2025 Spring CTF Report from Team 22

- Chris Talavera
- Jackson Xu
- Joel Han
- Yucheng Zhang

Executive Summary

- Capture the Flag challenged our ability to find flags on a target site by “thinking like an attacker”, with the goal of identifying and exploiting known vulnerabilities.
- Out of 15 flags, we successfully “captured” 11 of them and recorded our process of how we acquired (and in some cases stumbled!) upon the flags.
  - Out of the available flags, we were unable to capture any worth 400 points. Unsolved Flags: 4, 9, 13, 14
- Several tools we used were extremely valuable and uncovered hidden files/repositories containing flags.
- We all gained hands-on experience and learned new skills that will prove valuable for us in the future.

Two VERY useful tools used in vulnerability scan

Gobuster: Find hidden files and web directories

Gobuster helped us solve several challenges by finding hidden files and directories that we couldn’t see just by browsing the site. Before diving into how we solved each challenge, we’ll first show Gobuster outputs. These gave us clues like secret admin pages, robots.txt, .git, and cs40 homeworks that led us to the flags.

```
gobuster dir -u http://3.145.206.165 \
    -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt \
    -x php,html,js,json,txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://3.145.206.165
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Extensions:              php,html,js,json,txt
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/index.php            (Status: 301) [Size: 0] [--> http://3.145.206.165/]
/login.php            (Status: 302) [Size: 0] [--> admin.php?error]
/main.php             (Status: 302) [Size: 0] [--> admin.php]
/data.txt             (Status: 200) [Size: 200701]
/wp-content           (Status: 301) [Size: 169] [--> http://3.145.206.165/wp-content/]
/admin.php            (Status: 200) [Size: 472]
/board.php            (Status: 200) [Size: 1597]
/wp-login.php         (Status: 200) [Size: 5016]
/includes             (Status: 301) [Size: 169] [--> http://3.145.206.165/includes/]
/license.txt          (Status: 200) [Size: 19915]
/wp-includes          (Status: 301) [Size: 169] [--> http://3.145.206.165/wp-includes/]
```

```
/logout.php           (Status: 302) [Size: 0] [--> admin.php]
/readme.html          (Status: 200) [Size: 7466]
/robots.txt           (Status: 200) [Size: 8135]
/wp-trackback.php     (Status: 200) [Size: 135]
/scoreboard           (Status: 301) [Size: 169] [--> http://3.145.206.165/scoreboard/]
/wp-admin             (Status: 301) [Size: 169] [--> http://3.145.206.165/wp-admin/]
/xmlrpc.php           (Status: 405) [Size: 42]
/logger.php           (Status: 200) [Size: 99]
/wp-signup.php        (Status: 302) [Size: 0] [--> /wp-login.php?action=register]
Progress: 1323360 / 1323366 (100.00%)
===============================================================
Finished
===============================================================
```

WPScan: WordPress Security Scanner

WPScan is a powerful command-line tool used to scan WordPress websites for known vulnerabilities.

We used WPScan to analyze the target server and uncover various security issues.

Similar to Gobuster, WPScan provided valuable information, such as the discovery of `readme.html` and access to directories like http://3.145.206.165/wp-content/uploads/.

