Test for Configuration and deployment

Aim: To famializrize tools configuration and deployment testing

Description:

The intrinsic complexity of interconnected and heterogeneous web server infrastructure, which can include hundreds of web applications, makes configuration management and review a fundamental step in testing and deploying every single application. It takes only a single vulnerability to undermine the security of the entire infrastructure, and even small and seemingly unimportant problems may evolve into severe risks for another application on the same server. In order to address these problems, it is of utmost importance to perform an in-depth review of configuration and known security issues, after having mapped the entire architecture.

Proper configuration management of the web server infrastructure is very important in order to preserve the security of the application itself. If elements such as the web server software, the back-end database servers, or the authentication servers are not properly reviewed and secured, they might introduce undesired risks or introduce new vulnerabilities that might compromise the application itself.

For example, a web server vulnerability that would allow a remote attacker to disclose the source code of the application itself (a vulnerability that has arisen a number of times in both web servers or application servers) could compromise the application, as anonymous users could use the information disclosed in the source code to leverage attacks against the application or its users.

The following steps need to be taken to test the configuration management infrastructure:

- ✓ The different elements that make up the infrastructure need to be determined in order to understand how they interact with a web application and how they affect its security.
- ✓ All the elements of the infrastructure need to be reviewed in order to make sure that they don't contain any known vulnerabilities.
- ✓ A review needs to be made of the administrative tools used to maintain all the different elements.

- ✓ The authentication systems, need to reviewed in order to assure that they serve the needs of the application and that they cannot be manipulated by external users to leverage access.
- ✓ A list of defined ports which are required for the application should be maintained and kept under change control.

