

Web 1 Write-Up

URL Encoding

10 points

Problem:

Solve the following problem using Python; you need to submit the flags and your programs for full credit.

Decode this URL encoded string:

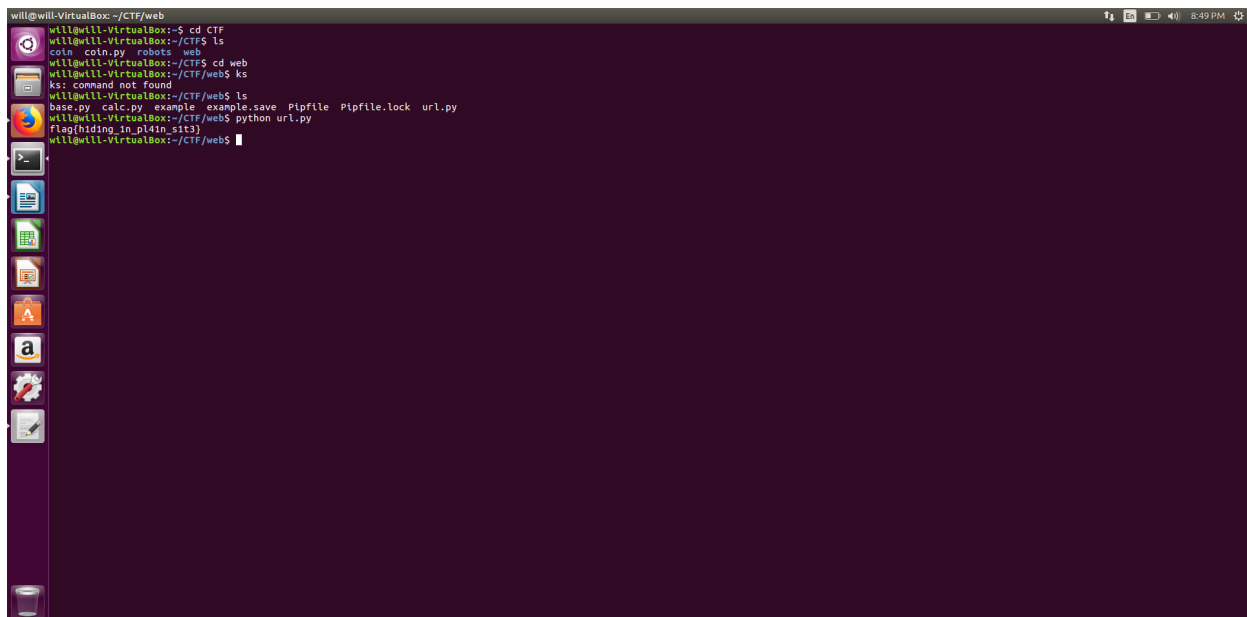
```
%66%6c%61%67%7b%68%31%64%31%6e%67%5f%31%6e%5f%70%6c%34%31%6e%5f%73%31%74%33%7d
```

Solution:

To decode URL encoded strings it is necessary to import the urllib library into the python script. Strings that are URL encoded are decoded with `urllib.unquote()`. Printing the decoded string was all that was necessary to get the flag.

```
import urllib
print urllib.unquote("%66%6c%61%67%7b%68%31%64%31%6e%67%5f%31%6e%5f%70%6c%34%31%6e%5f%73%31%74%33%7d")
```

The program is url.py.



The screenshot shows a terminal window titled 'will@will-VirtualBox: ~/CTF/web'. The user has navigated to the 'CTF' directory and listed files, including 'url.py'. They then run 'python url.py', which outputs the decoded string: 'flag(hiding_in_pl4in_s1t3)'.

```
will@will-VirtualBox:~/CTF/web$ cd CTF
will@will-VirtualBox:~/CTF$ ls
cota  coin.py  robots  web
will@will-VirtualBox:~/CTF$ cd web
will@will-VirtualBox:~/CTF/web$ ls
ks: command not found
will@will-VirtualBox:~/CTF/web$ ls
base.py  calc.py  example  example.save  Pipfile  Pipfile.lock  url.py
will@will-VirtualBox:~/CTF/web$ python url.py
flag(hiding_in_pl4in_s1t3)
will@will-VirtualBox:~/CTF/web$
```

Base 64

10 points

Problem:

Solve the following problem using Python; you need to submit the flags and your programs for full credit.

Do a while loop to decode this base64 string until you get the flag:

```
Vm0xd1NtUXlWa1pPVldoVFlUSlNjRlJVVGtOamJGWnhVMjA1VlUxV2NIbFdiVEZlWVZaYWRR  
W  
RnNhRmRXtTfKUVZrZDRxbVF3TlZsalJsWk9WakZLTmxaclVrZFVNVXB5VFZaV1dHSkhhRlJ  
W  
YkZwM1ZGWlpIVTFVVW1wTmF6VllWbGMxVjFaWFJqWldiRkpoVmpOb2FGUldXbHBrTWtaSl  
drWlNUbGRGU2paV2FrbzBZekZhV0ZKdVVtcGxiWE01
```

Solution:

This question is similar to URL Encoding. However, I did have some problems with this challenge. I had trouble copying over the new lines and spaces from the website into my python script. Because of this my program would not work. Once I solved that issue my script worked.

First I imported the base64 python library. I then started a while loop. Inside of the while loop I did a standard `base64.b64decode(the_encoded_string)`. In python documentation that what is required to decode a base64 encoded string. Knowing that the flag contained “flag” I searched for “flag” using `find()`. Once I found an occurrence of flag I printed the decoded message inside the while loop and ended the while loop.