Additionally, WPScan helped us enumerate usernames, crack the password — detailed further in the `bobo` section.

```
$ wpscan --url http://3.145.206.16
Interesting Finding(s):

[+] Headers
 | Interesting Entry: Server: nginx/1.22.1
 | Found By:
 | Confidence: 100%

[+] robots.txt found: http://3.145.206.165/robots.txt
 | Found By:
 | Confidence: 100%

[+] XML-RPC seems to be enabled: http://3.145.206.165/xmlrpc.php
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%
 | References:
    xxxx
[+] WordPress readme found: http://3.145.206.165/readme.html
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%

[+] Upload directory has listing enabled: http://3.145.206.165/wp-content/uploads/
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%

[+] The external WP-Cron seems to be enabled: http://3.145.206.165/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 6.2 identified (Insecure, released on 2023-03-29).
 | Found By:
 |  - http://3.145.206.165/, Match: 'wp-includes\/js\/wp-emoji-release.min.js?ver=6.2'
 | Confirmed By:
 |  - http://3.145.206.165/, Match: 'WordPress 6.2'
 |
 | [!] 18 vulnerabilities identified:
 |
 | [!] Title: WP <= 6.2 - Unauthenticated Blind SSRF via DNS Rebinding
 |     References:
 |      - https://wpscan.com/vulnerability/c8814e6e-78b3-4f63-a1d3-6906a84c1f11
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-3590
 |      - https://blog.sonarsource.com/wordpress-core-unauthenticated-blind-ssrf/
 |
 | [!] Title: WP < 6.2.1 - Directory Traversal via Translation Files
 |     Fixed in: 6.2.1
 |     References:
 |      - https://wpscan.com/vulnerability/2999613a-b8c8-4ec0-9164-5dfe63adf6e6
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2023-2745
 |
 | [!] Title: WP < 6.2.1 - Thumbnail Image Update via CSRF
 |     Fixed in: 6.2.1
 |     References:
 |      - https://wpscan.com/vulnerability/a03d744a-9839-4167-a356-3e7da0f1d532

 |
 | [!] Title: WP < 6.2.1 - Contributor+ Stored XSS via Open Embed Auto Discovery
 |     Fixed in: 6.2.1
 |     References:
 |      - https://wpscan.com/vulnerability/3b574451-2852-4789-bc19-d5cc39948db5
 |
 | [!] Title: WP < 6.2.2 - Shortcode Execution in User Generated Data
 |     Fixed in: 6.2.2
 |     References:
 |      - https://wpscan.com/vulnerability/ef289d46-ea83-4fa5-b003-0352c690fd89
 |      - https://wordpress.org/news/2023/05/wordpress-6-2-2-security-release/
 |
 | [!] Title: WP < 6.2.1 - Contributor+ Content Injection
 |     Fixed in: 6.2.1
 |     References:
 |      - https://wpscan.com/vulnerability/1527ebdb-18bc-4f9d-9c20-8d729a628670
 |
 | [!] Title: WP 5.6-6.3.1 - Contributor+ Stored XSS via Navigation Block
 |     Fixed in: 6.2.3
 |     References:
 |      - https://wpscan.com/vulnerability/cd130bb3-8d04-4375-a89a-883af131ed3a
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2023-38000
 |
 | [!] Title: WP 5.6-6.3.1 - Reflected XSS via Application Password Requests
 |     Fixed in: 6.2.3
 |     References:
 |      - https://wpscan.com/vulnerability/da1419cc-d821-42d6-b648-bdb3c70d91f2
```

```
|
| [!] Title: WP < 6.3.2 - Denial of Service via Cache Poisoning
|     Fixed in: 6.2.3
|     References:
|      - https://wpscan.com/vulnerability/6d80e09d-34d5-4fda-81cb-e703d0e56e4f
|
| [!] Title: WP < 6.3.2 - Subscriber+ Arbitrary Shortcode Execution
|     Fixed in: 6.2.3
|     References:
|      - https://wpscan.com/vulnerability/3615aea0-90aa-4f9a-9792-078a90af7f59
|
| [!] Title: WP < 6.3.2 - Contributor+ Comment Disclosure
|     Fixed in: 6.2.3
|     References:
|      - https://wpscan.com/vulnerability/d35b2a3d-9b41-4b4f-8e87-1b8ccb370b9f
|      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2023-39999
|
| [!] Title: WP < 6.3.2 - Unauthenticated Post Author Email Disclosure
|     Fixed in: 6.2.3
|     References:
|      - https://wpscan.com/vulnerability/19380917-4c27-4095-abf1-eba6f913b441
|      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2023-5561
|
| [!] Title: WordPress < 6.4.3 - Deserialization of Untrusted Data
|     Fixed in: 6.2.4
|     References:
|      - https://wpscan.com/vulnerability/5e9804e5-bbd4-4836-a5f0-b4388cc39225
|
| [!] Title: WordPress < 6.4.3 - Admin+ PHP File Upload
|     Fixed in: 6.2.4
|     References:
|      - https://wpscan.com/vulnerability/a8e12fbe-c70b-4078-9015-cf57a05bdd4a
|
| [!] Title: WP < 6.5.2 - Unauthenticated Stored XSS
|     Fixed in: 6.2.5
|     References:
|      - https://wpscan.com/vulnerability/1a5c5df1-57ee-4190-a336-b0266962078f
|
| [!] Title: WordPress < 6.5.5 - Contributor+ Stored XSS in HTML API
|     Fixed in: 6.2.6
|     References:
|      - https://wpscan.com/vulnerability/2c63f136-4c1f-4093-9a8c-5e51f19eae28
|      - https://wordpress.org/news/2024/06/wordpress-6-5-5/
|
| [!] Title: WordPress < 6.5.5 - Contributor+ Stored XSS in Template-Part Block
|     Fixed in: 6.2.6
|     References:
|      - https://wpscan.com/vulnerability/7c448f6d-4531-4757-bff0-be9e3220bbbb
|      - https://wordpress.org/news/2024/06/wordpress-6-5-5/
|
| [!] Title: WordPress < 6.5.5 - Contributor+ Path Traversal in Template-Part Block
|     Fixed in: 6.2.6
|     References:
|      - https://wpscan.com/vulnerability/36232787-754a-4234-83d6-6ded5e80251c
```

```
 |      - https://wordpress.org/news/2024/06/wordpress-6-5-5/

[+] WordPress theme in use: twentysixteen
 | Location: http://3.145.206.165/wp-content/themes/twentysixteen/
 | Last Updated: 2024-11-13T00:00:00.000Z
 | Readme: http://3.145.206.165/wp-content/themes/twentysixteen/readme.txt
 | [!] The version is out of date, the latest version is 3.4
 | Style Name: Twenty Sixteen
 | Style URI: https://wordpress.org/themes/twentysixteen/
 | Description: Twenty Sixteen is a modernized take on an ever-popular WordPress
   layout - the horizontal masthead wi...
 | Author: the WordPress team
 | Author URI: https://wordpress.org/
 |


[+] Enumerating All Plugins (via Passive Methods)

[i] No plugins Found.

[+] Enumerating Config Backups (via Passive and Aggressive Methods)


[i] No Config Backups Found.

[+] WPScan DB API OK
 | Plan: free
 | Requests Done (during the scan): 2
 | Requests Remaining: 23
```

Challenge 1: ROTten to the Core

Problem

Gubfr bs lbh jub ner gnxvat pbzchgre fpvrapr pynffrf va fpubby znl ng guvf cbvag or guvaxvat, bx,jr'ir tbg guvf fbegrq. Jr'er nyernql orvat gnhtug nyy nobhg cebtenzzvat. Ohg fbeel, guvf vf abg rabhtu. Lbh unir gb or jbexvat ba lbhe bja cebwrpgf, abg whfg yrneavat fghss va pynffrf. Lbh pna qb jryy va pbzchgre fpvrapr pynffrf jvgubhg rire ernyyl yrneavat gb cebtenz. Va snpg lbh pna tenqhngr jvgu n qrterr va pbzchgre fpvrapr sebz n gbc havirefvgl naq fgvyy abg or nal tbbq ng cebtenzzvat. Gung'f jul grpu pbzcnavrf nyy znxr lbh gnxr n pbqvat grfg orsber gurl'yy uver lbh, ertneqyrff bs jurer lbh jrag gb havirefvgl be ubj jryy lbh qvq gurer. Gurl xabj tenqrf naq rknzerfhygf cebir abguvat.

xrl{4n247351p63n867os26q505q095p37284rsp3802087onpnp363n418184pp7506}

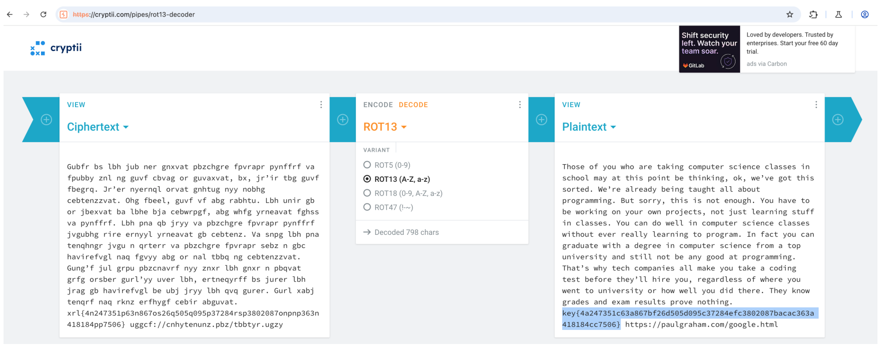
uggcf://cnhytenunz.pbz/tbbtyr.ugzy

Method This problem is very straightforward.