<u>Procedure:</u>

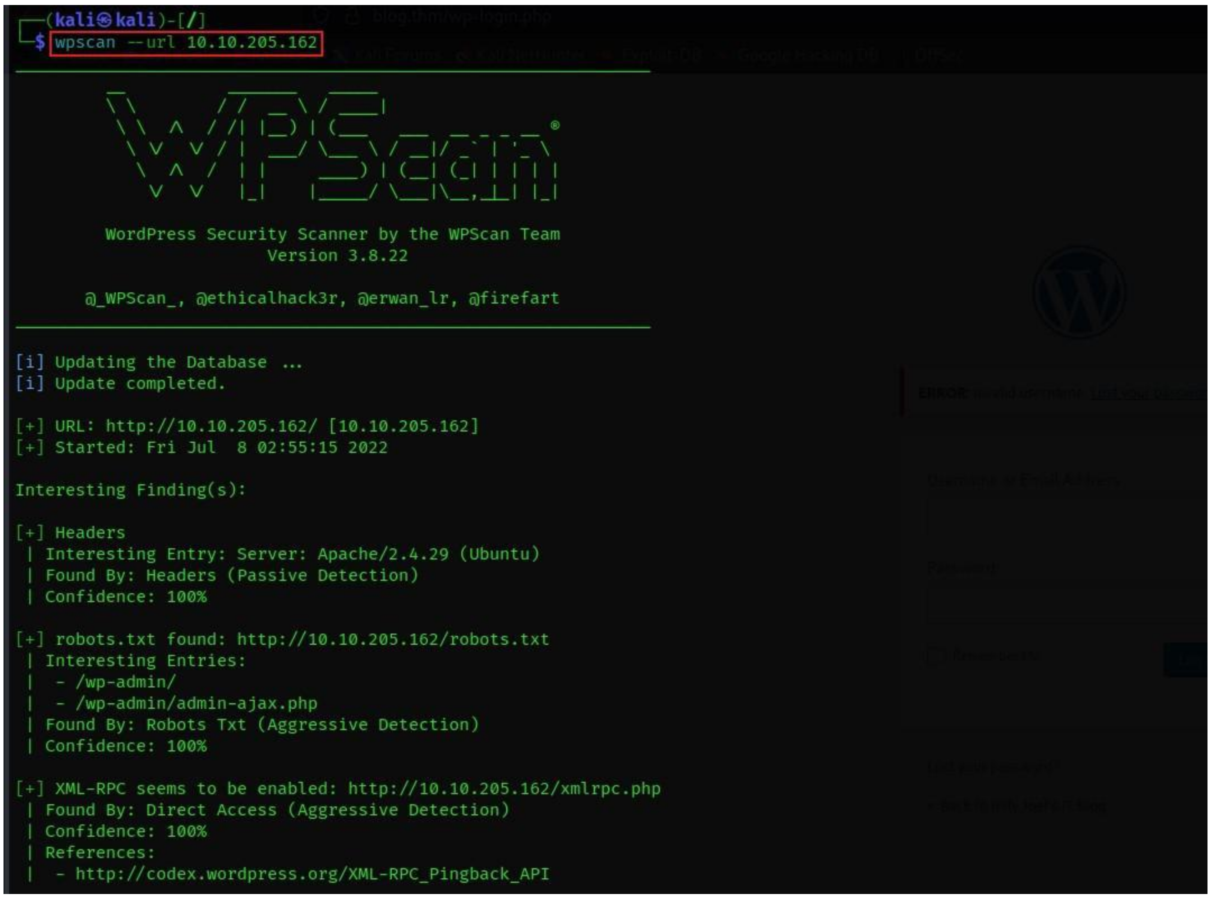
<u>tools:</u>

1.WPScan

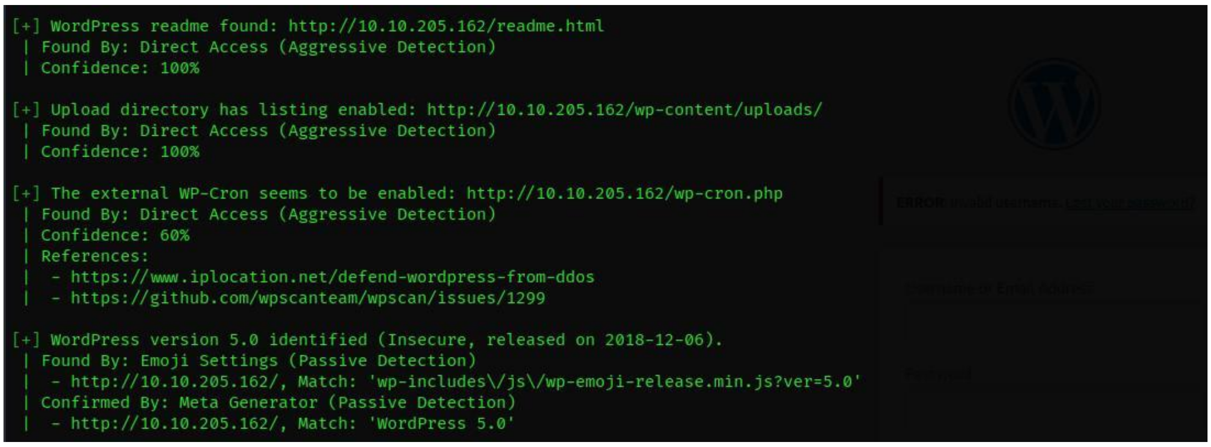
Wpscan is a WordPress security scanner used to test WordPress installations and WordPress-powered websites. This is a command line tool used in Kali Linux. This tool can be used to find any vulnerable plugins, themes, or backups running on the site. It is usually used by individual WordPress site owners to test their own websites for vulnerabilities and also by large organizations to maintain a secure website. This tool can also be used to enumerate users and perform brute-force attacks on known WordPress users. In this article, We are going to take you through different commands of wpscan tool, the most commonly used attacks on WordPress sites, and tips to defend against them. The below functionalities of this tool can be used from the point of view of a hacker or even just someone who wants to test if their WordPress site is secure enough.

