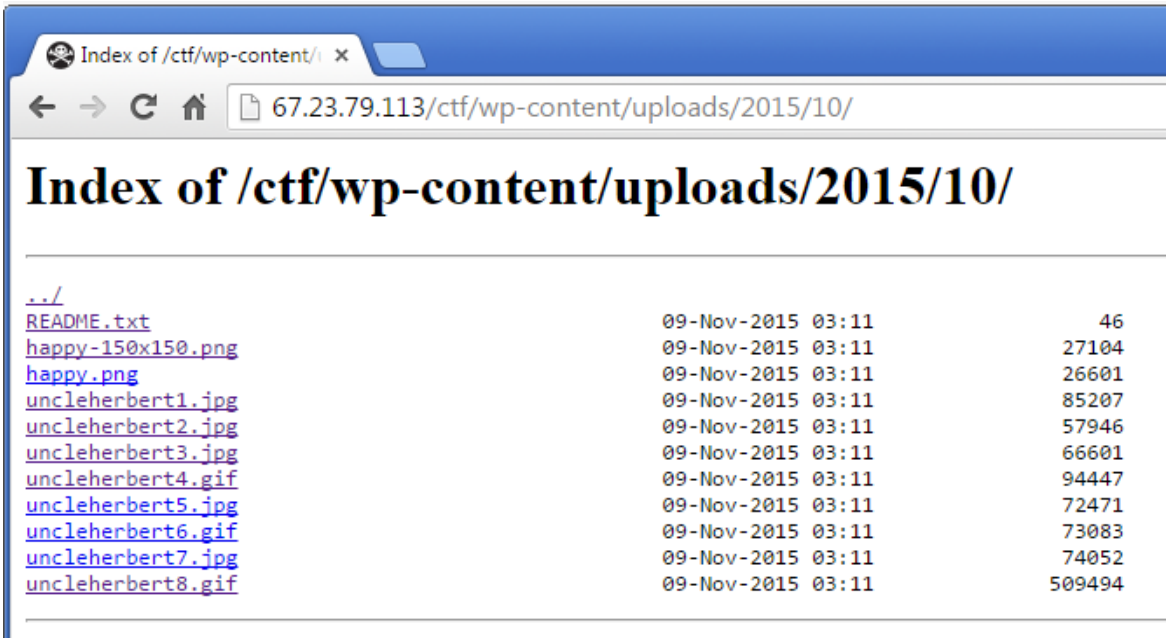
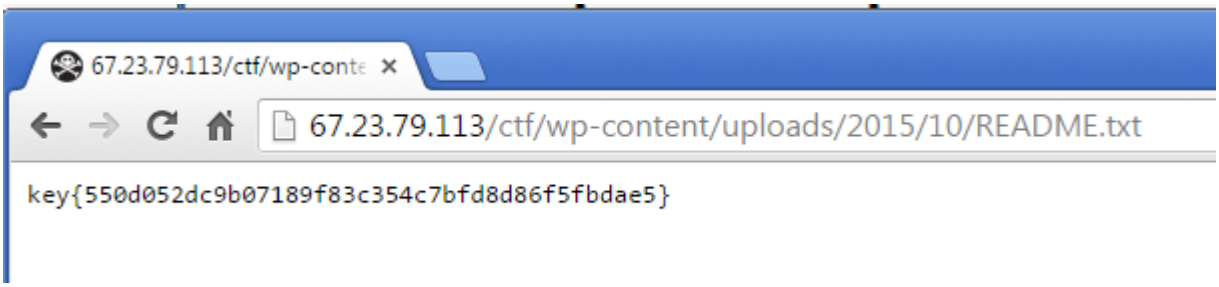
Comp 116, Assignment 3: Capture The Flags Write-Up