`xrl{4n247351p63n867os26q505q095p37284rsp3802087onpnp363n418184pp7506}` match the pattern of `key{xxxxxx}`.

With rot13-decoder, the text is decoded and and the below key is revealed

`key{4a247351c63a867bf26d505d095c37284efc3802087bacac363a418184cc7506}`


https://cryptii.com/pipes/rot13-decoder
cryptii
Shift security left. Watch your team soar.
GitLab
Loved by developers. Trusted by enterprises. Start your free 60 day trial.
ads via Carbon
VIEW
Ciphertext
ENCODE
DECODE
ROT13
VARIANT
ROT5 (0-9)
ROT18 (0-9, A-Z, a-z)
ROT47 (!-~)
Decoded 798 chars
VIEW
Plaintext

Variant: ROT13 (A-Z, a-z)

```
Gubfr bs lbh jub ner gnxvat pbzchgre fpvrapr pynffrf va
fpubby znl ng guvf cbvag or guvaxvat, bx, jr'ir tbg guvf
fbegrq. Jr'er nyernql orvat gnhtug nyy nobhg
cebtenzzvat. Ohg fbeel, guvf vf abg rabhtu. Lbh unir gb
or jbexvat ba lbhe bja cebwrpgf, abg whfg yrneavat fghss
va pynffrf. Lbh pna qb jryy va pbzchgre fpvrapr pynffrf
jvgubhg rire ernyyl yrneavat gb cebtenz. Va snpg lbh pna
tenqhngr jvgu n qrterr va pbzchgre fpvrapr sebz n gbc
havirefvgl naq fgvyy abg or nal tbbq ng cebtenzzvat.
Gung'f jul grpu pbzcnavrf nyy znxr lbh gnxr n pbqvat
grfg orsber gurl'yy uver lbh, ertneqyrff bs jurer lbh
jrag gb havirefvgl be ubj jryy lbh qvq gurer. Gurl xabj
tenqrf naq rknz erfhygf cebir abguvat.
xrl{4n247351p63n867os26q505q095p37284rsp3802087onpnp363n
418184pp7506} uggcf://cnhytenunz.pbz/tbbtyr.ugzy
```

```
Those of you who are taking computer science classes in
school may at this point be thinking, ok, we've got this
sorted. We're already being taught all about
programming. But sorry, this is not enough. You have to
be working on your own projects, not just learning stuff
in classes. You can do well in computer science classes
without ever really learning to program. In fact you can
graduate with a degree in computer science from a top
university and still not be any good at programming.
That's why tech companies all make you take a coding
test before they'll hire you, regardless of where you
went to university or how well you did there. They know
grades and exam results prove nothing.
key{4a247351c63a867bf26d505d095c37284efc3802087bacac363a
418184cc7506} https://paulgraham.com/google.html
```

Figure 1: Challenge 1 ROTten to the Core

Challenge 2: I hope I didn't make this too easy: another flag is on the blog.

Problem

There must be another flag on the blog…

Solution

Screenshot of flag `key{5925189030bc2af596c7ccc8d925c292ca0e25165965caba71e9d5fafaebd744}`

Exact Location A Flag Is Here… Blog Post

Method This is a multi-layer Base64 encoded text. We used the below python script to decode 20 layers to finally got the key `key{5925189030bc2af596c7ccc8d925c292ca0e25165965caba71e9d5fafaebd744}`.

```
import base64

# Read the input
data = open("encoded.txt", "rb").read()

# Try decoding up to 50 layers
for i in range(50):
    try:
        print(f"[+] Decoding layer {i+1}")
        data = base64.b64decode(data)
    except Exception as e:
        print(f"[-] Stopped decoding at layer {i+1}: {e}")
        break

# Save final decoded data
with open("decoded_final.bin", "wb") as f:
    f.write(data)
```

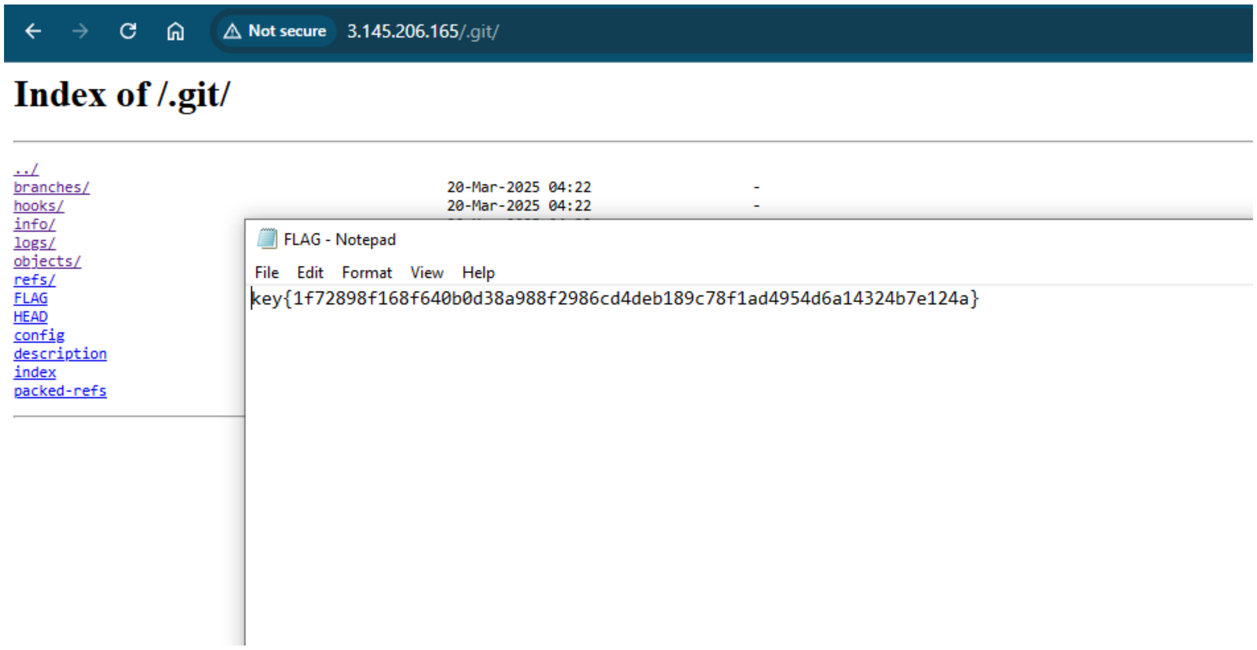
- Joel: Alternatively - plugging it into CyberChef and applying 19 Base46 decodes was mildly entertaining :)

Challenge 3: .git the FLAG.

