

**Introduction to WPScan**

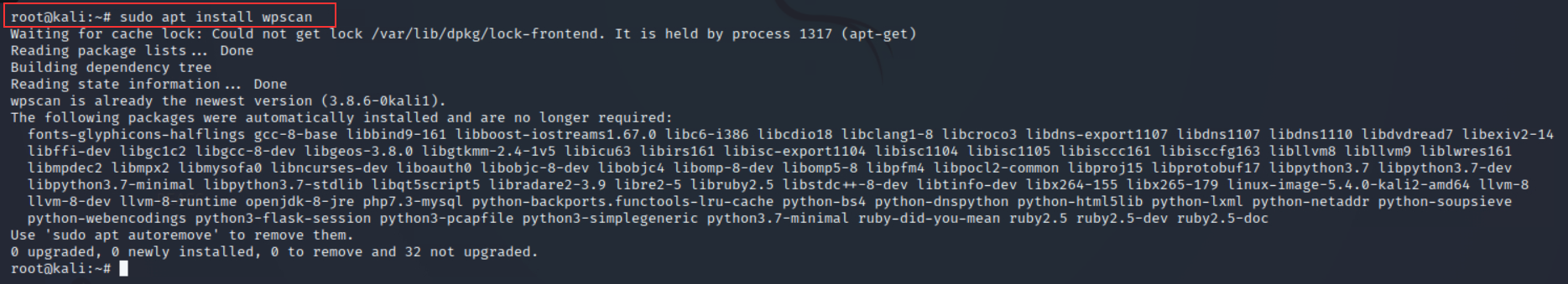
First released in June 2011, WPScan has survived the tests of time and stood out as a tool that every pentester should have in their toolkits.

The WPScan framework is capable of enumerating & researching a few security vulnerability categories present in WordPress sites - including - but not limited to:

* Sensitive Information Disclosure (Plugin & Theme installation versions for disclosed vulnerabilities or CVE's)
* Path Discovery (Looking for misconfigured file permissions i.e. wp-config.php)
* Weak Password Policies (Password bruteforcing)
* Presence of Default Installation (Looking for default files)
* Testing Web Application Firewalls (Common WAF plugins)

**Installing WPScan**

Thankfully for us, WPScan comes pre-installed on the latest versions of penetration testing systems such as Kali Linux and Parrot. If you are using an older version of Kali Linux (such as 2019) for example, WPScan is in the apt repository, so can be installed by a simple sudo apt update && sudo apt install wpscan

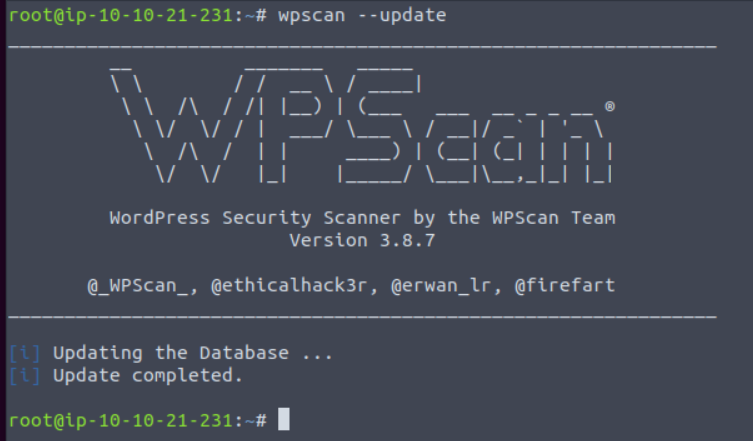


﻿Installing WPScan on other operating systems such as Ubuntu or Debian involves extra steps. Whilst the TryHackMe AttackBox comes pre-installed with WPScan, you can follow the [developer's installation guide](https://github.com/wpscanteam/wpscan#install) for your local environment.

**A Primer on WPScan's Database**

WPScan uses information within a local database as a primary reference point when enumerating for themes and plugins. As we'll come to detail later, a technique that WPScan uses when enumerating is looking for common themes and plugins. Before using WPScan, it is highly recommended that you update this database before performing any scans.

