /wp-
content/plugins/mail-masta ***** [[IC0]] Name Last_modified Size Description
=====
[[PARENTDIR]] Parent_Directory - [[DIR]] amazon_api/ 2020-05-13 18:01 - [[DIR]]
inc/ 2020-05-13 18:01 - [[DIR]] lib/ 2020-05-13 18:01 - [[ ]] plugin-
interface.php 2020-05-13 18:01 88K [[TXT]] readme.txt 2020-05-13 18:01 2.2K
```

```
=====
Apache/2.4.29 (Ubuntu) Server at blog.inlanefreight.com Port 80
```

This type of access is called `Directory Indexing`. It allows us to navigate the folder and access files that may contain sensitive information or vulnerable code. It is best practice to disable directory indexing on web servers so a potential attacker cannot gain direct access to any files or folders other than those necessary for the website to function properly.

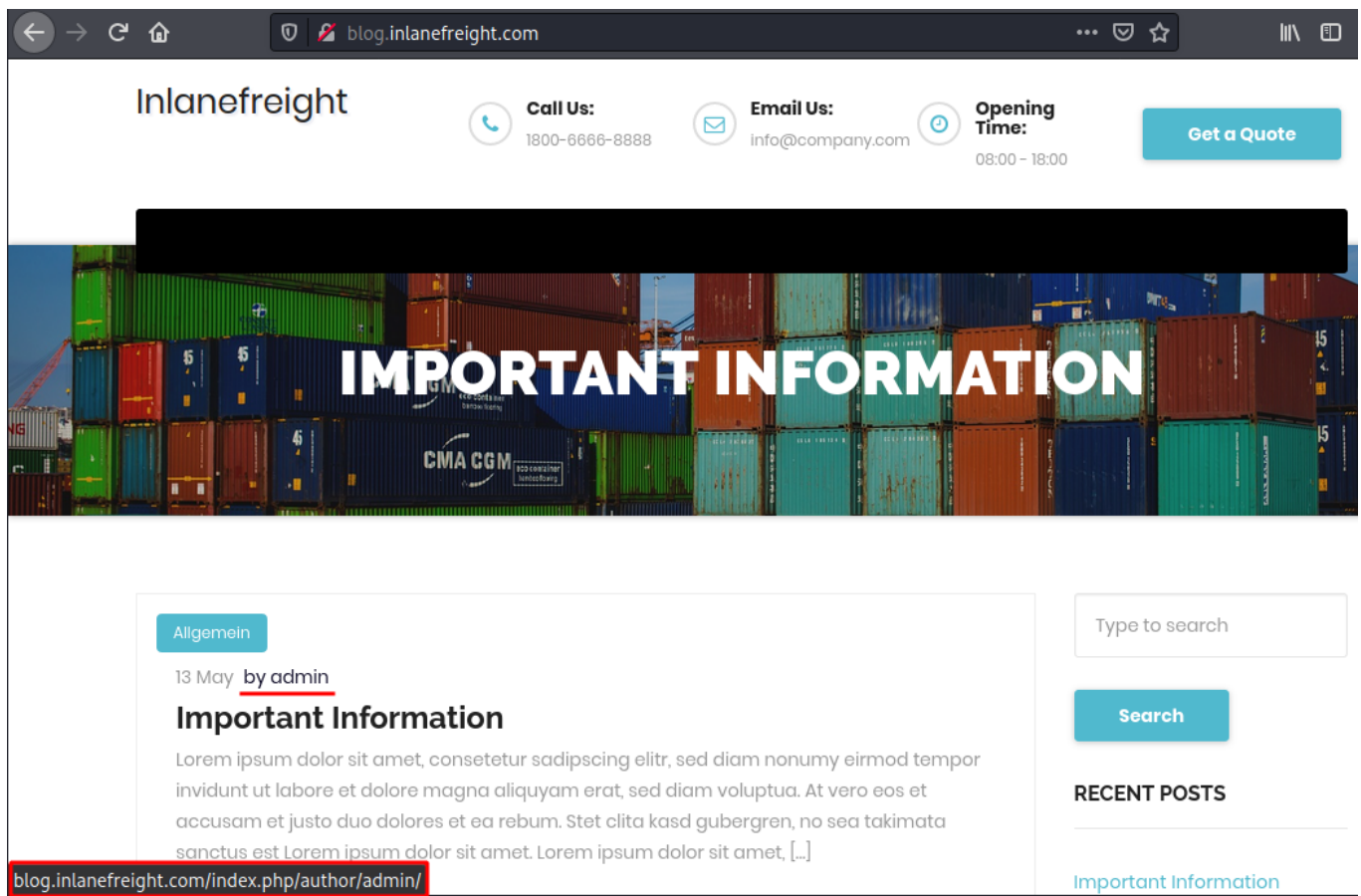
User Enumeration

Enumerating a list of valid users is a critical phase of a WordPress security assessment. Armed with this list, we may be able to guess default credentials or perform a brute force password attack. If successful, we may be able to log in to the WordPress backend as an author or even as an administrator. This access can potentially be leveraged to modify the WordPress website or even interact with the underlying web server.

There are two methods for performing manual username enumeration.

First Method

The first method is reviewing posts to uncover the ID assigned to the user and their corresponding username. If we mouse over the post author link titled "by admin," as shown in the below image, a link to the user's account appears in the web browser's lower-left corner.



The `admin` user is usually assigned the user ID `1`. We can confirm this by specifying the user ID for the `author` parameter in the URL.

<http://blog.inlanefreight.com/?author=1>

This can also be done with `cURL` from the command line. The HTTP response in the below output shows the author that corresponds to the user ID. The URL in the `Location` header confirms that this user ID belongs to the `admin` user.

Existing User