Team 13: Phillip Braunstein, Rachel Hogue, Shawyoun Shaidani, Mary Matthews

Location: http://67.23.79.113/ctf/wp-content/uploads/2015/10/README.txt

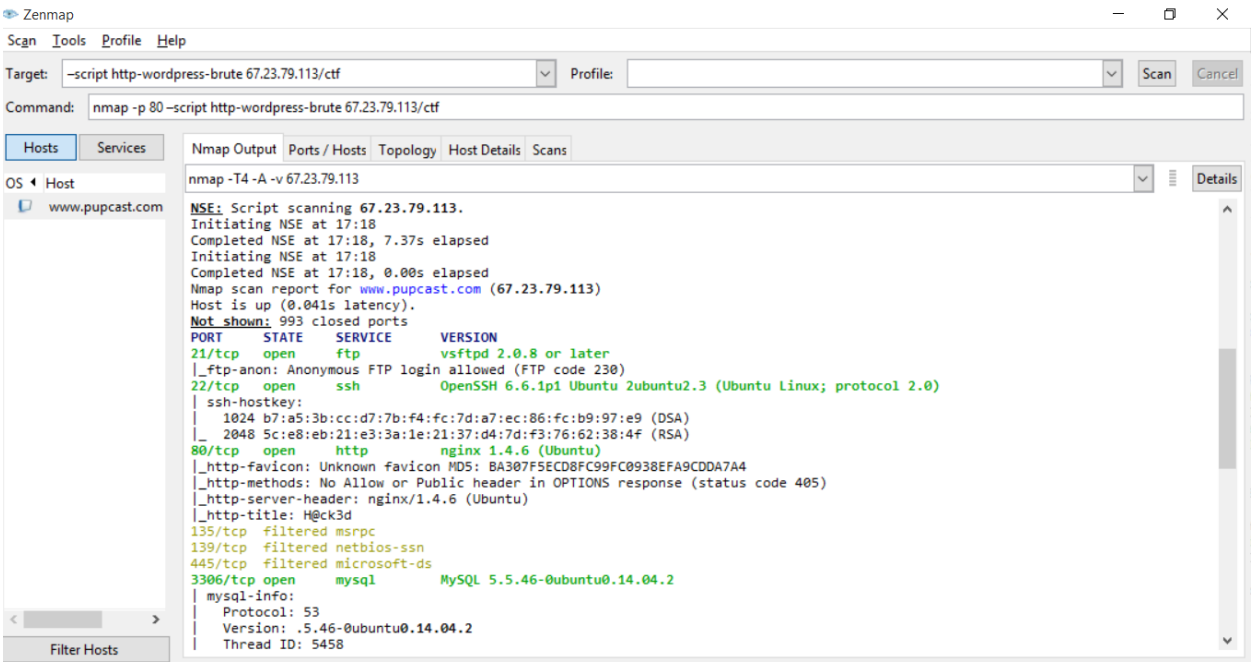
Exploit: Basic Wordpress vulnerability (one of many). Ming highlighted that the directory with media assets is publicly accessible by default. We found this simply by accessing the folder and poking around.

