

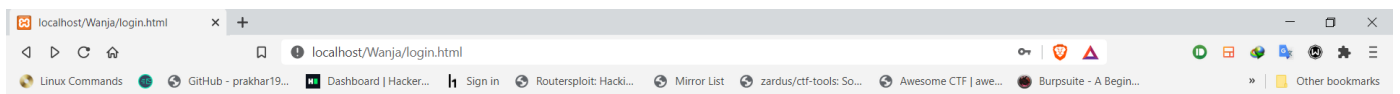
WINJA CTF SCREENING ASSIGNMENT

CTF Challenge Write-up

ANEESH DUA

(workstuff.dua@gmail.com)

Level 1: HardCoded Password Leak - Login Bypass

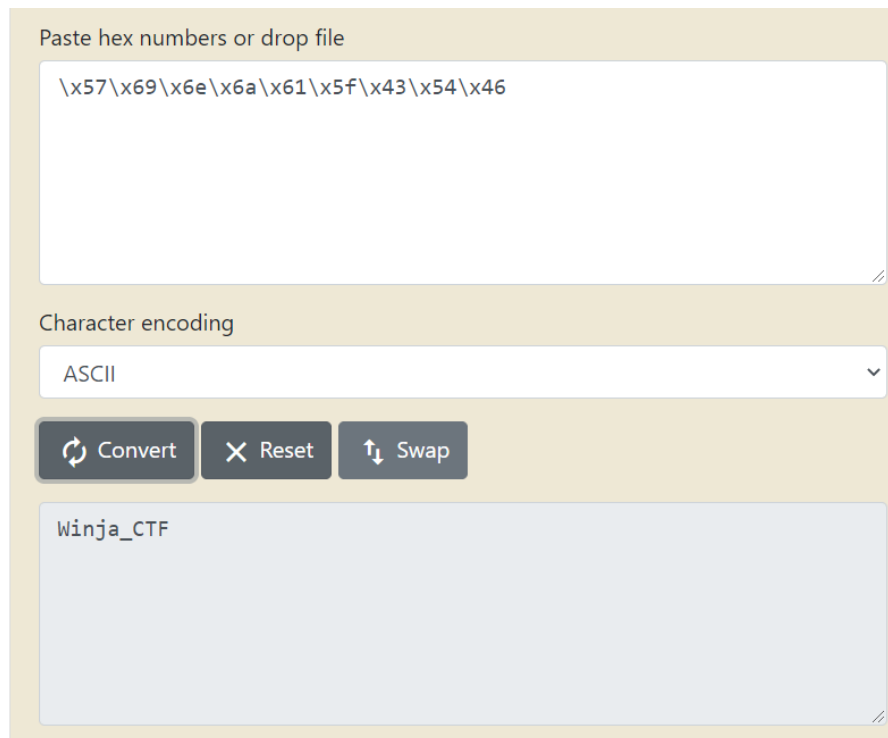


There is a treasure on this portal. We need to find it. Can you bypass the first login gate?

1. On visiting the challenge URL, we are presented with a login page. On inspecting the code using Ctrl+U or by simply right clicking and Select View Page Source, we find the `loginFunction()` inside the *script* tags of the page. This function validates the username and password credentials.

```
21
22     function loginFunction()
23     {
24         var username = document.getElementById("username").value;
25         var password = document.getElementById("password").value;
26         var x = password.slice(0,9);
27         var y = password.slice(9);
28         var z = md5(y);
29         if( x == "\\x57\\x69\\x6e\\x6a\\x61\\x5f\\x43\\x54\\x46" ){
30             if(z=="c604f2c30681a857d34cfb2813fb0c0e"){
31                 window.location = "ultimate_login.html";
32                 return false;
33             }
34             else{
35                 alert("Try harder!!!!")
36             }
37         }
38         else{
39             alert("Incorrect credentials")
40         }
41     }
42 }
```

2. Assessing the code workflow, we understand that we **DO NOT** require a username for logging in as the `loginFunction()` does not process the username value. As you can see this javascript function is taking the input from the login form and the password is being sliced in 2 parts. The first part is being stored in x variable and getting compared with the **hex value**. Whereas the another half of the password is stored in y variable and being compared with the **MD5 hash** of the string. Let's decode both of them.