Thankfully, this is an easy process to do. Simply run wpscan --update



In the next task, we will explore some of the more useful features of WPScan!

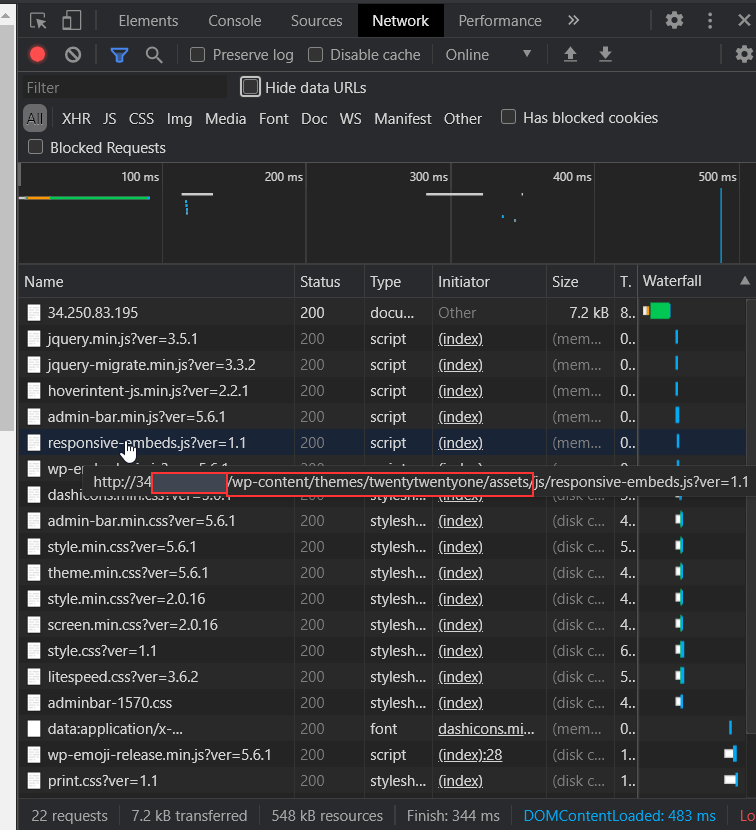
2.1. WPScan Modes

We briefly discussed the various things that ﻿WPScan is capable of discovering on a system running WordPress in Task 7. However, let's dive into this a bit further, demonstrate a few examples of the various scans used to retrieve this information and highlighting how these scans work exactly.

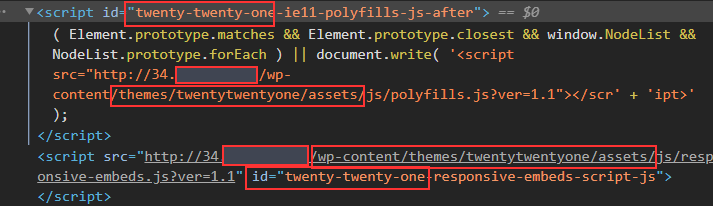
**Enumerating for Installed Themes**

WPScan has a few methods of determining the active theme on a running WordPress installation. At a premise, it boils down to a technique that we can manually do ourselves. Simply, we can look at the assets our web browser loads and then looks for the location of these on the webserver. Using the "Network" tab in your web browsers developer tools, you can see what files are loaded when you visit a webpage.

Take the screenshot below, we can see many assets are loaded, some of these will be scripts & the stylings of the theme that determines how the browser renders the website. Highlighted in the screenshot below is the URL: <http://redacted/wp-content/themes/twentytwentyone/assets/>