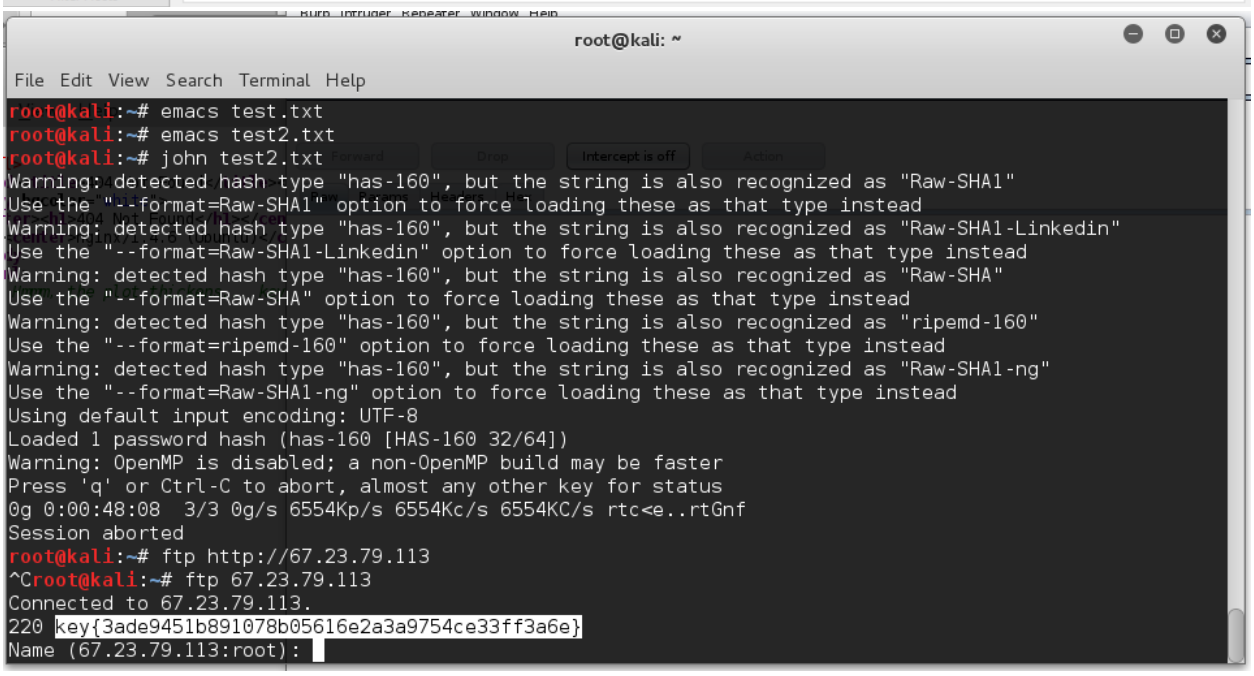




Location: The FTP login greeting banner at 67.23.79.113: 21

Exploit: We performed an nmap scan on the provided IP and tried accessing the services we found. Anonymous login was enabled on FTP, which looked suspicious.





Location: http://67.23.79.113/ctf/runme.exe

Exploit: We ran “strings” against the binary file. That gave us the message that we needed to “watch the video” to find a number, and then SHA-1 the number. We performed a sort of “parallel computing” by designating one teammate to analyze the packets of the Sesame Street video using Wireshark, and designating another teammate “brute-force” the key by entering “ONE”, “TWO”, “THREE”.... into a online SHA-1 tool. The brute-force person won by using the SHA-1 digest of the word “SEVEN”. While we were not successful in downloading the actual video, this clue was relatively exploitable via a social engineering pathway since we could see that the video was a sesame street video, and a number in a sesame street video couldn’t be too high, so it was feasible to brute force it as described above. The video we tried to download (based on the 200 get request found in the packet) was at 192.168.1.8/Movies/sesamestreet.mp4

```
w5Vm
w5VT
w5VX
GJ9^
w5VL
iWatch the video. The key is the SHA1 sum of the number, as a word in all cap
s, in the video.
4JP@
GJ:^
@JQ@
GJ:^
w5V@
.JR@
GJ:^
HTTP/1.1 400 Bad Request
Date: Sun, 01 Nov 2015 02:24:50 GMT
Server: Apache/2.2.22 (Debian)
Vary: Accept-Encoding
Content-Length: 301
Connection: close
Content-Type: text/html; charset=iso-8859-1
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
```