Problem

Find the flag, HINT: `.git`

Solution



3.145.206.165/.git/

Index of /.git/

```
../
branches/                                          20-Mar-2025 04:22                   -
hooks/                                             20-Mar-2025 04:22                   -
info/
logs/
objects/
refs/
FLAG
HEAD
config
description
index
packed-refs
```

FLAG - Notepad

```
key{1f72898f168f640b0d38a988f2986cd4deb189c78f1ad4954d6a14324b7e124a}
```

Figure 2: Challenge 3 Flag

Screenshot of flag

Exact Location FLAG file from http://3.145.206.165/.git/ endpoint

Method After uncovering a hidden endpoint by brute-forcing the URL with `/.git/`, we were able to access a number of files and directories exposed by the Git repository. Among these was a file named `FLAG`, which, once downloaded and opened in a text editor, revealed the flag.

Challenge 5: Don't ask me if something looks wrong. Look again, pay careful attention

Problem

Questionable login page... is there any way to bypass this without knowing the username?

Solution

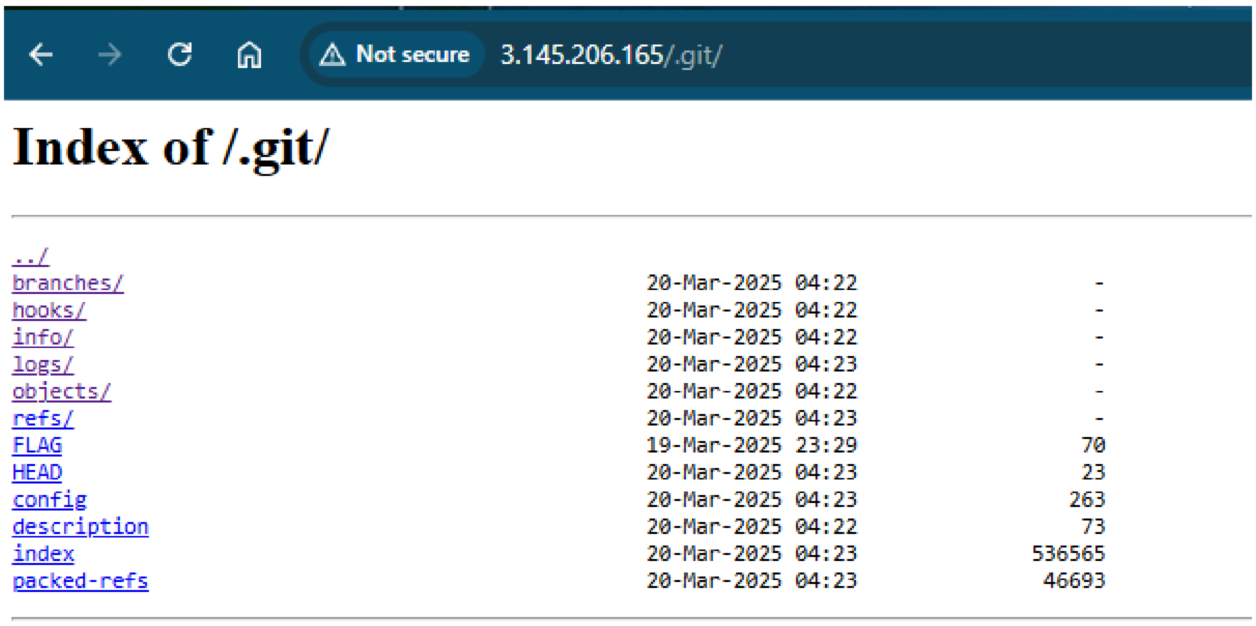
SQL Injection

Screenshot of flag

Exact Location http://3.145.206.165/main.php

Method

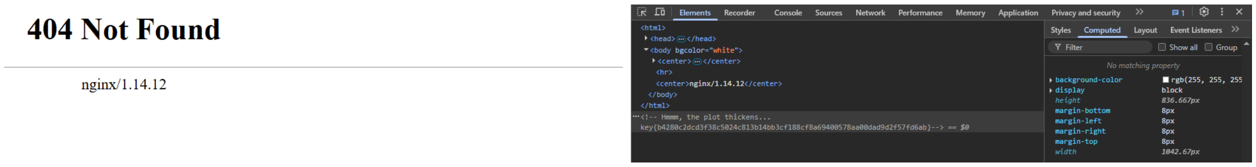
- Seeing the login information, we recalled homework 7 and decided to try sql injection.
- We tried different username & `a' OR '1=1` as passwords but was not able to login
- Eventually we decided to attempt SQL injection in both the username and password by entering `a' OR '1=1` for both and was able to login. We were then directed to a **404 papge** and the flag is not immediately accessible. However, we were able to locate the flag after clicking **inspect** on the webpage.



```
Index of /.git/

../
branches/                                          20-Mar-2025 04:22                   -
hooks/                                             20-Mar-2025 04:22                   -
info/                                              20-Mar-2025 04:22                   -
logs/                                              20-Mar-2025 04:23                   -
objects/                                           20-Mar-2025 04:22                   -
refs/                                              20-Mar-2025 04:23                   -
FLAG                                               19-Mar-2025 23:29                  70
HEAD                                               20-Mar-2025 04:23                  23
config                                             20-Mar-2025 04:23                 263
description                                        20-Mar-2025 04:22                  73
index                                              20-Mar-2025 04:23              536565
packed-refs                                        20-Mar-2025 04:23               46693
```

Figure 3: .git Directory



404 Not Found

nginx/1.14.12

```
<html>
  <head>…</head>
  <body bgcolor="white">
    <center>…</center>
    <hr>
    <center>nginx/1.14.12</center>
  </body>
</html>
<!-- Hmmm, the plot thickens...
key{b4280c2dcd3f38c5024c813b14bb3cf188cf8a69400578aa00dad9d2f57fd6ab}--> == $0
```

```
background-color    rgb(255, 255, 255
display             block
height              836.667px
margin-bottom       8px
margin-left         8px
margin-right        8px
margin-top          8px
width               1042.67px
```

Figure 4: Challenge 5 flag

- **Note to Ming**: We found this flag prior to game stoppage last week. After the game came back online, we were no longer able to find this flag. When we tried SQL injection on the same page referenced here, we were instead directed to the flag for challenge 6 (see below).