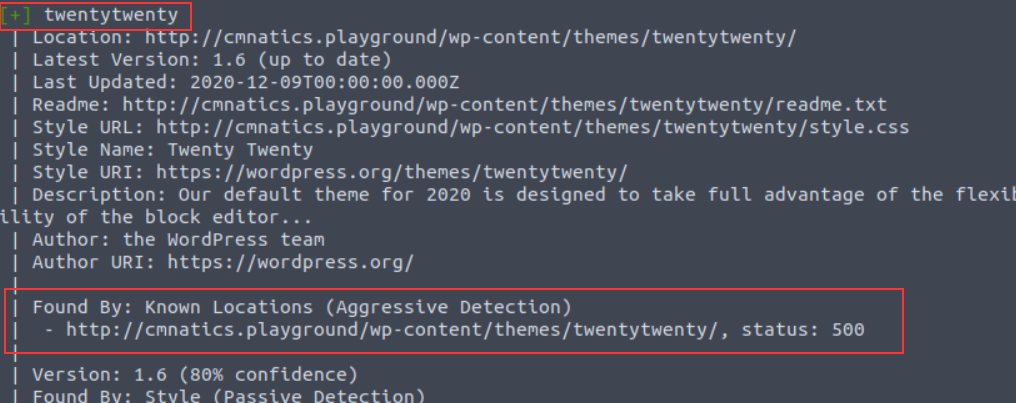
 We can take a pretty good guess that the name of the current theme is "twentytwentyone". After inspecting the source code of the website, we can note additional references to "twentytwentyone"



However, let's use WPScan to speed this process up by using the --enumerate flag with the t argument like so:

wpscan --url http://cmnatics.playground/ --enumerate t

After a couple of minutes, we can begin to see some results:



The great thing about WPScan is that the tool lets you know how it determined the results it has got. In this case, we're told that the "twentytwenty" theme was confirmed by scanning "*Known Locations*". The "twentytwenty" theme is the default WordPress theme for WordPress versions in 2020.

**Enumerating for Installed Plugins**

A very common feature of webservers is "Directory Listing" and is often enabled by default. Simply, "Directory Listing" is the listing of files in the directory that we are navigating to (just as if we were to use Windows Explorer or Linux's ls command. URL's in this context are very similar to file paths. The URL <http://cmnatics.playground/a/directory> is actually the configured root of the *webserver/a/directory:*

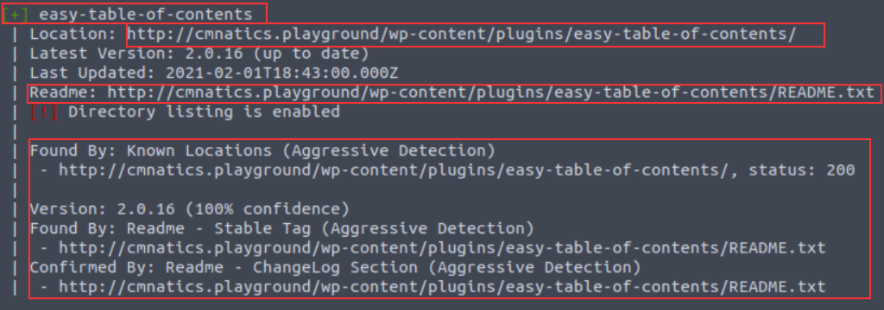
https://assets.tryhackme.com/additional/web-enumeration-redux/webserver-fs.png

"Directory Listing" occurs when there is no file present that the webserver has been told to process. A very common file is "index.html" and "index.php". As these files aren't present in /a/directory, the contents are instead displayed:

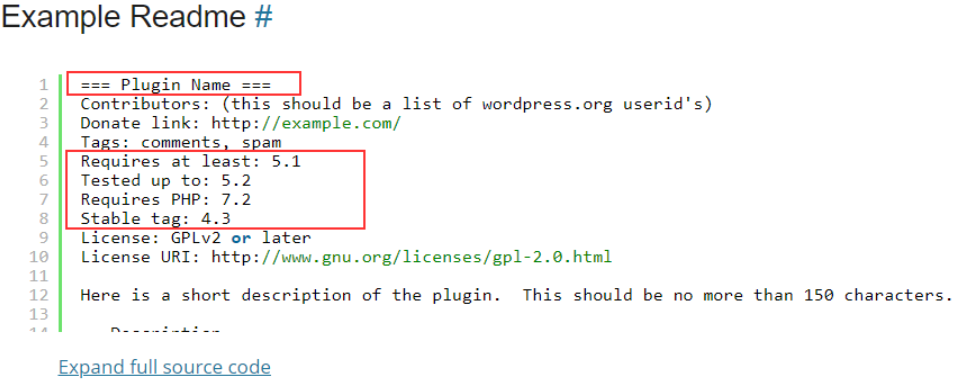


WPScan can leverage this feature as one technique to look for plugins installed. Since they will all be located in /wp-content/plugins/pluginname, WPScan can enumerate for common/known plugins.