SHA1 online

www.sha1-online.com

Home Page | SHA1 in JAVA | Secure password generator | Linux

SHA1 and other hash functions online generator

SEVEN hash

sha-1

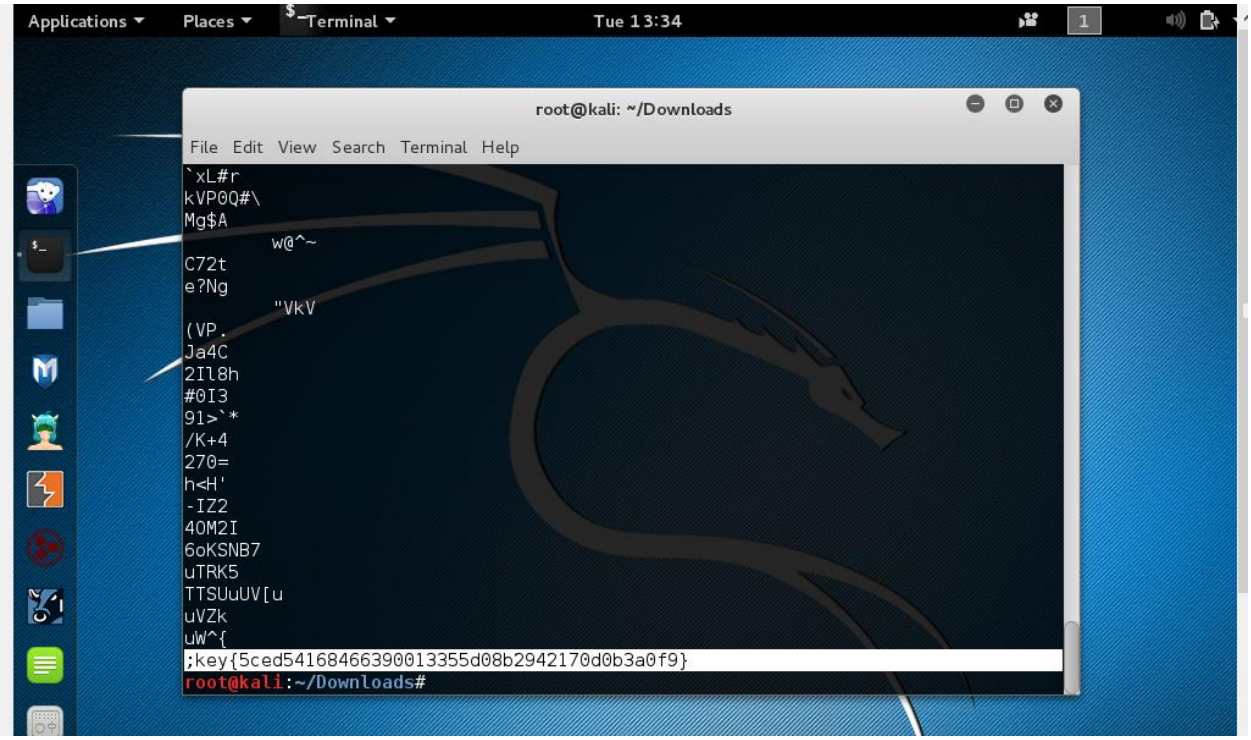
Result for sha1: cabd534c35ee6a39365f4ed3bce4eafdcc3d4b8d