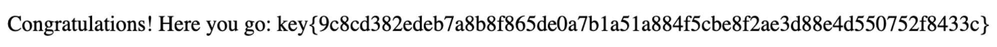
Challenge 6: Don't ask me if something looks wrong. Look again, pay really careful attention

Problem

Questionable login page... is there any way to bypass this without knowing the username?

Solution

SQL Injection



Congratulations! Here you go: key{9c8cd382edeb7a8b8f865de0a7b1a51a884f5cbe8f2ae3d88e4d550752f8433c}

Figure 5: Challenge 6 flag

Screenshot of flag

Exact Location http://3.145.206.165/main.php

Method

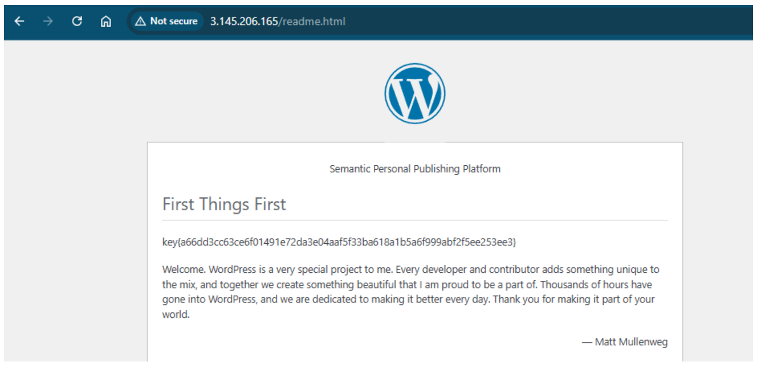
- Seeing the login information, we recalled homework 7 and decided to try sql injection.
- We tried different username & `a' OR '1=1` as passwords but was not able to login
- Eventually we decided to attempt SQL injection in both the username and password by entering `a' OR '1=1` for both and was able to login. The flag is immediately available after we login.

Challenge 7: That readme is peculiar...

Problem

Where is the readme? Where is the repo? Can we git clone?

Solution



Semantic Personal Publishing Platform

First Things First

key{a66dd3cc63ce6f01491e72da3e04aaf5f33ba618a1b5a6f999abf2f5ee253ee3}

Welcome. WordPress is a very special project to me. Every developer and contributor adds something unique to the mix, and together we create something beautiful that I am proud to be a part of. Thousands of hours have gone into WordPress, and we are dedicated to making it better every day. Thank you for making it part of your world.

— Matt Mullenweg

Figure 6: Readme Spotted

Screenshot of flag

Exact Location http://3.145.206.165/readme.html

We found this flag on the readme page.

Method How many file extensions for readme could there be?

We tried this the brute force way trying different file extensions until we got a bite

- readme.php-
- readme.txt-
- readme.png
- readme.html

We eventually found it again on `Gobuster`...

Challenge 8: A whole bunch of CS40 homeworks found

Problem

Find the homeworks, find the flag

Method

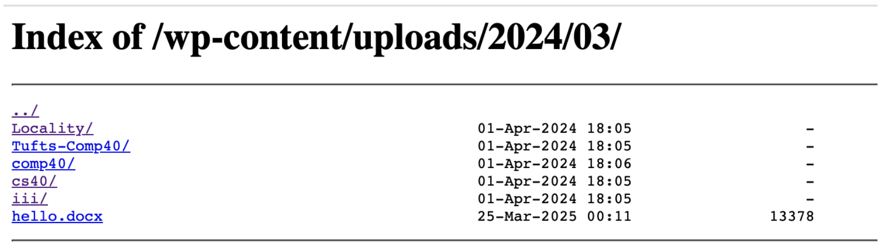
During our scan with WPSscan, several subpages revealed.

```
wpscan --url http://3.145.206.165
```

One interesting directory we found was:

```
http://3.145.206.165/wp-content/uploads/
```

Inside, there are three folders named: **2022**, **2024**, and **2025**. Under `2024/03`, we discovered CS40 homeworks:



Index of /wp-content/uploads/2024/03/

```
../
Locality/                                          01-Apr-2024 18:05                   -
Tufts-Comp40/                                      01-Apr-2024 18:05                   -
comp40/                                            01-Apr-2024 18:06                   -
cs40/                                              01-Apr-2024 18:05                   -
iii/                                               01-Apr-2024 18:05                   -
hello.docx                                         25-Mar-2025 00:11               13378
```

Figure 7: cs40 homework

While searching through these files for a flag, we found one named `hello.docx`. At first, we tried to open it in Microsoft Word, but it didn't render properly. So we checked the file type using `file`:

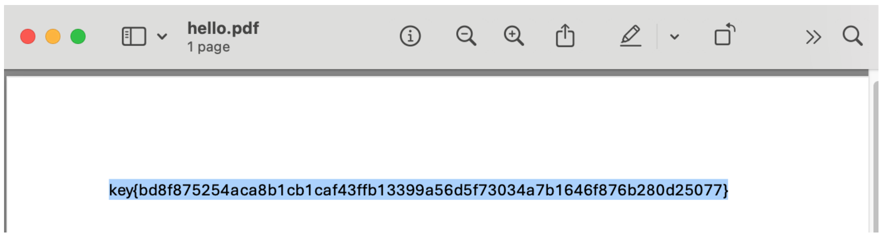
```
$ file hello.docx
hello.docx: PDF document, version 1.7, 1 pages
```

It identified the file as a PDF rather than a Word document. After renaming it to `hello.pdf`, We opened it and successfully found the flag inside the PDF.

Challenge 10: About my friend bobo

Problem

Bo Bo is the only other user besides admin (Ming) that has posted. Also the only friend of admin. Location of the flag will have something to do with Bo Bo



key{bd8f875254aca8b1cb1caf43ffb13399a56d5f73034a7b1646f876b280d25077}

Figure 8: cs40 flag

Method

Enumerate usernames with WPScan Withi WPScan, we identified two users: `admin` and `bobo`

```
$ wpscan --url http://3.145.206.165 --enumerate u
[i] User(s) Identified:

[+] admin
 | Found By: Author Posts - Display Name (Passive Detection)
 | Confirmed By:
 |
 |  Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 |
[+] bobo
 | Found By: Author Posts - Display Name (Passive Detection)
 | Confirmed By:
 |
 |  Author Id Brute Forcing - Author Pattern (Aggressive Detection)
```

Crack passwords for admin and bobo To crack the passwords, we used `rockyou.txt` as the wordlist.

```
$ wpscan --url http://3.145.206.165 --passwords rockyou.txt --usernames admin,bobo
```

`admin`'s password was not cracked, but we got bobo's password: `Football`

```
[+] Performing password attack on Wp Login against 2 user/s
[SUCCESS] - bobo / Football
Trying admin / contraviento Time: 04:24:04 <== > (689415 / 28702202) 2.40% ETA: ??:??:??
[!] Valid Combinations Found:
| Username: bobo, Password: Football
```

Using username as `bobo` and Password as `Football`, we successfully logged in `http://3.145.206.165/wp-login.php`.