```
mayala@htb[/htb] $ curl -s -I -X GET http://blog.inlanefreight.com/?author=1
HTTP/1.1 301 Moved Permanently Date: Wed, 13 May 2020 20:47:08 GMT Server:
Apache/2.4.29 (Ubuntu) X-Redirect-By: WordPress Location:
http://blog.inlanefreight.com/index.php/author/admin/ Content-Length: 0 Content-
Type: text/html; charset=UTF-8
```

The above `cURL` request then redirects us to the user's profile page or the main login page. If the user does not exist, we receive a `404 Not Found` error.

Non-Existing User

```
mayala@htb[/htb] $ curl -s -I -X GET http://blog.inlanefreight.com/?author=100
HTTP/1.1 404 Not Found Date: Wed, 13 May 2020 20:47:14 GMT Server: Apache/2.4.29
(Ubuntu) Expires: Wed, 11 Jan 1984 05:00:00 GMT Cache-Control: no-cache, must-
revalidate, max-age=0 Link: <http://blog.inlanefreight.com/index.php/wp-json/>;
rel="https://api.w.org/" Transfer-Encoding: chunked Content-Type: text/html;
charset=UTF-8
```

Second Method

The second method requires interaction with the `JSON` endpoint, which allows us to obtain a list of users. This was changed in WordPress core after version 4.7.1, and later versions only show whether a user is configured or not. Before this release, all users who had published a post were shown by default.

JSON Endpoint

```
mayala@htb[/htb] $ curl http://blog.inlanefreight.com/wp-json/wp/v2/users | jq [ {
  "id": 1, "name": "admin", "url": "", "description": "", "link":
  "http://blog.inlanefreight.com/index.php/author/admin/", <SNIP> }, { "id": 2,
  "name": "ch4p", "url": "", "description": "", "link":
  "http://blog.inlanefreight.com/index.php/author/ch4p/", <SNIP> }, <SNIP>
```

Login

Once we are armed with a list of valid users, we can mount a password brute-forcing attack to attempt to gain access to the WordPress backend. This attack can be performed via the login page or the `xmlrpc.php` page.

If our POST request against `xmlrpc.php` contains valid credentials, we will receive the following output:

cURL - POST Request