The program is base.py

```
will@will-VirtualBox:~/CTF/web
will@will-VirtualBox:~/CTF/web$ cd CTF
will@will-VirtualBox:~/CTF$ ls
cat.py  cat.py  robots  web
will@will-VirtualBox:~/CTF$ cd web
will@will-VirtualBox:~/CTF/web$ ls
ks: command not found
will@will-VirtualBox:~/CTF/web$ ls
base.py  calc.py  example  example.save  Pipfile  Pipfile.lock  url.py
will@will-VirtualBox:~/CTF/web$ python url.py
flag[hiding_in_plain_sight]
will@will-VirtualBox:~/CTF/web$ python base64.py
python: can't open file 'base64.py': [Errno 2] No such file or directory
will@will-VirtualBox:~/CTF/web$ python base.py
flag[all_your_base64_strings_2_us]
will@will-VirtualBox:~/CTF/web$
```

Calculator

20 points

Problem:

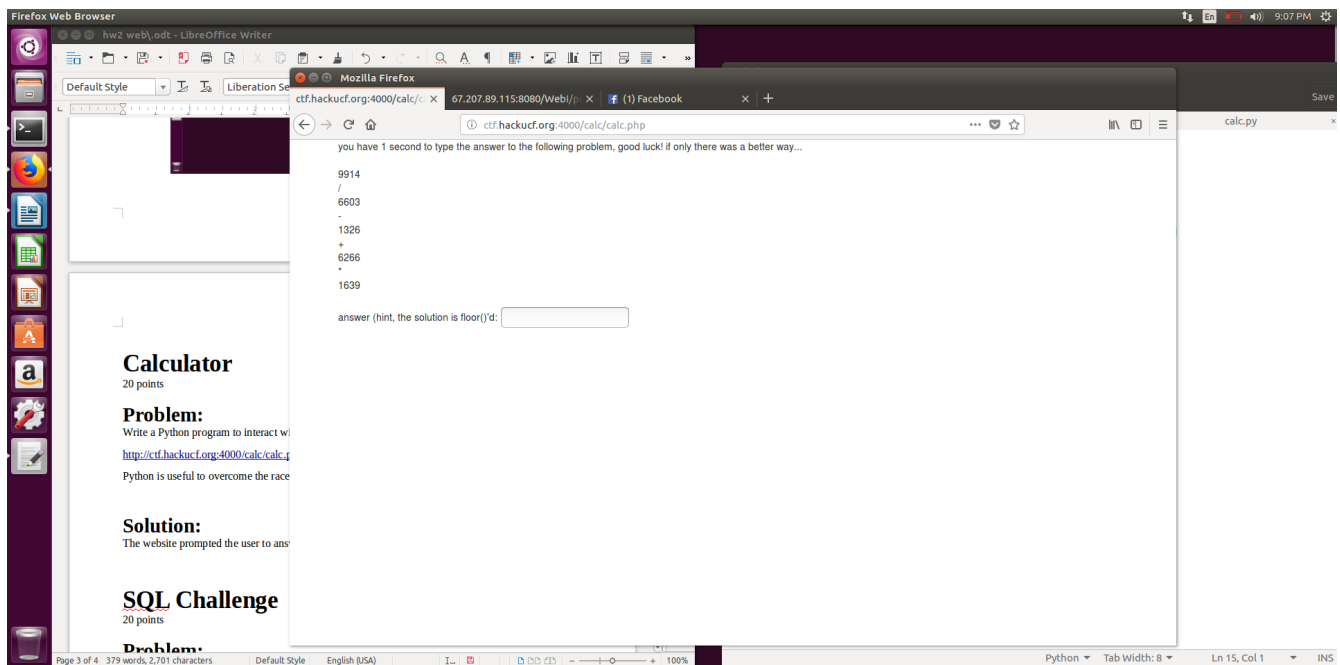
Write a Python program to interact with the following website to get the flag:

<http://ctf.hackucf.org:4000/calc/calc.php>

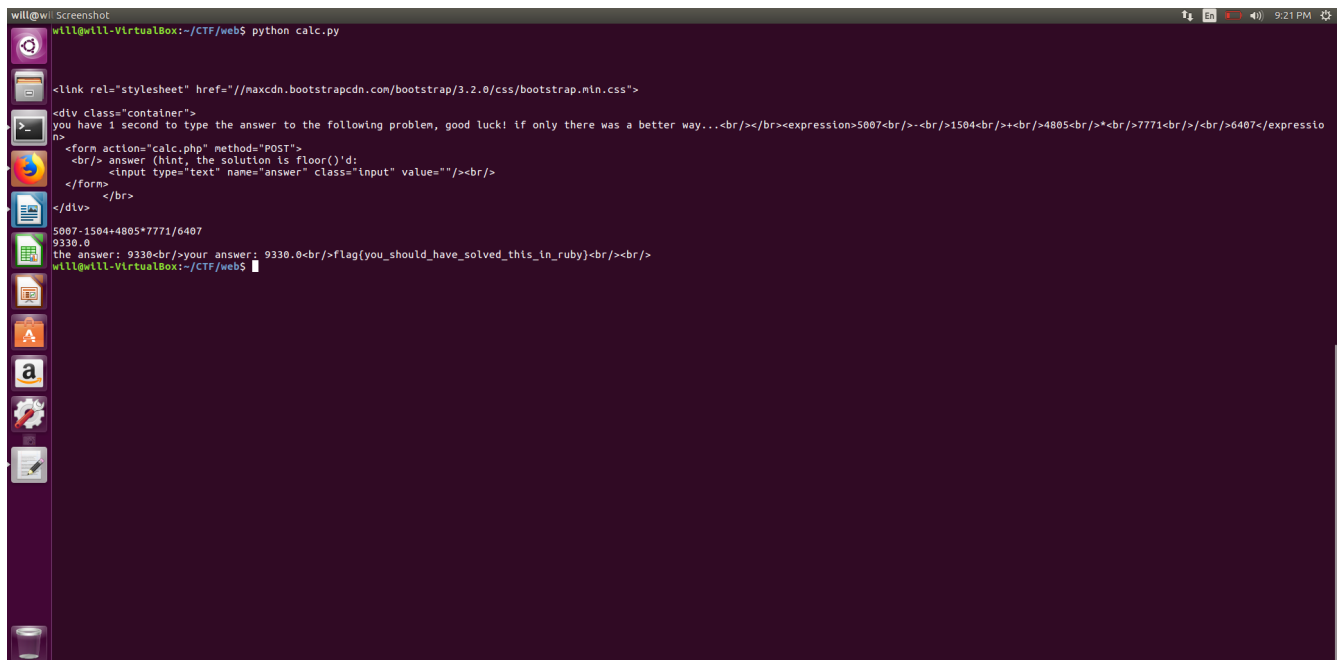
Python is useful to overcome the race condition. For HTTP requests, use the Python requests library.