And we found the flag in Dashboard:

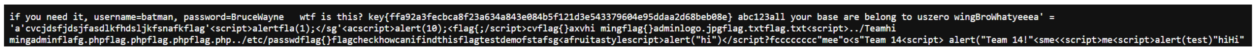
Challenge 11: XSS gone sinister

Problem

A little bit of sneaky XSS will get us there.

Solution

Screenshot of flag



```
if you need it, username=batman, password=BruceWayne   wtf is this? key{ffa92a3fecbca8f23a634a843e084b5f121d3e543379604e95ddaa2d68beb08e} abc123all your base are belong to uszero wingBroWhatyeeea' =
'a'cvcjdsfjdsjfasdlkfhdsljkfsnafkflag'<script>alertfla(1);</sg'<acscript>alert(10);<flag{;/script>cvflag{}axvhi mingflag{}adminlogo.jpgflag.txtflag.txt<script>../Teamhi
mingadminflafg.phpflag.phpflag.phpflag.php../etc/passwdflag{}flagcheckhowcanifindthisflagtestdemofstafsg<afruitastylescript>alert("hi")</script?fcccccccc"mee"o<s"Team 14<script> alert("Team 14!"<sme<<script>me<script>alert(test)"hiHi"
```

Figure 9: data.txt

Exact Location Key found at this URL: http://3.145.206.165/data.txt

Method Inspected the board.php page and found there was a code snippet in the first comment. Compared to all the other comments, it looked strangely out of place... so doing what a curious person should do: we dug deeper to figure out what exactly it did. After some digging, we concluded it was used to log user keystrokes!

After understanding that, we tested going to http://3.145.206.165/logger.php, since that's where it looked like everything was being directed to. To our surprise, we were met with a blank page didn't find anything noteable until inspecting the page... information on **JohnHoder's Javascript Keylogger GitHub**.

After finding the GitHub, we saw some information on how the logger worked and where it was sesnding information to be stored. But to where? Process of elimination... If it's not the `.js` or `data.php` file, it's probably the `data.txt` file right?

Lo and behold - after plugging in http://3.145.206.165/data.txt, the key was listed there (along with a lot of other information input which we assume is from users submitting information to the `board.php` page.)

A fun goose chase to go from one point to another!

Challenge 12: Where are the robots?

Problem

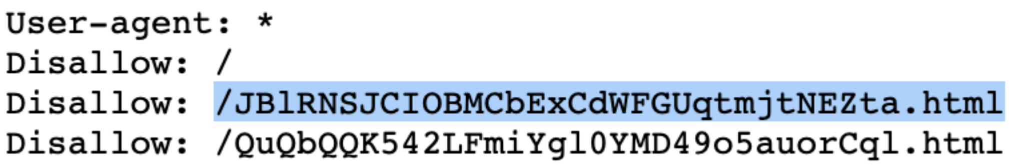
These are not the droids you're looking for...

Solution

Gobuster helped me discovered `robots.txt`

```
/readme.html          (Status: 200) [Size: 7466]
/robots.txt           (Status: 200) [Size: 8135]
/wp-trackback.php     (Status: 200) [Size: 135]
```

On the `robots.txt` page, we found two disallowed paths listed.



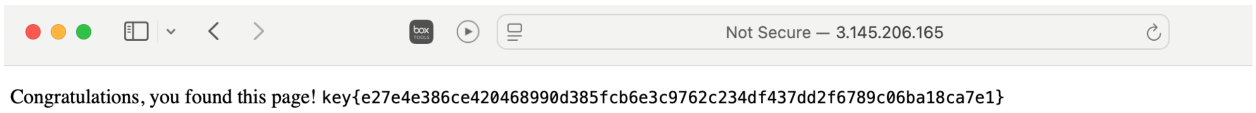
```
User-agent: *
Disallow: /
Disallow: /JBlRNSJCIOBMCbExCdWFGUqtmjtNEZta.html
Disallow: /QuQbQQK542LFmiYgl0YMD49o5auorCql.html
```

Figure 10: robots.txt

The first one let us to the flag, and the other one helped me locate the notuber page.

http://3.145.206.165/JBlRNSJCIOBMCbExCdWFGUqtmjtNEZta.html

Found the key `key{e27e4e386ce420468990d385fcb6e3c9762c234df437dd2f6789c06ba18ca7e1}`



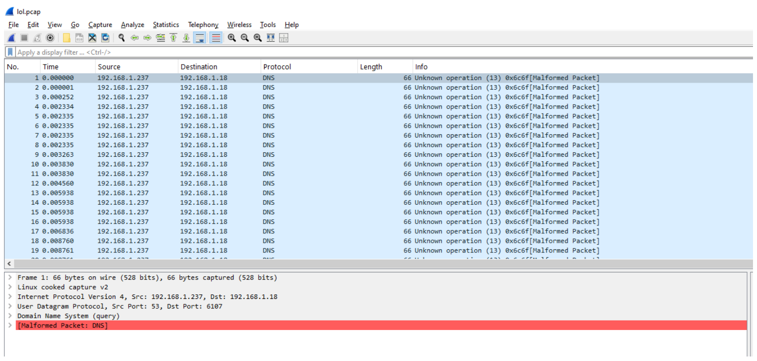
Not Secure — 3.145.206.165

Congratulations, you found this page! `key{e27e4e386ce420468990d385fcb6e3c9762c234df437dd2f6789c06ba18ca7e1}`

Figure 11: robots_key

Challenge 13: LOLCAP (Not Found)