In the screenshot below, "easy-table-of-contents" has been discovered. Great! This could be vulnerable. To determine that, we need to know the version number. Luckily, this handed to us on a plate by WordPress.



Reading through WordPress' developer documentation, we can learn about "[Plugin Readme's](https://developer.wordpress.org/plugins/wordpress-org/how-your-readme-txt-works/#how-the-readme-is-parsed)" to figure out how WPScan determined the version number. Simply, plugins must have a "README.txt" file. This file contains meta-information such as the plugin name, the versions of WordPress it is compatible with and a description.



[An Example ReadMe. (WordPress Developer Documentation., 2021)](https://developer.wordpress.org/plugins/wordpress-org/how-your-readme-txt-works/#example-readme)

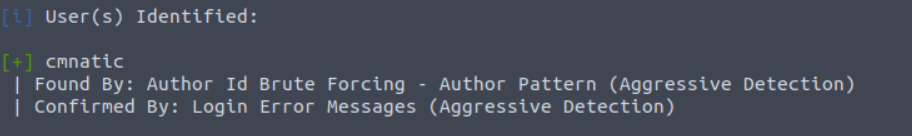
WPScan uses additional methods to discover plugins (such as looking for references or embeds on pages for plugin assets). We can use the --enumerate flag with the p argument like so:

wpscan --url http://cmnatics.playground/ --enumerate p

**Enumerating for Users**

We've highlighted that WPScan is capable of performing brute-forcing attacks. Whilst we must provide a password list such as*rockyou.txt*, the way how WPScan enumerates for users is interestingly simple. WordPress sites use authors for posts. Authors are in fact a type of user. 

And sure enough, this author is picked up by our WPScan:



This scan was performed by using the --enumerate flag with the u argument like so:

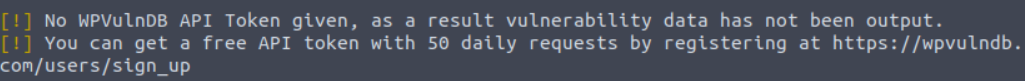
wpscan --url http://cmnatics.playground/ --enumerate u

**The "Vulnerable" Flag**

In the commands so far, we have only enumerated WordPress to discover what themes, plugins and users are present. At the moment, we'd have to look at the output and use sites such as MITRE, NVD and CVEDetails to look up the names of these plugins and the version numbers to determine any vulnerabilities.

WPScan has the v argument for the--enumerate flag. We provide this argument alongside another (such as p for plugins). For example, our syntax would like so: wpscan --url http://cmnatics.playground/ --enumerate vp

**Note, that this requires setting up WPScan to use the WPVulnDB API which is out-of-scope for this room.**



**Performing a Password Attack**

After determining a list of possible usernames on the WordPress install, we can use WPScan to perform a bruteforcing technique against the username we specify and a password list that we provide. Simply, we use the output of our username enumeration to build a command like so: wpscan –-url http://cmnatics.playground –-passwords rockyou.txt –-usernames cmnatic

https://assets.tryhackme.com/additional/web-enumeration-redux/password-attack.png

**Adjusting WPScan's Aggressiveness (WAF)**

Unless specified, WPScan will try to be as least "noisy" as possible. Lots of requests to a web server can trigger things such as firewalls and ultimately result in you being blocked by the server.

This means that some plugins and themes may be missed by our WPScan. Luckily, we can use arguments such as --plugins-detection and an aggressiveness profile (passive/aggressive) to specify this. For example: --plugins-detection aggressive

**Summary - Cheatsheet**

|  |  |  |
| --- | --- | --- |
| Flag | Description | Full Example |
| p | Enumerate Plugins | --enumerate p |
| t | Enumerate Themes | --enumerate t |
| u | Enumerate Usernames | --enumerate -u |
| v | Use WPVulnDB to cross-reference for vulnerabilities. Example command looks for vulnerable plugins (p) | --enumerate vp |
| aggressive | This is an aggressiveness profile for WPScan to use. | --plugins-detection aggressive |