Solution:

The website prompted the user to answer a math problem in 1 second as follows:



To find the flag I created a python script. Using the requests library and get(url) I could interact with the website. I used request.text to get the server's response to the get() request which showed the HTML of the website.



Displays the program running with the website HTML and flag displaying.

I broke up the text into a new variable using `[thing.rfind("<expression>"):thing.rfind("</expression>")]`. I did this because the HTML of the expression was located between those two expression brackets. After I trimmed the HTML I used `replace()` to get rid of all "`<expressions>`" and "`
`" in the HTML and replace them with empty strings. I did this to get only the math expression by itself. To evaluate the expression I used `eval()` from the `math` library and then the `floor` function. In order to use proper syntax with `request()` to send my answer to the server I had to use `ans={'answer':ans}`. 'answer' is to match

the syntax of the post request that the server is expecting. I found the correct syntax in the HTML. After setting the cookies, I sent the answer back using `ResponseB=requests.post()`. Through printing data of the post request back to the server using `responseB.text` we could see the flag. Unfortunately it only works about $\frac{3}{4}$ of the time for some reason. I found the answer correct on the first try and that is how I knew that it worked.

The program is `calc.py`.

SQL Challenge

20 points

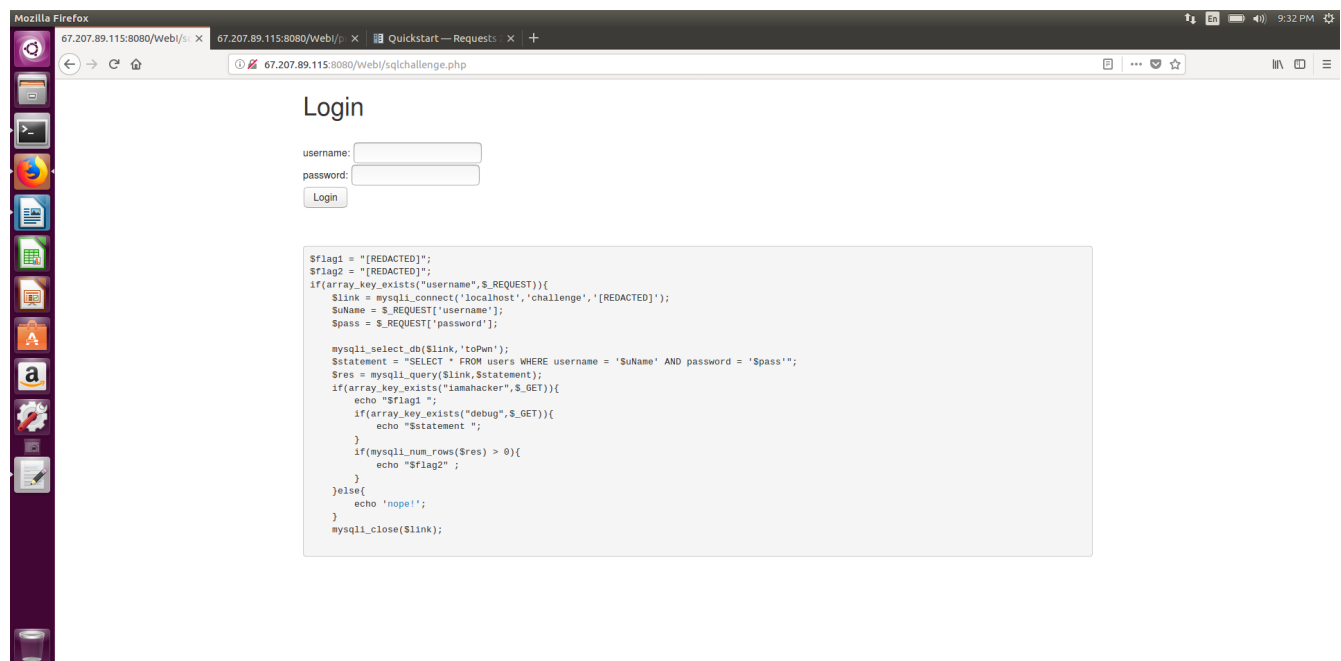
Problem:

This developer was so confident in their code they decided to provide the source. Can you figure out how to print the flag?