Problem



lol.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time       Source          Destination    Protocol  Length  Info
   1 0.000000   192.168.1.237   192.168.1.18   DNS           66 Unknown operation (13) 0x6c6f[Malformed Packet]
   2 0.000001   192.168.1.237   192.168.1.18   DNS           66 Unknown operation (13) 0x6c6f[Malformed Packet]
   3 0.000252   192.168.1.237   192.168.1.18   DNS           66 Unknown operation (13) 0x6c6f[Malformed Packet]
   4 0.002334   192.168.1.237   192.168.1.18   DNS           66 Unknown operation (13) 0x6c6f[Malformed Packet]
   5 0.002335   192.168.1.237   192.168.1.18   DNS           66 Unknown operation (13) 0x6c6f[Malformed Packet]
   6 0.002335   192.168.1.237   192.168.1.18   DNS           66 Unknown operation (13) 0x6c6f[Malformed Packet]
   7 0.002335   192.168.1.237   192.168.1.18   DNS           66 Unknown operation (13) 0x6c6f[Malformed Packet]
   8 0.002335   192.168.1.237   192.168.1.18   DNS           66 Unknown operation (13) 0x6c6f[Malformed Packet]
   9 0.003263   192.168.1.237   192.168.1.18   DNS           66 Unknown operation (13) 0x6c6f[Malformed Packet]
  10 0.003830   192.168.1.237   192.168.1.18   DNS           66 Unknown operation (13) 0x6c6f[Malformed Packet]
  11 0.003830   192.168.1.237   192.168.1.18   DNS           66 Unknown operation (13) 0x6c6f[Malformed Packet]
  12 0.004560   192.168.1.237   192.168.1.18   DNS           66 Unknown operation (13) 0x6c6f[Malformed Packet]
  13 0.005938   192.168.1.237   192.168.1.18   DNS           66 Unknown operation (13) 0x6c6f[Malformed Packet]
  14 0.005938   192.168.1.237   192.168.1.18   DNS           66 Unknown operation (13) 0x6c6f[Malformed Packet]
  15 0.005938   192.168.1.237   192.168.1.18   DNS           66 Unknown operation (13) 0x6c6f[Malformed Packet]
  16 0.005938   192.168.1.237   192.168.1.18   DNS           66 Unknown operation (13) 0x6c6f[Malformed Packet]
  17 0.006836   192.168.1.237   192.168.1.18   DNS           66 Unknown operation (13) 0x6c6f[Malformed Packet]
  18 0.008760   192.168.1.237   192.168.1.18   DNS           66 Unknown operation (13) 0x6c6f[Malformed Packet]
  19 0.008761   192.168.1.237   192.168.1.18   DNS           66 Unknown operation (13) 0x6c6f[Malformed Packet]

> Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
> Linux cooked capture v2
> Internet Protocol Version 4, Src: 192.168.1.237, Dst: 192.168.1.18
> User Datagram Protocol, Src Port: 53, Dst Port: 6107
> Domain Name System (query)
> [Malformed Packet: DNS]
```

Figure 12: Wireshark lol.pcap

Download the LOLCAP.pcap file and find flag Packets in pcap file are all malformed

Solution

Wireshark We followed the UDP stream, and got the text in ASCII format:

lol wtflol wtflol wtflol wtflol wtflol wtflol wtflol wtflol wtflol wtflol wtflol wtflol wtflol wtflolwtflol wtf

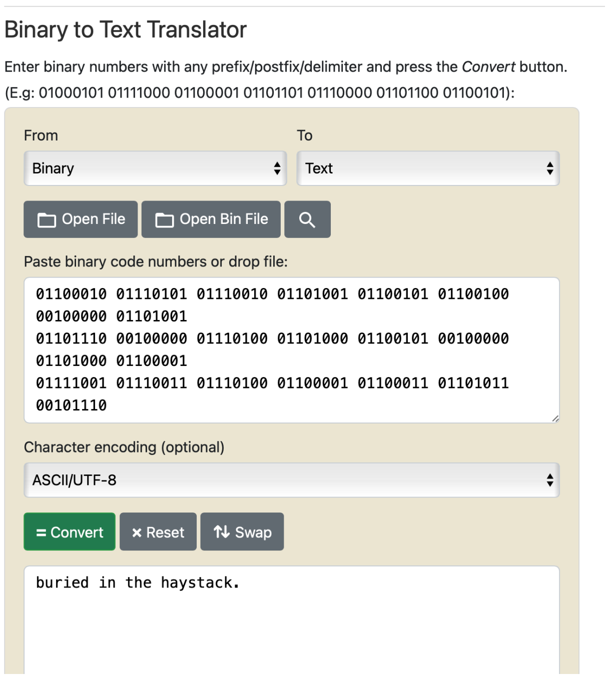
From the text, we observed two recurring strings: **lol** and **wtflol**. This suggested a potential binary encoding. experimented by replacing **lol** with **0** and **wtflol** with **1**, as well as the reverse — replacing **lol** with **1** and **wtflol** with **0**.

After converting the resulting binary data to ASCII text, we found that using **lol** = **0** and **wtflol** = **1** produced meaningful output.

Here is the binary data obtained using that mapping:

```
01100010 01110101 01110010 01101001 01100101 01100100 00100000 01101001
01101110 00100000 01110100 01101000 01100101 00100000 01101000 01100001
01111001 01110011 01110100 01100001 01100011 01101011 00101110
```

We thought we got the key, but the result is disappointing, because the decoded text is `buried in the haystack.`



Binary to Text Translator

Enter binary numbers with any prefix/postfix/delimiter and press the *Convert* button.
(E.g: 01000101 01111000 01100001 01101101 01110000 01101100 01100101):

From: Binary
To: Text

Paste binary code numbers or drop file:

```
01100010 01110101 01110010 01101001 01100101 01100100
00100000 01101001
01101110 00100000 01110100 01101000 01100101 00100000
01101000 01100001
01111001 01110011 01110100 01100001 01100011 01101011
00101110
```

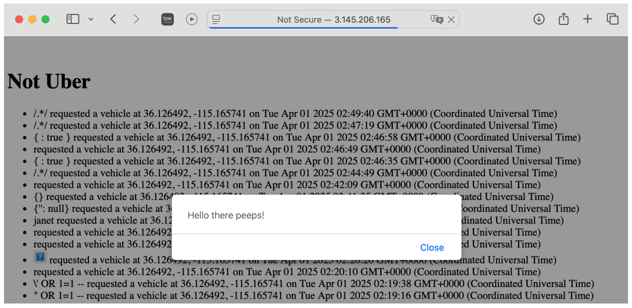
Character encoding (optional): ASCII/UTF-8