Location: 67.23.79.113/ctf/board.php/crying.gif

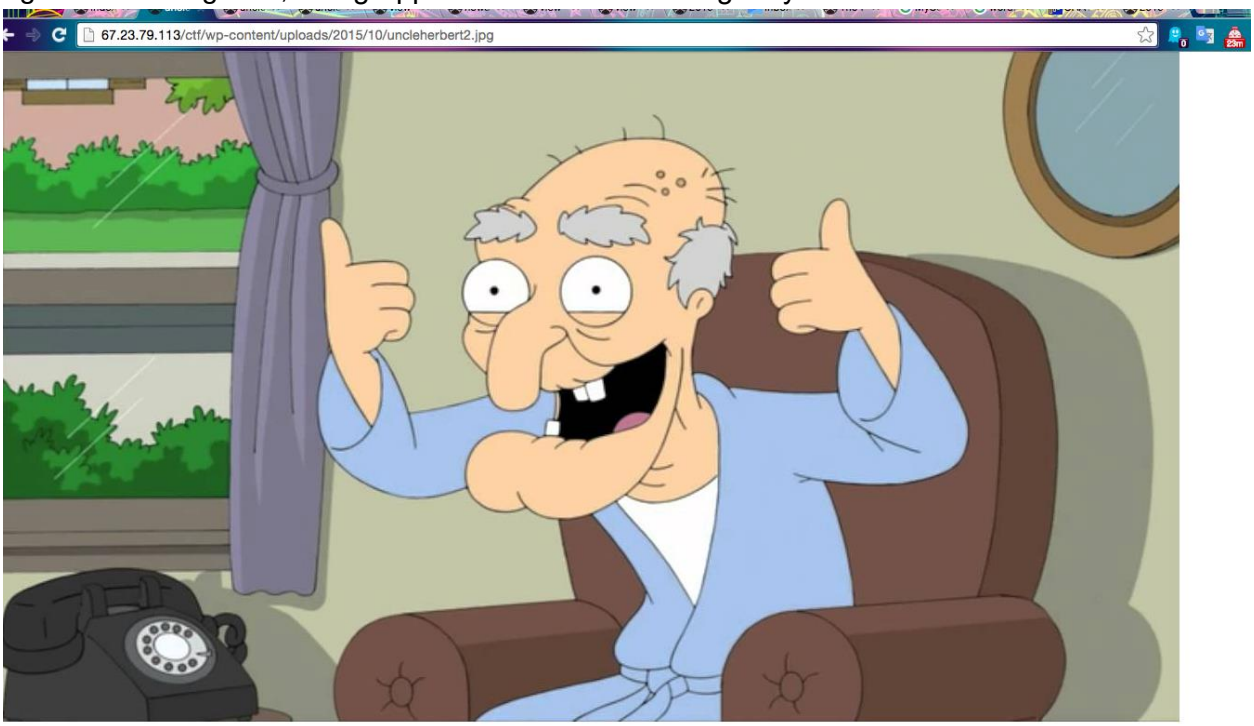
Exploit: We started at board.php. You can manipulate the returned set of posts with different values in the id parameter of the request; the important posts could be visible without specifying any value for that parameter. Originally, you'd get bombarded with a ton of javascript alerts and eventually get redirected to 4chan. Either using view-source, a javascript blocker or OWASP ZAP (simple pen-testing tool for web sites), you can examine the source of all the returned posts, which shows that some of them contain javascript calls, which were responsible for the alerts and redirect. A few of them were also performing DOM manipulation, and one in particular was replacing the content of the div with id of “logo”. The crying picture was the original content of that div, but was getting removed, which seemed suspect. We were able to find the key by downloading the picture and then using the strings command.