<http://67.207.89.115:8080/WebI/sqlchallenge.php>

Solution:

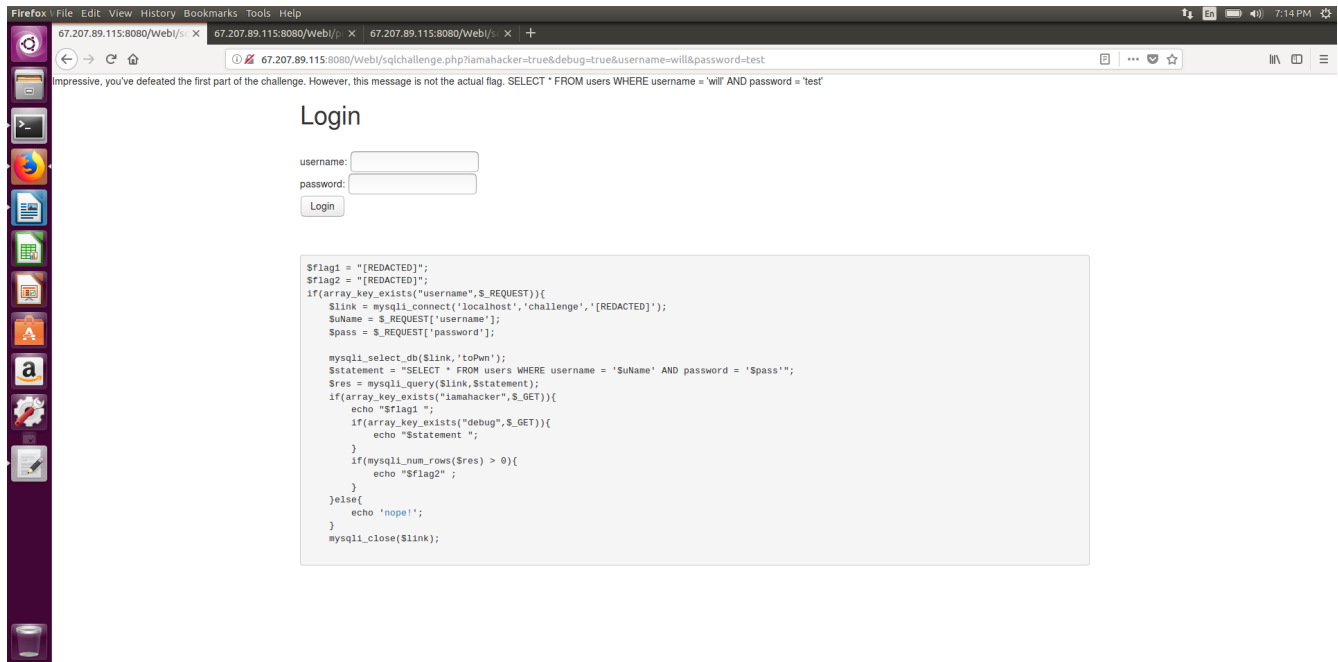
The page is as follows:



I immediately notice two flags and a statement that can be printed through satisfying conditions. The SQL connects to a database with `$link = mysqli_connect('localhost','challenge','[REDACTED]')`. After selecting a database a query is made using the username and password. This left me to believe the program was a SQL injection. To get flag\$1, I had to send 'iamahacker'=true to the SQL logic by typing it into the URL. This is because `if(array_key_exists("iamahacker",$_GET))` from the website is

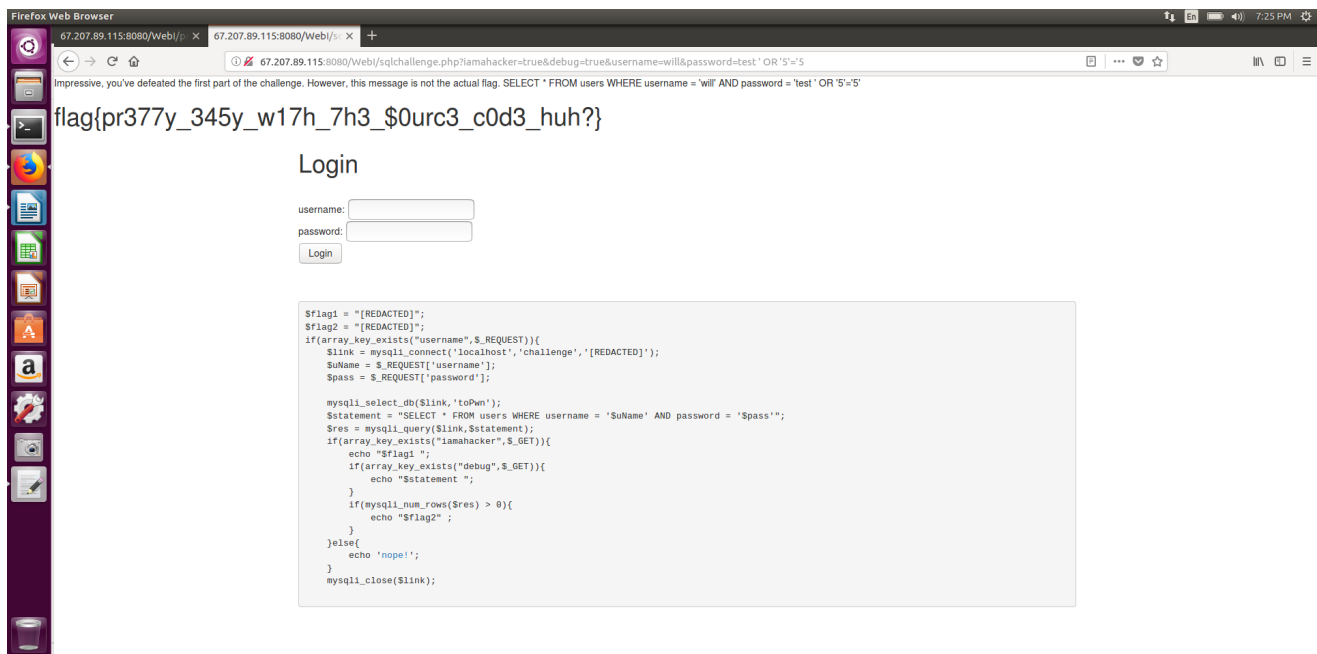
expecting the 'iamahacker' key to exist. _GET requests are received through the URL. To get \$statement to print I had to do the same thing for the 'debug' key for the same reasons.