- ***–url : (basic scan)***

  wpscan –url IP_ADDRESS_OF_WEBSITE



```
┌──(kali㉿kali)-[/]
└─$ wpscan --url 10.10.205.162
_______________________________________________________________
         __          _______   _____
         \ \        / /  __ \ / ____|
          \ \  /\  / /| |__) | (___   ___  __ _ _ __ ®
           \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
            \  /\  /  | |     ____) | (__| (_| | | | |
             \/  \/   |_|    |_____/ \___|\__,_|_| |_|

         WordPress Security Scanner by the WPScan Team
                         Version 3.8.22

       @_WPScan_, @ethicalhack3r, @erwan_lr, @firefart
_______________________________________________________________

[i] Updating the Database ...
[i] Update completed.

[+] URL: http://10.10.205.162/ [10.10.205.162]
[+] Started: Fri Jul  8 02:55:15 2022

Interesting Finding(s):

[+] Headers
 | Interesting Entry: Server: Apache/2.4.29 (Ubuntu)
 | Found By: Headers (Passive Detection)
 | Confidence: 100%

[+] robots.txt found: http://10.10.205.162/robots.txt
 | Interesting Entries:
 |  - /wp-admin/
 |  - /wp-admin/admin-ajax.php
 | Found By: Robots Txt (Aggressive Detection)
 | Confidence: 100%

[+] XML-RPC seems to be enabled: http://10.10.205.162/xmlrpc.php
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%
 | References:
 |  - http://codex.wordpress.org/XML-RPC_Pingback_API
```

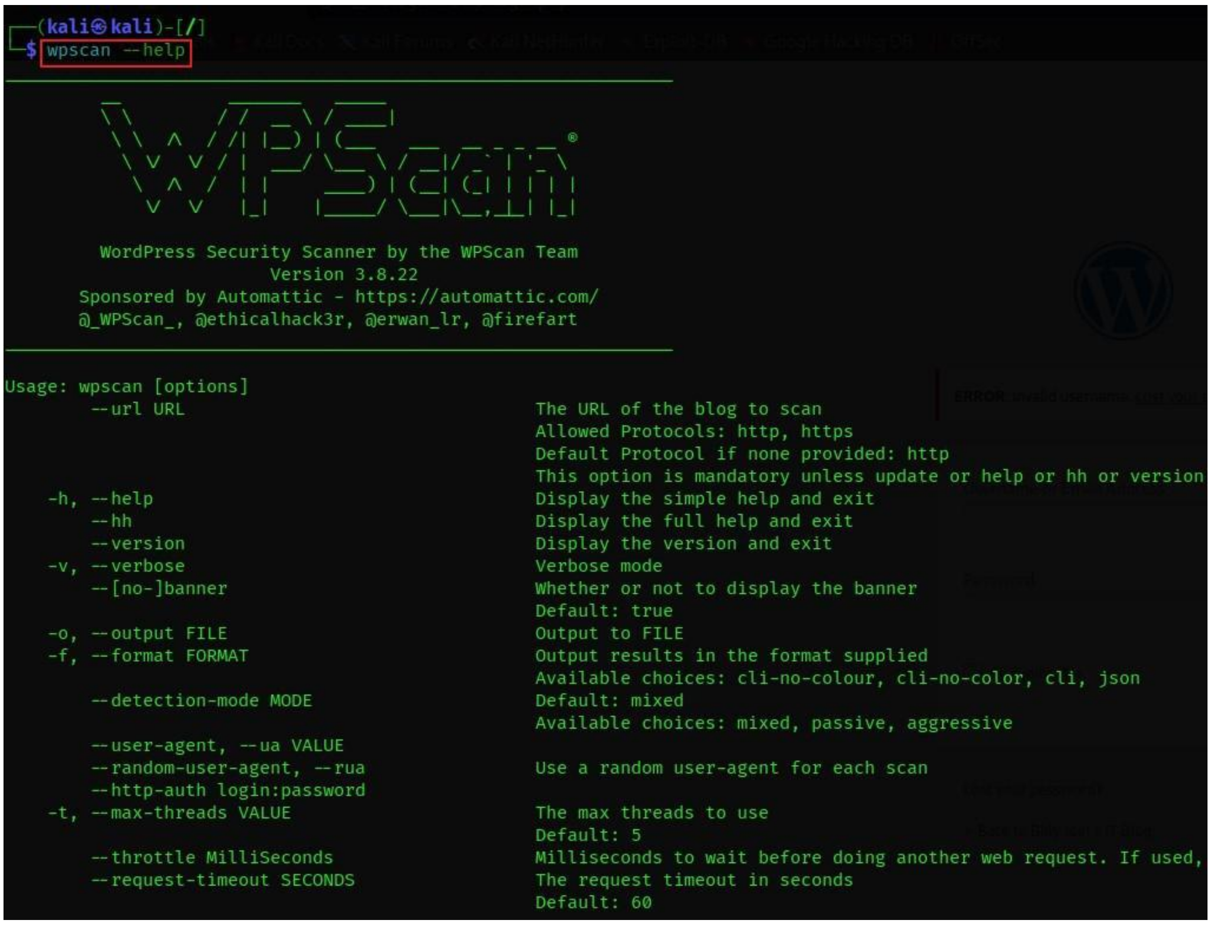


```
[+] WordPress readme found: http://10.10.205.162/readme.html
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%

[+] Upload directory has listing enabled: http://10.10.205.162/wp-content/uploads/
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%

[+] The external WP-Cron seems to be enabled: http://10.10.205.162/wp-cron.php
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 60%
 | References:
 |  - https://www.iplocation.net/defend-wordpress-from-ddos
 |  - https://github.com/wpscanteam/wpscan/issues/1299

[+] WordPress version 5.0 identified (Insecure, released on 2018-12-06).
 | Found By: Emoji Settings (Passive Detection)
 |  - http://10.10.205.162/, Match: 'wp-includes\/js\/wp-emoji-release.min.js?ver=5.0'
 | Confirmed By: Meta Generator (Passive Detection)
 |  - http://10.10.205.162/, Match: 'WordPress 5.0'
```