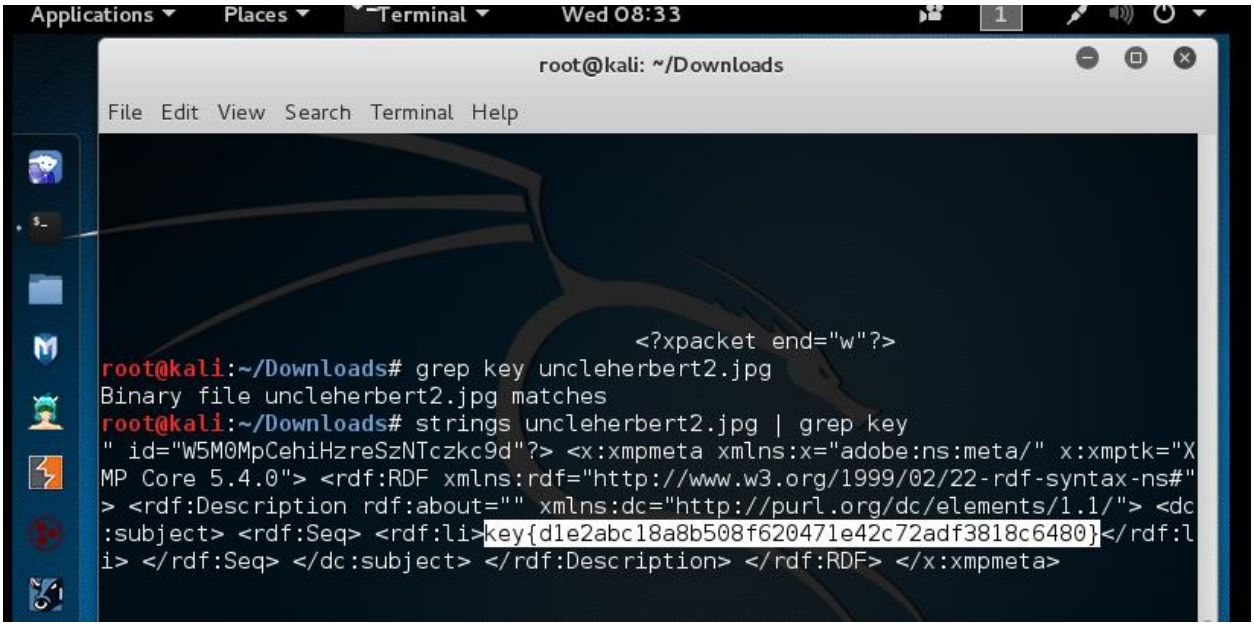
```
<!DOCTYPE>
<head>
     <head>
          <title>The Happening</title>
          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
          <link rel="alternate stylesheet" type="text/css" href="site_burichan.css" title="Burichan" />
          <link rel="alternate stylesheet" type="text/css" href="site_futaba.css" title="Futaba" />
          <link rel="stylesheet" type="text/css" href="site_kusabax.css" title="Kusabax" />
     </head>

     <body>

     <div id="header"><h1><img id="logo" src="crying.gif" /><br/>The Happening</h1></div><form id="posting" method="post"><h4>New
Post</h4><p>Title: <input type="text" name="title" /></p><p>Post: <textarea name="post"></textarea></p><p><input type="submit"></p>
</form><h2><a href="board.php?id=389">ZAP</a></h2>
<p></p><script>alert(1);</script><p></p>
<h2><a href="board.php?id=388">ZAP</a></h2>
<p>0W45pz4p</p>
<h2><a href="board.php?id=387">ZAP</a></h2>
<p>zApPX15sS</p>
<h2><a href="board.php?id=386">ZAP</a></h2>
<p>|</p>
<h2><a href="board.php?id=385">ZAP</a></h2>
```

```
The piper's calling you to join him
Dear lady can you hear the wind blow and did you know
Your stairway lies on the whispering wind

And as we wind on down the road
Our shadows taller than our souls
There walks a lady we all know
Who shines white light and wants to show
How everything still turns to gold
And if you listen very hard
The tune will come to you at last
When all are one and one is all, yeah
To be a rock and not to roll
Ooooooooooooh

And she's buying a stairway to heaven</p>
<h2><a href="board.php?id=1">Welcome to the 2015 CTF</a></h2>
<p>Thrash away!<script>document.getElementById("logo").src="logo.png"</script></p>
     <div id="footer">
          <hr/>
          <h3><a href="board.php">Home</a> | <a href="admin.php">Administration</a></h3>
     </div>
     </body>
</head>
```