At this point my URL was <http://67.207.89.115:8080/WebI/sqlchallenge.php?iamahacker=true&debug=true&username=will&password=test> . I set the username and password to random things because they are irrelevant at the time. The web page looked like this.



It showed the SQL statement being executed and proved flag2 was the actual flag. To get flag2 to print the number of rows stored in \$res must be one or more. This means that \$res must contain something other than null. For this to happen the SQL statement must be evaluated as true. The query must be valid. This is because \$res = mysqli_query(\$link,\$statement); and flag2 is printed inside of if(mysqli_num_rows(\$res) > 0). To get the SQL statement to always evaluate to true I performed a SQL injection at the end of my URL statement. I used ...&password=test ' OR '5'='5 so that the SQL statement would always evaluate to true. I found info on making the page always evaluate to true in the slides. This makes the page the SQL evaluate to SELECT * FROM users WHERE username = 'will' AND password = 'test ' OR '5'='5'. It adds in a ' at the end of test to make the server think the SQL for password entry is over and then OR '5'='5' makes the SQL always true. Now that \$res is populated with a 'valid' query, it has rows and the flag is printed.

Final URL: <http://67.207.89.115:8080/WebI/sqlchallenge.php?iamahacker=true&debug=true&username=will&password=test' OR '5'='5>



Profile 1

20 points

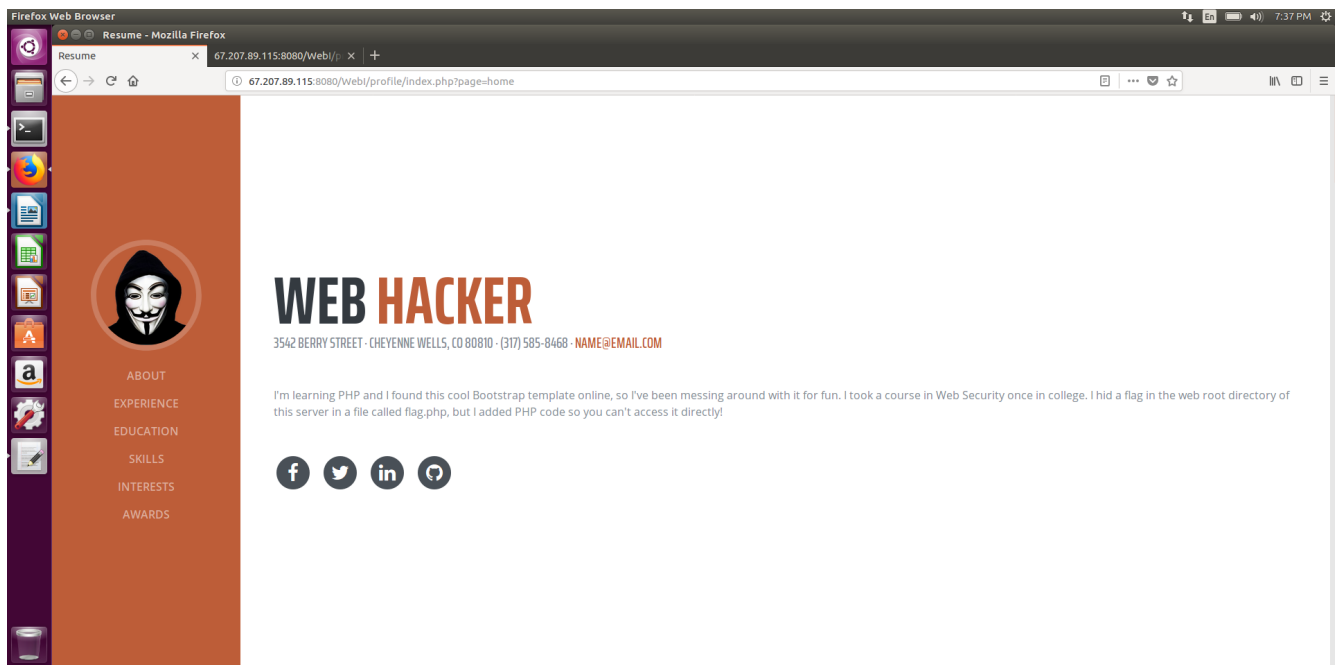
Problem:

We found the website of a person who claims to be a Web Hacker. They were so confident in their development skills that they challenged us to retrieve the flag.php file from the web application root (not the server root, but the directory that follows .com). Find a way to print the flag.

<http://67.207.89.115:8080/WebI/profile/index.php?page=home>

Solution:

The page is as follows:



This is a LFI problem because the hacker is tasking us to expose his directory hierarchy.

flag.php is in the web application root and the web application root immediately follows .com. The current URL is <http://67.207.89.115:8080/WebI/profile/index.php?page=home>. This is of the form server address: port number / path on server. A server address can be an actual website such as <http://class.n0l3ptr.com/>. Therefore, flag.php is located in a directory following <http://67.207.89.115:8080/>. This is the same directory as Web1. Upon inspection of the source code of the website, I noticed `<!-- HINT!!!` Is there anything suspicious about this URL? How does the page appear to be loading? -->