```
buried in the haystack.
```

Figure 13: Binary to ASCII

Challenge 14: notuber (Not Found)

From `robots.txt`, we found the notuber page. We can also see some students already conducted XSS attacks to this page. We tried similar way, unfortunately, we did not get the key.



Not Secure — 3.145.206.165

Not Uber

- /.*/ requested a vehicle at 36.126492, -115.165741 on Tue Apr 01 2025 02:49:40 GMT+0000 (Coordinated Universal Time)
- /.*/ requested a vehicle at 36.126492, -115.165741 on Tue Apr 01 2025 02:47:19 GMT+0000 (Coordinated Universal Time)
- { : true } requested a vehicle at 36.126492, -115.165741 on Tue Apr 01 2025 02:46:58 GMT+0000 (Coordinated Universal Time)
- requested a vehicle at 36.126492, -115.165741 on Tue Apr 01 2025 02:46:49 GMT+0000 (Coordinated Universal Time)
- { : true } requested a vehicle at 36.126492, -115.165741 on Tue Apr 01 2025 02:46:35 GMT+0000 (Coordinated Universal Time)
- /.*/ requested a vehicle at 36.126492, -115.165741 on Tue Apr 01 2025 02:44:49 GMT+0000 (Coordinated Universal Time)
- requested a vehicle at 36.126492, -115.165741 on Tue Apr 01 2025 02:42:09 GMT+0000 (Coordinated Universal Time)
- {} requested a vehicle at 36.126 [illegible] nated Universal Time)
- {'': null} requested a vehicle at 3 [illegible] oordinated Universal Time)
- janet requested a vehicle at 36.12 [illegible] dinated Universal Time)
- requested a vehicle at 36.126492 [illegible] ed Universal Time)
- requested a vehicle at 36.126492 [illegible] ed Universal Time)
- requested a vehicle at 36.126 [illegible] nated Universal Time)
- requested a vehicle at 36.126492, -115.165741 on Tue Apr 01 2025 02:20:10 GMT+0000 (Coordinated Universal Time)
- \' OR 1=1 -- requested a vehicle at 36.126492, -115.165741 on Tue Apr 01 2025 02:19:38 GMT+0000 (Coordinated Universal Time)
- " OR 1=1 -- requested a vehicle at 36.126492, -115.165741 on Tue Apr 01 2025 02:19:16 GMT+0000 (Coordinated Universal Time)

Hello there peeps!

Figure 14: Challenge 14

Lessons Learned

- There are many public tools available that are helpful for scraping and detecting vulnerabilities (Cyberchef, Gobuster, etc.)
- It's very difficult to be completely invulnerable. . . This lab is setup to have varying levels of vulnerabilities, but as we learned from the examples discussed about in class: schools, corporations, and even encryption companies are not invulnerable.
- **ALWAYS VALIDATE USER PROVIDED INPUT**
- Exploring the vulnebilities in this website helped us understand how simple design flaws can lead to serious security risks. Attackers can take over the server so easy, such as the message board.
- (Joel)
  - Inspecting the elements of a webpage has TONS of information - super cool to actually dig in to

and see what lies underneath the hood!

- (Yucheng)
  - Collaboration is important in solving ctf challenges and working in cybersecurity. Each memeber of our team comes from different background and have different set of skills. Looking at the challenges from different angles help us figure out solution much more efficient.
- (Chris)
  - Using common file names that are easily guessable can be a vulnerability. But boy does it feel good to find a vulnerability!
  - For quite a few flags, if we locate the hidden pages, the flags are revealed. Hiding pages does not mean they are secure, we have to make sure they are not exposed to attackers.

Conclusions

- While the challenges seemed difficult, most of the challenges we solved were just extensions of what we had learned in class and required some out-of-the-box thinking.
- (Joel) While we didn't get any of the 400 point flags, they all were "fair" - nothing out of the ordinary but it either took some extra steps or a modified approach to get it right. In my opinion, it feels like some of it just comes from practice and seeing more and more examples to identify what the vulnerability is and exploiting it… I will say that not getting any of the 400 point flags was (and still is) frustrating. Nevertheless, CTF's are a great way to hone our cybersecurity knowledge!
- (Yucheng) Playing CTF challanges is a very efficient approach to improve our cybersecurity skills. Security vulnerabilites can exist in unexpected places. We have to pay attention to details. **Weak password is a serious security risk**.
- (Chris) Definitely had a blast testing out the skills we've learned in class a very interactive way to learn from classmates and hone our skills. Strong passwords are definitely a must who knows what that admin password was.

What Would You Do Differently For Next Time?

- (Joel) I definitely would search for more available tools! Yucheng introduced me to Gobuster to find the available files, so I'm curious to know what other tools there are. I also spent of lot of unnecessary time by overcomplicating things, when in reality they were a lot simpler (like decrypting multiple times in a row). For the future, I'd probably go with my gut instinct to try simple things first before making it more complicated. More of a personal thing: I also would love to be able to do this in person! While online is great for remote students, I'd love to be able to work in-person with folks tackling the same problems. I think it changes the dynamic and makes it more fun and collaborative. And most of all, like being thrown into the deep end of a pool to learn how to swim… there's no better way to learn than from hands-on experience.
- (Yucheng) I learnt that playing ctf and working in cybersecurity require us to have a broad range of skills. We need to understand how website, network, and database work. In our team, we realized none of us had strong knowledge in areas such as javascript, sql and web development. This might be the reason why we were not able to solve the 400-point challenges. We will improve our knowledge and skills in these fields. I also found that even if I learnt wireshark and burpsuite from class, my skills are not enough for me to solve real-world challenges. This requires me to have a deeper understanding and more hands-on practice.
- (Chris) CTF games are very involved and there are so many tools to help find vulnerabilities. For the most part I think the more experience you have the better you are at finding the vulnerabilities. So I would definitely like to learn more ways to find vulnerabilities or interpret vulnerabilities in JavaScript code.
- (Jackson) Through the ctf game, which reinforced many of the same concepts and tools we learnt in class, I came to appreciate that identifying/protecting against/exploiting cybersecurity issues require a very particular lense and familiarity with very particular concepts that take a lot of practice. And it's sometimes more efficient to first know and familiarize yourself with the issues and tricks before starting to solve the challenges, as opposed to sitting in front of the computer and just try to spin wheels. I

would definitely go through the tricks and tools we learnt from class one more time if I could do this again before starting to work on the problems.