```
mayala@htb[/htb] $ curl -X POST -d "<methodCall>
<methodName>wp.getUsersBlogs</methodName><params><param><value>admin</value>
</param><param><value>CORRECT-PASSWORD</value></param></params></methodCall>"
```



```
http://blog.inlanefreight.com/xmlrpc.php <?xml version="1.0" encoding="UTF-8"?>
<methodResponse> <params> <param> <value> <array><data> <value><struct> <member>
<name>isAdmin</name><value><boolean>1</boolean></value></member> <member>
<name>url</name><value><string>http://blog.inlanefreight.com/</string></value>
</member> <member><name>blogid</name><value><string>1</string></value></member>
<member><name>blogName</name><value><string>Inlanefreight</string></value>
</member> <member><name>xmlrpc</name><value>
<string>http://blog.inlanefreight.com/xmlrpc.php</string></value></member>
</struct></value> </data></array> </value> </param> </params> </methodResponse>
```

If the credentials are not valid, we will receive a 403 faultCode error.

Invalid Credentials - 403 Forbidden

```
mayala@htb[/htb] $ curl -X POST -d "<methodCall>
<methodName>wp.getUsersBlogs</methodName><params><param><value>admin</value>
</param><param><value>asdasd</value></param></params></methodCall>"
http://blog.inlanefreight.com/xmlrpc.php <?xml version="1.0" encoding="UTF-8"?>
<methodResponse> <fault> <value> <struct> <member> <name>faultCode</name>
<value><int>403</int></value> </member> <member> <name>faultString</name>
<value><string>Incorrect username or password.</string></value> </member>
</struct> </value> </fault> </methodResponse>
```

These last few sections introduced several methods for performing manual enumeration against a WordPress instance. It is essential to understand manual methods before attempting to use automated tools. While automated tools greatly speed up the penetration testing process, it is our responsibility to understand their impact on the systems we are assessing. A solid understanding of manual enumeration methods will also assist with troubleshooting should any automated tools not function properly or provide unexpected output.

WPScan Overview

Using WPScan

[WPScan](#) is an automated WordPress scanner and enumeration tool. It determines if the various themes and plugins used by a WordPress site are outdated or vulnerable. It is installed by default on Parrot OS but can also be installed manually with `gem`.

```
mayala@htb[/htb] $ gem install wpscan
```

Once the installation completes, we can issue a command such as `wpscan --hh` to verify the installation. This command will show us the usage menu with all of the available command-line switches.

```
mayala@htb[/htb] $ wpscan --hh
```

```
_____  
\\ / / _ \\ / ____| \\ \\ / / / | |_) | ( _ _ _ _ _ ® \\ \\ \\ / | _/  
\\ _ \\ / _|/ _` | ' _ \\ \\ / / | | ____ ) | ( _| ( _| | | | \\ \\ \\ | _| ____/  
\\ _|\\ _ , _| | | _| WordPress Security Scanner by the WPScan Team Version 3.8.1  
@_WPScan_, @ethicalhack3r, @erwan_lr, @firefart  
_____  
Usage: wpscan  
[options] --url URL The URL of the blog to scan Allowed Protocols: http, https  
Default Protocol if none provided: http This option is mandatory unless update  
or help or hh or version is/are supplied -h, --help Display the simple help and  
exit --hh Display the full help and exit --version Display the version and exit  
--ignore-main-redirect Ignore the main redirect (if any) and scan the target url  
-v, --verbose Verbose mode --[no-]banner Whether or not to display the banner  
Default: true --max-scan-duration SECONDS Abort the scan if it exceeds the time  
provided in seconds -o, --output FILE Output to FILE -f, --format FORMAT Output  
results in the format supplied Available choices: cli-no-colour, cli-no-color,  
json, cli <SNIP>
```

There are various enumeration options that can be specified, such as vulnerable plugins, all plugins, user enumeration, and more. It is important to understand all of the options available to us and fine-tune the scanner depending on the goal (i.e., are we just interested to see if the WordPress site is using any vulnerable plugins, do we need to perform a full audit of all aspects of the site or are we just interested in creating a user list to use in a brute force password guessing attack?).

WPScan can pull in vulnerability information from external sources to enhance our scans. We can obtain an API token from [WPVulnDB](#), which is used by WPScan to scan for vulnerability and exploit proof of concepts (POC) and reports. The free plan allows up to 50 requests per day. To use the WPVulnDB database, just create an account and copy the API token from the users page. This token can then be supplied to WPScan using the `--api-token` parameter.

Review the various WPScan options using the below Parrot instance by opening a shell and issuing the command `wpscan --hh`.

WPScan Enumeration

Enumerating a Website with WPScan

The `--enumerate` flag is used to enumerate various components of the WordPress application such as plugins, themes, and users. By default, WPScan enumerates vulnerable plugins, themes, users, media, and backups. However, specific arguments can be supplied to restrict enumeration to specific components. For example, all plugins can be enumerated using the arguments `--enumerate ap`. Let's run a normal enumeration scan against a WordPress website.

Note: The default number of threads used is 5, however, this value can be changed using the `-t` flag.

WPScan Enumeration