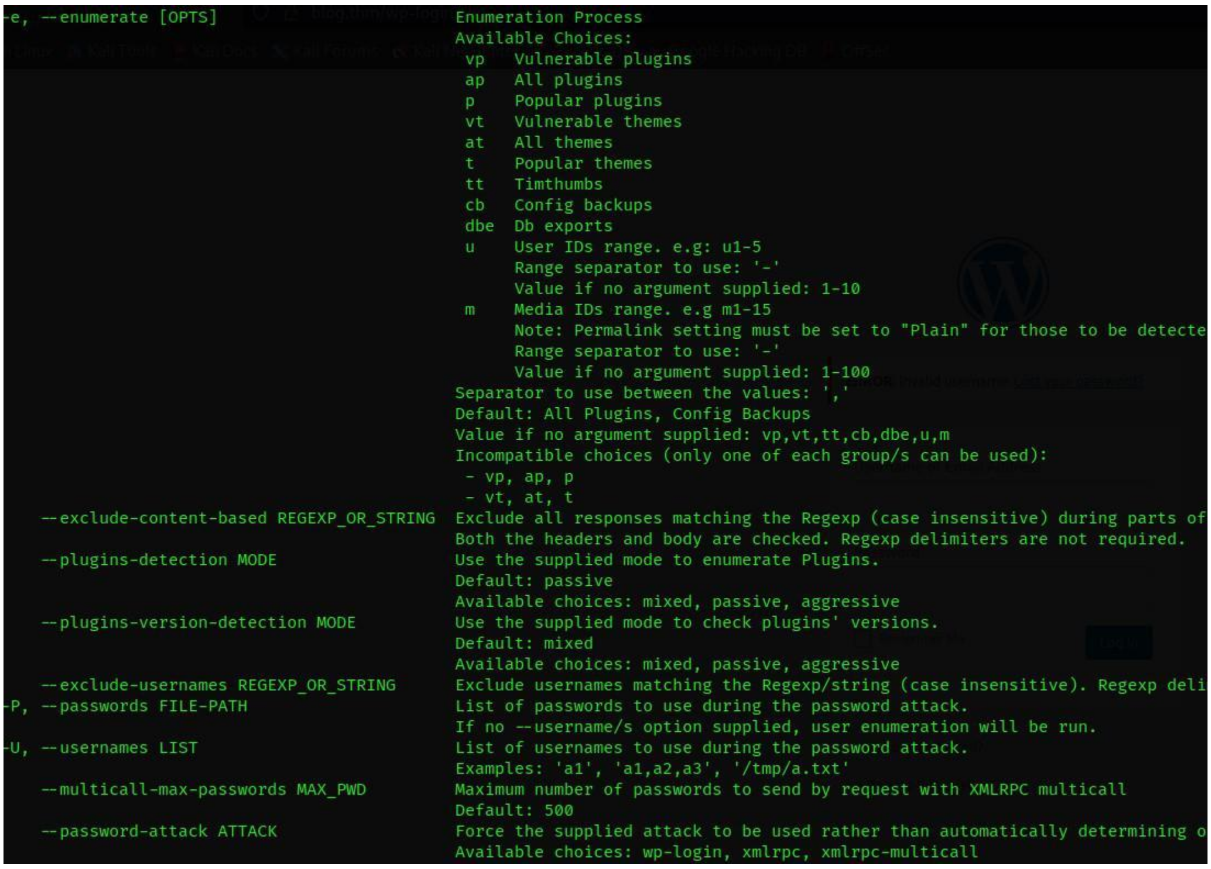
- **–help: (help options)**

  wpscan --help

It’s a common practice in Linux to use the “–help” option to get the complete list of the usability of the tool using different switches for different functionalities.



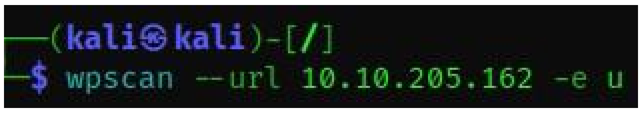
```
┌──(kali㉿kali)-[/]
└─$ wpscan --help
_______________________________________________________________
         __          _______   _____
         \ \        / /  __ \ / ____|
          \ \  /\  / /| |__) | (___   ___  __ _ _ __ ®
           \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
            \  /\  /  | |     ____) | (__| (_| | | | |
             \/  \/   |_|    |_____/ \___|\__,_|_| |_|

         WordPress Security Scanner by the WPScan Team
                         Version 3.8.22
       Sponsored by Automattic - https://automattic.com/
       @_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
    -h, --help                                        Display the simple help and exit
        --hh                                          Display the full help and exit
        --version                                     Display the version and exit
    -v, --verbose                                     Verbose mode
        --[no-]banner                                 Whether or not to display the banner
                                                      Default: true
    -o, --output FILE                                 Output to FILE
    -f, --format FORMAT                               Output results in the format supplied
                                                      Available choices: cli-no-colour, cli-no-color, cli, json
        --detection-mode MODE                         Default: mixed
                                                      Available choices: mixed, passive, aggressive
        --user-agent, --ua VALUE
        --random-user-agent, --rua                    Use a random user-agent for each scan
        --http-auth login:password
    -t, --max-threads VALUE                           The max threads to use
                                                      Default: 5
        --throttle MilliSeconds                       Milliseconds to wait before doing another web request. If used,
        --request-timeout SECONDS                     The request timeout in seconds
                                                      Default: 60
```