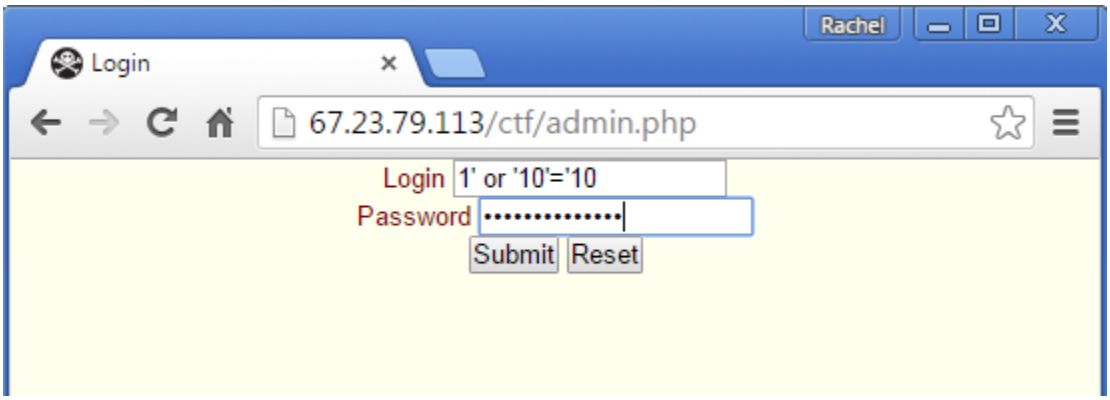

Applications
Places
Terminal
Tue 13:34
root@kali: ~/Downloads
File Edit View Search Terminal Help
`xL#r
kVP0Q#\
Mg$A
w@^~
C72t
e?Ng
"VkV
(VP.
Ja4C
2Il8h
#0I3
91>`*
/K+4
270=
h<H'
-IZ2
40M2I
6oKSNB7
uTRK5
TTSUuUV[u
uVZk
uW^{
;key{5ced5416846639001335 5d08b2942170d0b3a0f9}
root@kali:~/Downloads#

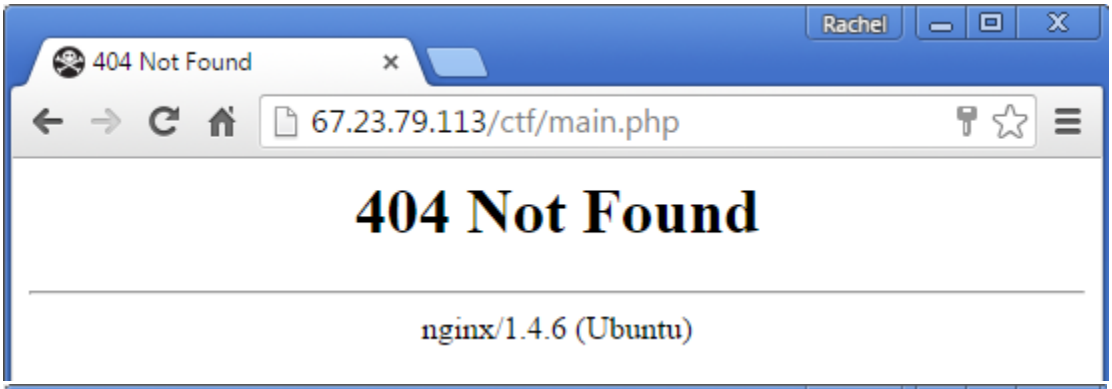
Location: 67.23.79.113/ctf/wp-content/uploads/2015/10/uncleherbert2.jpg
Exploit: Using the same basic exploit as the one used to get the README file (that is, directly accessing the wordpress contents via URL), but with an addition: we ran the “strings” command against the image file, and grepped it to match the string “key.”