```
mayala@htb[/htb] $ wpscan --url http://blog.inlanefreight.com --enumerate --api-  
token Kffr4fdJzy9qVcTk<SNIP> [+] URL: http://blog.inlanefreight.com/ [+] Headers  
| - Server: Apache/2.4.38 (Debian) | - X-Powered-By: PHP/7.3.15 | Found By:  
Headers (Passive Detection) [+] XML-RPC seems to be enabled:  
http://blog.inlanefreight.com/xmlrpc.php | Found By: Direct Access (Aggressive  
Detection) | - http://codex.wordpress.org/XML-RPC_Pingback_API [+] The external  
WP-Cron seems to be enabled: http://blog.inlanefreight.com/wp-cron.php | Found  
By: Direct Access (Aggressive Detection) | - https://www.iplocation.net/defend-  
wordpress-from-ddos [+] WordPress version 5.3.2 identified (Latest, released on  
2019-12-18). | Found By: Rss Generator (Passive Detection) | -  
http://blog.inlanefreight.com/?feed=rss2, <generator>https://wordpress.org/?  
v=5.3.2</generator> [+] WordPress theme in use: twentytwenty | Location:  
http://blog.inlanefreight.com/wp-content/themes/twentytwenty/ | Readme:  
http://blog.inlanefreight.com/wp-content/themes/twentytwenty/readme.txt | [!]  
The version is out of date, the latest version is 1.2 | Style Name: Twenty  
Twenty [+] Enumerating Vulnerable Plugins (via Passive Methods) [i] Plugin(s)  
Identified: [+] mail-masta | Location: http://blog.inlanefreight.com/wp-  
content/plugins/mail-masta/ | Latest Version: 1.0 (up to date) | Found By: Urls  
In Homepage (Passive Detection) | [!] 2 vulnerabilities identified: | [!]  
Title: Mail Masta 1.0 - Unauthenticated Local File Inclusion (LFI) | -  
https://www.exploit-db.com/exploits/40290/ | [!] Title: Mail Masta 1.0 -  
Multiple SQL Injection | - https://wpvulndb.com/vulnerabilities/8740 [+] wp-  
google-places-review-slider | [!] 1 vulnerability identified: | [!] Title: WP  
Google Review Slider <= 6.1 - Authenticated SQL Injection | Reference:  
https://wpvulndb.com/vulnerabilities/9933 [i] No themes Found. <SNIP> [i] No  
Config Backups Found. <SNIP> [i] No Medias Found. [+] Enumerating Users (via  
Passive and Aggressive Methods) <SNIP> [i] User(s) Identified: [+] admin | Found  
By: Author Posts - Display Name (Passive Detection) | Confirmed By: | Author Id
```

```
Brute Forcing - Author Pattern (Aggressive Detection) | Login Error Messages  
(Aggressive Detection) [+] david <SNIP> [+] roger <SNIP>
```

WPScan uses various passive and active methods to determine versions and vulnerabilities, as shown in the scan output above.