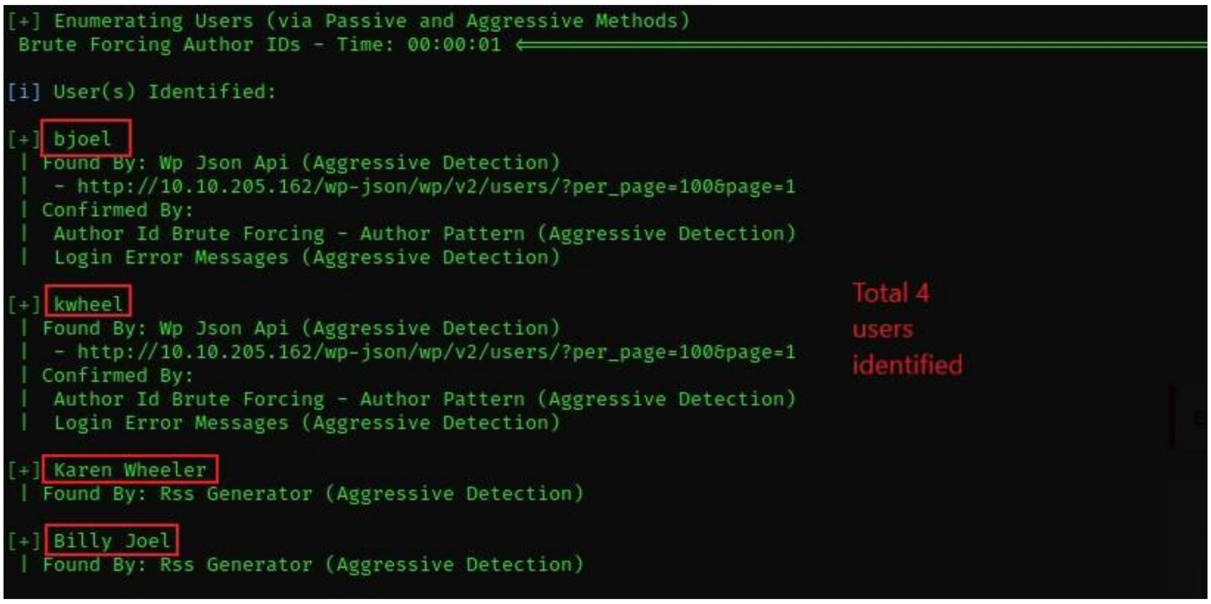
```
-e, --enumerate [OPTS]                         Enumeration Process
                                               Available Choices:
                                                vp   Vulnerable plugins
                                                ap   All plugins
                                                p    Popular plugins
                                                vt   Vulnerable themes
                                                at   All themes
                                                t    Popular themes
                                                tt   Timthumbs
                                                cb   Config backups
                                                dbe  Db exports
                                                u    User IDs range. e.g: u1-5
                                                     Range separator to use: '-'
                                                     Value if no argument supplied: 1-10
                                                m    Media IDs range. e.g m1-15
                                                     Note: Permalink setting must be set to "Plain" for those to be detecte
                                                     Range separator to use: '-'
                                                     Value if no argument supplied: 1-100
                                               Separator to use between the values: ','
                                               Default: All Plugins, Config Backups
                                               Value if no argument supplied: vp,vt,tt,cb,dbe,u,m
                                               Incompatible choices (only one of each group/s can be used):
                                                - vp, ap, p
                                                - vt, at, t
    --exclude-content-based REGEXP_OR_STRING   Exclude all responses matching the Regexp (case insensitive) during parts of
                                               Both the headers and body are checked. Regexp delimiters are not required.
    --plugins-detection MODE                   Use the supplied mode to enumerate Plugins.
                                               Default: passive
                                               Available choices: mixed, passive, aggressive
    --plugins-version-detection MODE           Use the supplied mode to check plugins' versions.
                                               Default: mixed
                                               Available choices: mixed, passive, aggressive
    --exclude-usernames REGEXP_OR_STRING       Exclude usernames matching the Regexp/string (case insensitive). Regexp deli
-P, --passwords FILE-PATH                      List of passwords to use during the password attack.
                                               If no --username/s option supplied, user enumeration will be run.
-U, --usernames LIST                           List of usernames to use during the password attack.
                                               Examples: 'a1', 'a1,a2,a3', '/tmp/a.txt'
    --multicall-max-passwords MAX_PWD          Maximum number of passwords to send by request with XMLRPC multicall
                                               Default: 500
    --password-attack ATTACK                   Force the supplied attack to be used rather than automatically determining o
                                               Available choices: wp-login, xmlrpc, xmlrpc-multicall
```

- ***-e u: (enumerating website users)***
  wpscan –url IP_ADDRESS_OF_WEBSITE -e u

This lets the wpscan tool enumerate the WordPress site for valid login usernames. After the scan, it would give all the usernames the tool has enumerated which are valid users of the WordPress site and are often times brute forced to gain unauthorized access to the WordPress admin/author dashboard.



```
┌──(kali㉿kali)-[/]
└─$ wpscan --url 10.10.205.162 -e u
```



```
[+] Enumerating Users (via Passive and Aggressive Methods)
 Brute Forcing Author IDs - Time: 00:00:01 <==========================

[i] User(s) Identified:

[+] bjoel
 | Found By: Wp Json Api (Aggressive Detection)
 |  - http://10.10.205.162/wp-json/wp/v2/users/?per_page=100&page=1
 | Confirmed By:
 |  Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 |  Login Error Messages (Aggressive Detection)

[+] kwheel
 | Found By: Wp Json Api (Aggressive Detection)
 |  - http://10.10.205.162/wp-json/wp/v2/users/?per_page=100&page=1
 | Confirmed By:
 |  Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 |  Login Error Messages (Aggressive Detection)

[+] Karen Wheeler
 | Found By: Rss Generator (Aggressive Detection)

[+] Billy Joel
 | Found By: Rss Generator (Aggressive Detection)
```