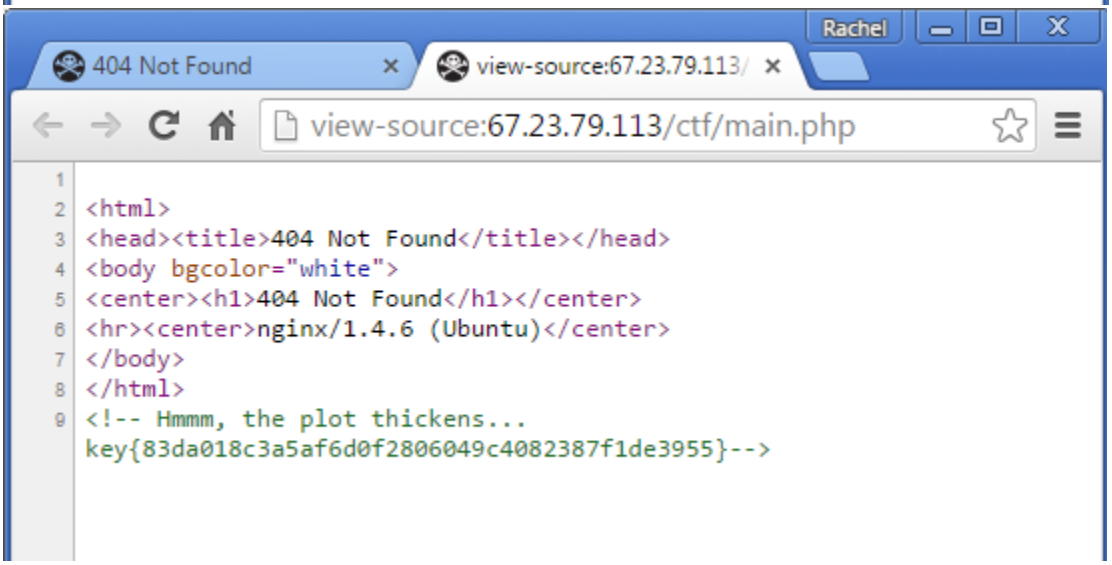




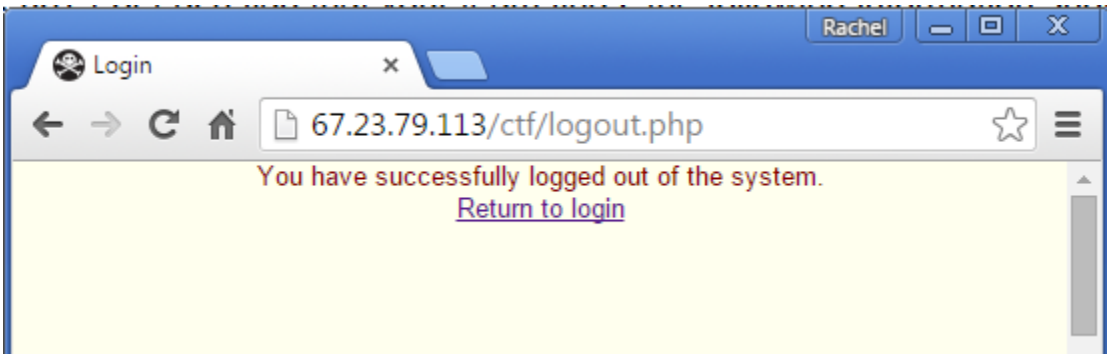
Location: 67.23.79.113/ctf/main.php
Exploit: SQL injection. We used it to force a query that would always return true, so even though we lacked credentials, we were able to bypass the login screen. The key was in the source of the error page we found.







Location: 67.23.79.113/ctf/logout.php
Exploit: Then we tried a standard logout URL and found the second key in the source code of that page (after approximately a million breakline tags).



Location: 67.23.79.113/ctf/?p=33&preview=true

Exploit: This was a weak password (supermodel). We were able to use a dictionary attack in order to gain access to the blog and then the key was in one of the drafts. One of the tools we used wasn't very successful (http-wordpress-brute, a script that comes with nmap). Another tool, wpscan, gave us a list of users (although we had already been given the username bobo from the blog source code and hints from Ming), and then we ran a dictionary attack against that user account. The exact command used was "wpscan --url 67.23.79.113/ctf --wordlist /root/Downloads/10_million_password_list_top_1000000.txt --username bobo".