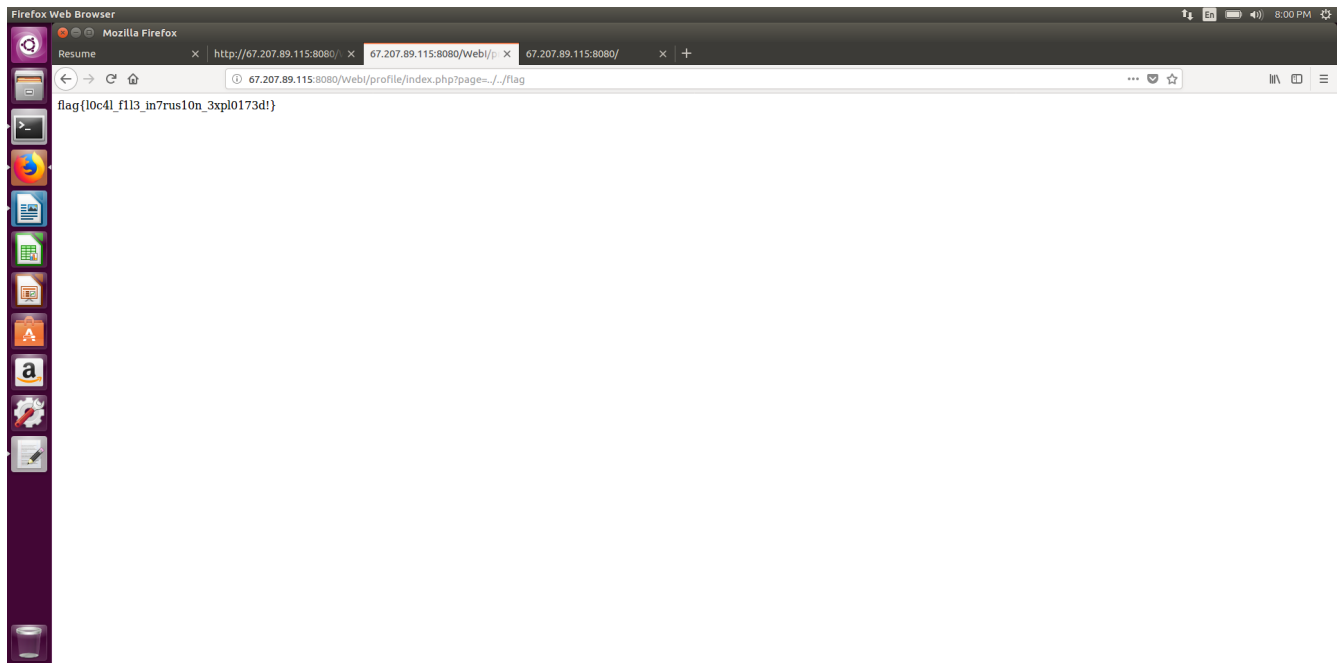
Which made me realize the page was being loaded from `?page=home`. To make a flag.php load I had to set `?page=flag.php`. However, I noticed another hint in the source code of the webpage inside some hidden text that said the hacker set the web page to automatically include the .php extension. So it is actually `?page=flag` without the extension. To get to the web directory, I had to go up two directories using `../../` and then I searched for the flag.php file. I went up two directories because /Web1/ is located two directories above index.php which is the current directory of the web page. Web1 is inside the web root directory which is the same directory where flag.php is located. My final url to get the flag was: <http://67.207.89.115:8080/WebI/profile/index.php?page=../../flag>

This problem proved a little difficult at first because I thought you could scale directories using <http://67.207.89.115:8080/../../page=../../flag>

Once I realized that wasn't the case. I tried

<http://67.207.89.115:8080/WebI/profile/index.php?page=../../flag>

Then I realized you need `../..` instead of `/../..`. This issue took me quite some time and a lot of reviewing pathnames.



Top Hacker

30 points

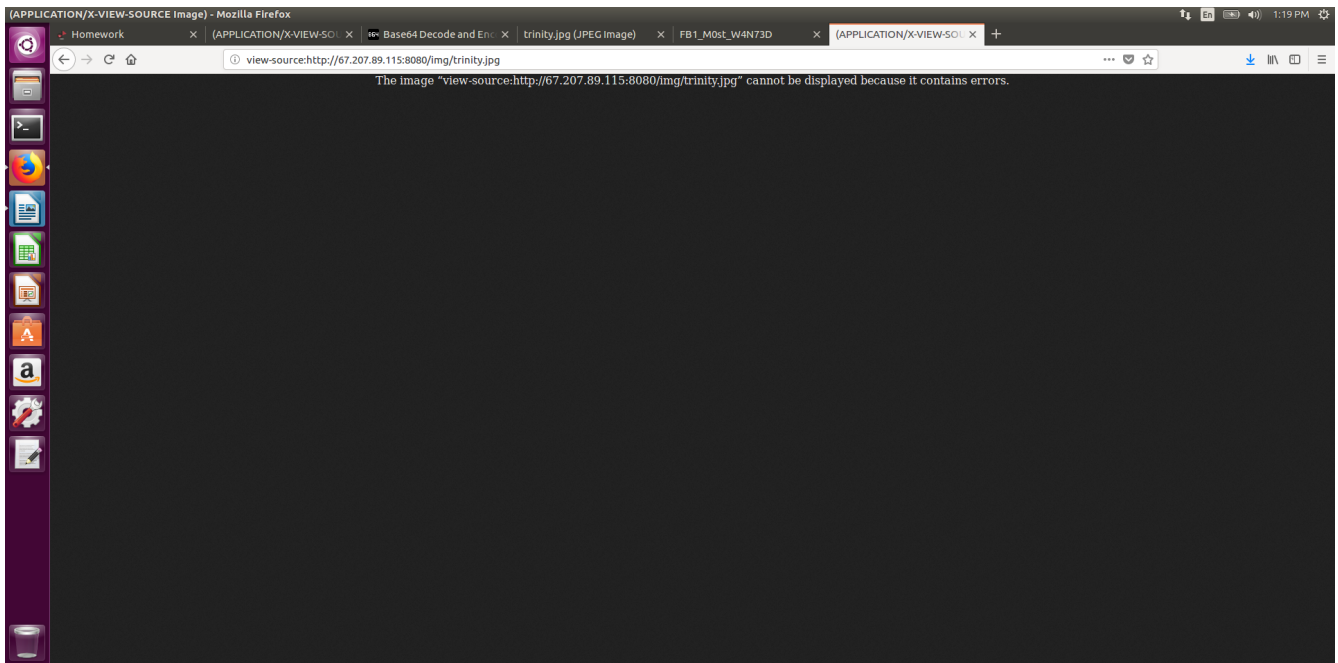
Problem:

We created a site to highlight some of the most infamous hackers of all time, according to Kaspersky. Unfortunately, the site was hacked and modified. We aren't sure exactly what changed, but we understand that there is now a flag hidden somewhere on the page. You might have to use different tools than before to solve this one. See if you can figure out what the hackers did and find a flag.

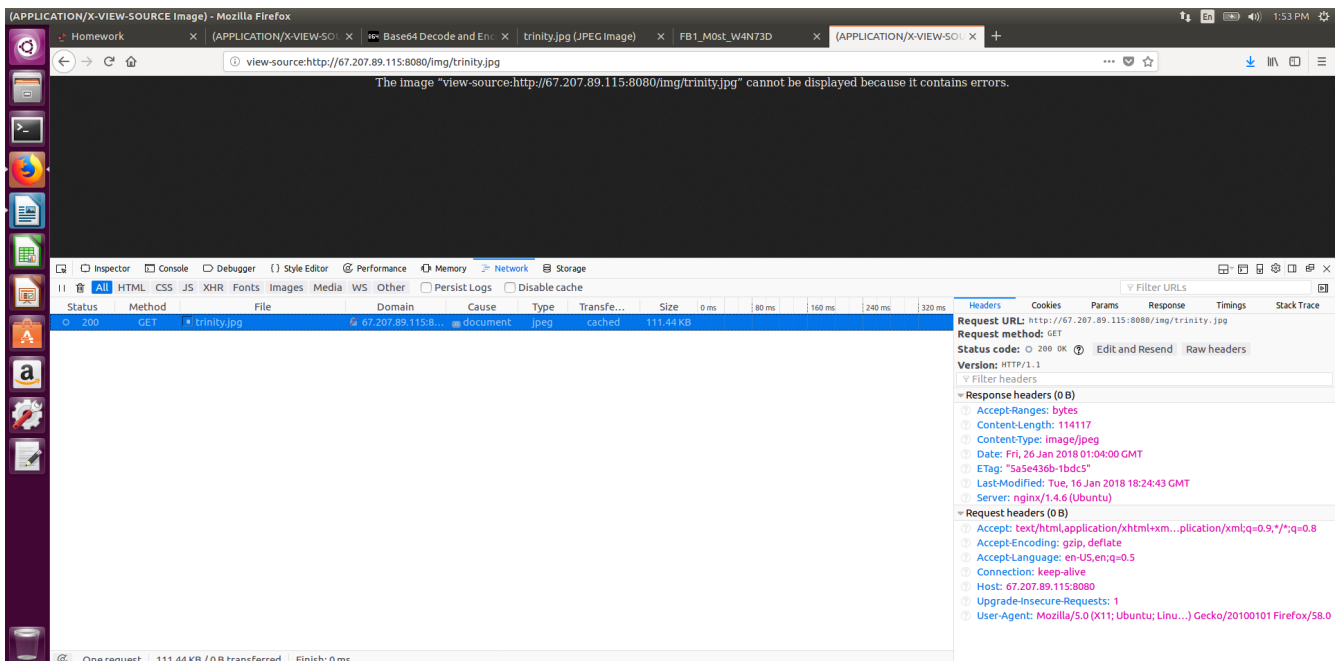
<http://67.207.89.115:8080/WebI/tophackers.php>

Solution:

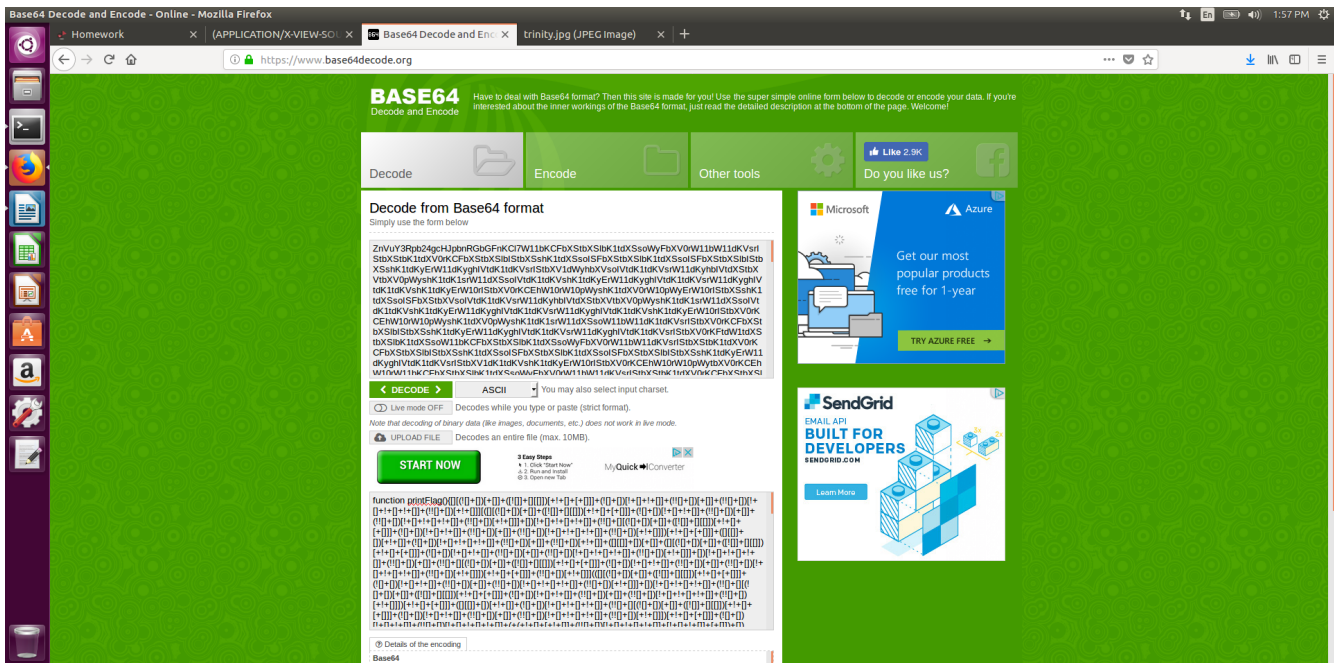
The website is as follows:



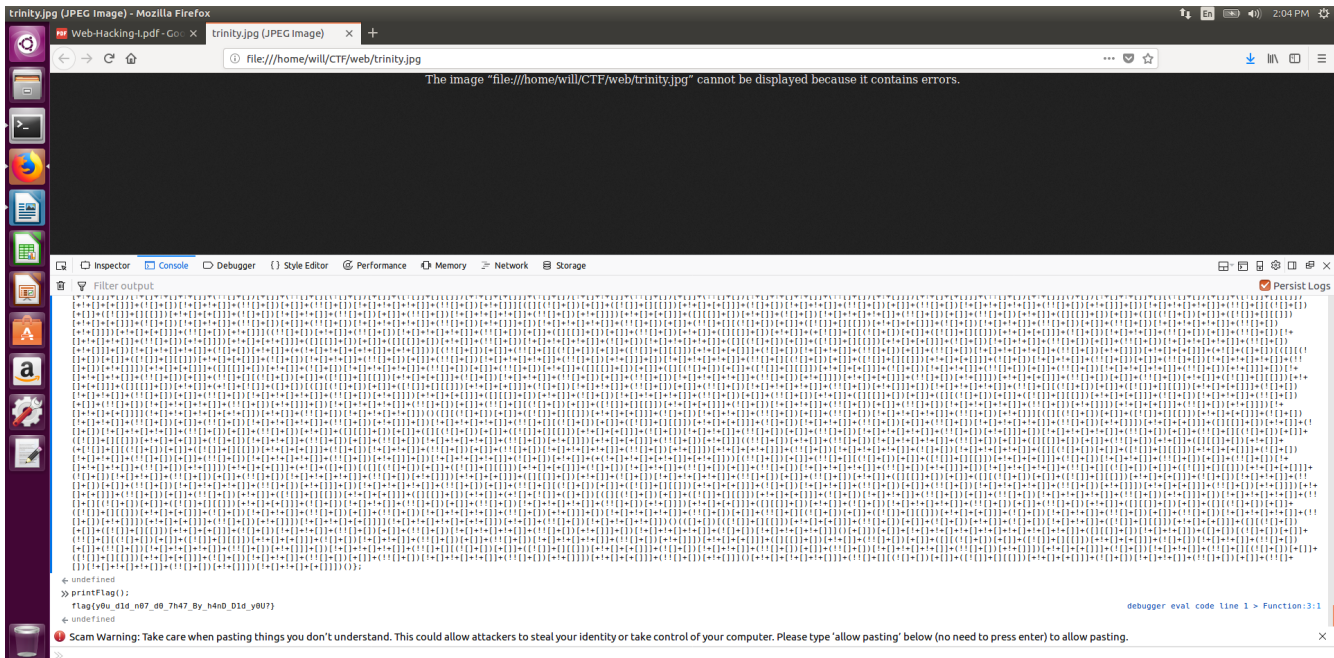
This means that the web page is having trouble loading the image as a jpg file. The reason for this could be that the file is not formatted as a jpg. The hacker could have just added a jpg extension to a non jpg file to hide the flag. My next step is to inspect trinity.jpg. To do that I must download the file. I did so by pressing f12 to bring up the network tab of the website after performing the _GET request of the web page. From there I could save the file that was trying to be sent.



I proceeded to use “cat trinity.jpg” in the terminal. That revealed a long list of alphanumeric characters. Since the entire file only contained ASCII characters and was passed with ‘==’, I figured that it was base64 encoded because base64 only produces characters in the ASCII range and ‘=’. After copying the file into a base64 decoder I got the following string:



We see 'function printFlag()' followed by what appears to be machine code. This appears to be a function definition. After inspecting the web 1 slides on XSS, I see that JavaScript defines functions in this way. Therefore, this must be a JavaScript function. Firefox has a built-in JavaScript interpreter. First, I navigated to the web page of the broken image because I assumed that page contained the flag. I accessed the JavaScript interpreter by clicking f12 and navigating to the console tab. Upon copying in the decoded base64 string into the console and adding a semicolon to execute the command (this is necessary for proper JavaScript syntax) I noticed nothing happened. I then called the printFlag() function and the flag was printed.



This question took me a very long time and Jacob Mills helped immensely to guide me in the correct direction. I spent a lot of time trying to decode meaningless expressions that I found in the network

activity. Jacob told me to look at the picture files. I spent even longer trying to make sense of the decoded base64 string. I was confident I was decoding it wrong and `printFlag()` was not the correct message. Once I was told that it was correct in my decoding, I was able to research how to utilize the function and get the flag to print.