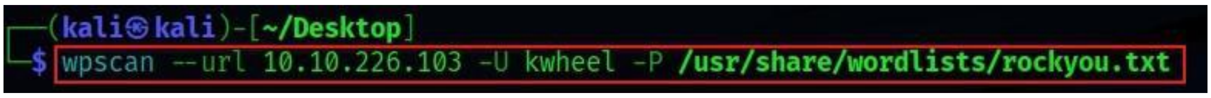
- ***-U -P: (brute-forcing password for the identified user)***

  wpscan –url IP_ADDRESS_OF_WEBSITE -U USER_NAME -P PATH_TO_WORDLIST

As we managed to enumerate some usernames for the WordPress site above, let's try to brute-force the user "kwheel".

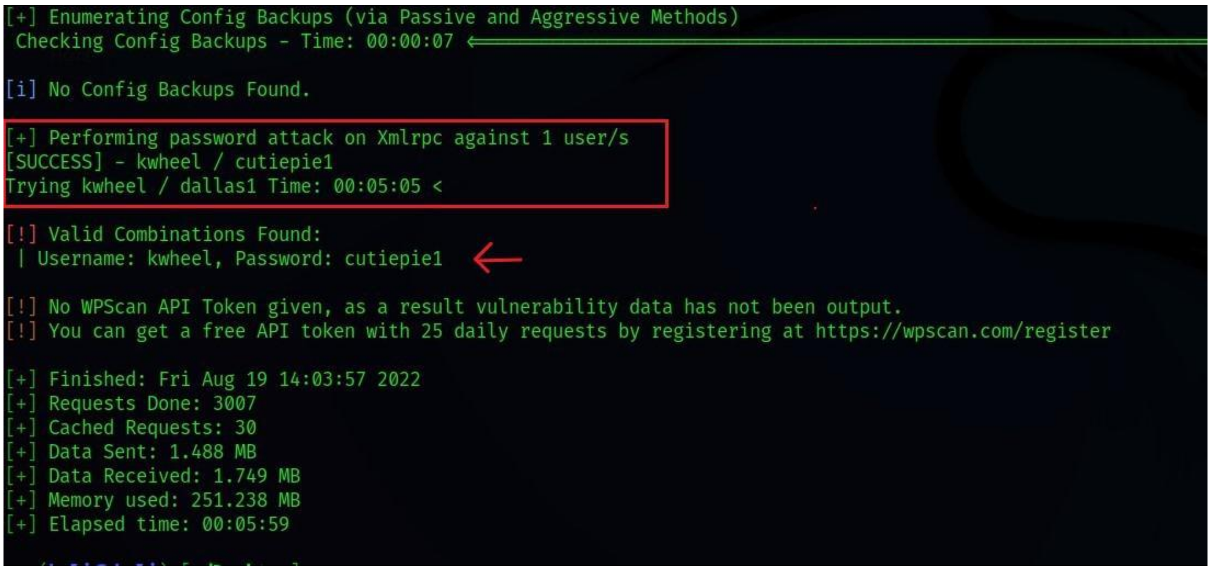
- **Brute forcing**: It is a technique where a wordlist is used against a tool that effectively finds the matching password for the given username.
- **Wordlist**: The password cracking tool uses a wordlist, which is nothing but a text file containing a set of commonly used passwords. Ex: 12345678, qwerty, pizza123,etc.

Now let's try to brute force the user "kwheel".



```
┌──(kali㉿kali)-[~/Desktop]
└─$ wpscan --url 10.10.226.103 -U kwheel -P /usr/share/wordlists/rockyou.txt
```

rockyou.txt is a quite commonly used wordlist for brute force attacks.



```
[+] Enumerating Config Backups (via Passive and Aggressive Methods)
 Checking Config Backups - Time: 00:00:07 <=====================================

[i] No Config Backups Found.

[+] Performing password attack on Xmlrpc against 1 user/s
[SUCCESS] - kwheel / cutiepie1
Trying kwheel / dallas1 Time: 00:05:05 <

[!] Valid Combinations Found:
 | Username: kwheel, Password: cutiepie1

[!] No WPScan API Token given, as a result vulnerability data has not been output.
[!] You can get a free API token with 25 daily requests by registering at https://wpscan.com/register

[+] Finished: Fri Aug 19 14:03:57 2022
[+] Requests Done: 3007
[+] Cached Requests: 30
[+] Data Sent: 1.488 MB
[+] Data Received: 1.749 MB
[+] Memory used: 251.238 MB
[+] Elapsed time: 00:05:59
```

2. OWASP ZAP

ZAP has installers for Windows, Linux, and Mac OS/X, as well as Docker images. Download the appropriate installer from the download page and install it on the machine where you will run the penetration test.

Java 8 or higher is required to run ZAP. The Mac OS/X installer includes the appropriate Java version, but Java 8+ must be installed separately for Windows, Linux, and cross-platform versions. The Docker version already includes Java.

When you start ZAP for the first time, you need to choose whether to make the ZAP session persistent. If you persist the session, it will be saved to a local HSQLDB. Otherwise, files will be deleted when you log out of ZAP.

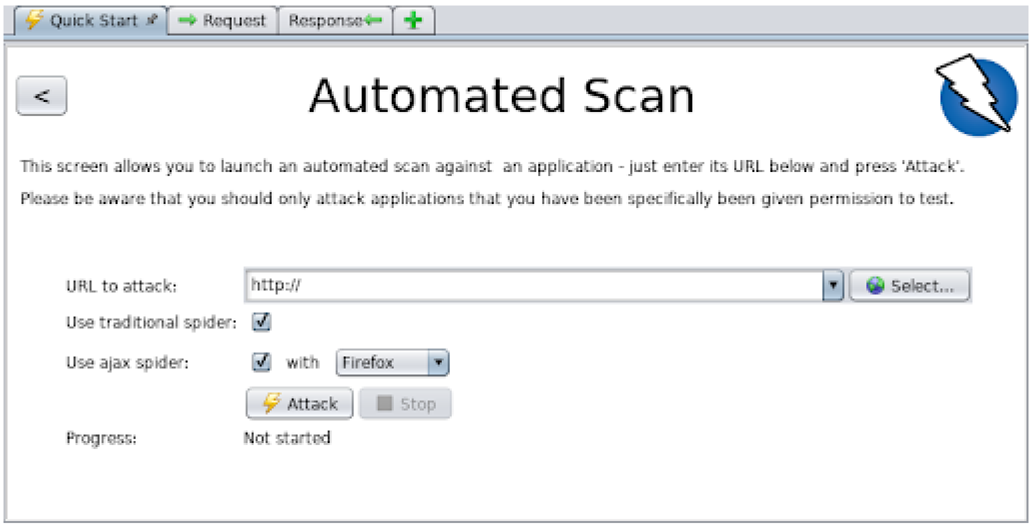
Before proceeding, ensure you have permission from the web application owner to perform a penetration test.

Run a quick start auto scan:

- Start ZAP and click the Quick Launch tab in the workspace window.
- Click the Auto Scan button.

- In the Attack URL text box, enter the full URL of the web application.
- Select either Use traditional spider, Use ajax spider, or both (more details below)
- Click Attack.

1. Click **Attack**.



This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.
Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://
Use traditional spider: ✓
Use ajax spider: ✓ with Firefox
Progress: Not started

ZAP uses a crawler to go through the web application and scan pages it finds. It then uses the active scanner to attack every page, function, and parameter it finds.

ZAP-spiders

ZAP provides two spiders for scraping web applications, which you can select in the automated scan dialog:

- The traditional ZAP spider inspects HTML in a web application's response to detect links. Although this spider is fast, it is less effective when navigating AJAX web applications that use JavaScript to generate links.

- The ZAP AJAX spider is more effective for JavaScript applications. It navigates a web application by invoking a browser, rendering the full JavaScript of the page, and following any links on the resulting page. AJAX spiders are slower than traditional spiders and require additional configuration to be used in a headless environment. ZAP uses two forms of scanning:

- Passive scanning investigates all proxy requests and responses, but does not change the response in any way and is considered safe. It can be done on a background thread so it doesn't slow down the application. This can find some vulnerabilities and can help you understand the basic security posture of a web application.

- Active scanning attempts to find additional vulnerabilities using known attack vectors against the selected target. Do not use active scans against targets you don't have permission to test, as active scans are real attacks that might cause damage .