Paste hex numbers or drop file

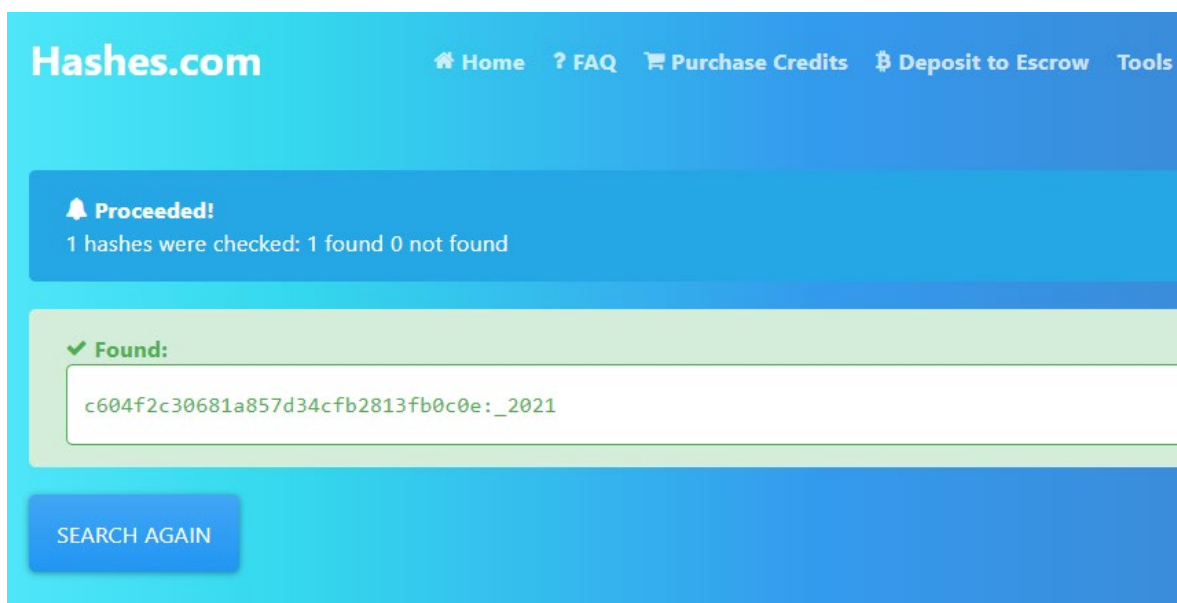
```
\x57\x69\x6e\x6a\x61\x5f\x43\x54\x46
```

Character encoding

ASCII

Convert Reset Swap

Winja_CTF



Hashes.com

Home FAQ Purchase Credits Deposit to Escrow Tools

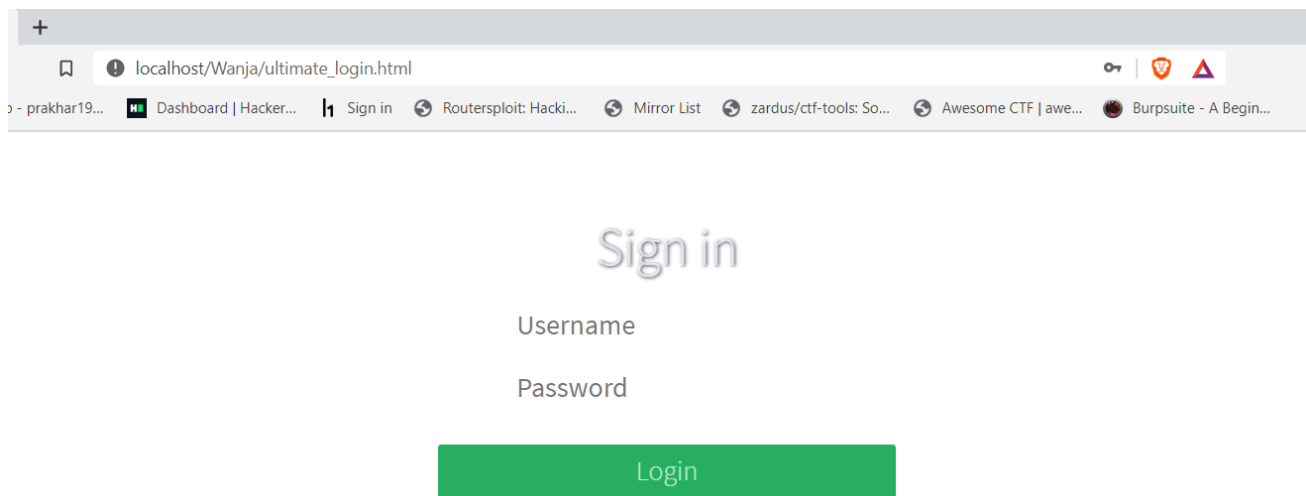
Proceeded!
1 hashes were checked: 1 found 0 not found

Found:
c604f2c30681a857d34cfb2813fb0c0e:_2021

SEARCH AGAIN

3. We have successfully got the password, now we can login by combining both of them. The password is: **Winja_CTF_2021**.

Level 2: SQL Injection



The key to this gate to the treasure can only be obtained by an injection.

1. We are redirected to the “ULTIMATE” login portal and we are required to bypass it. On inspecting the code same as before we are unable to find anything in the code. Trying default username-password combinations like admin-admin, root-toor etc also does not work.

2. However, we are given a hint related to injection. This directs us to the common SQL Injection vulnerability. Hence we try injecting the username field with the string **'OR 1 = 1 LIMIT 1 --'** and any random string as password. This injection works when the username input is not sanitised. For example- For the SQL query

```
SELECT * FROM users WHERE username = '$username' AND password = '$password'
```

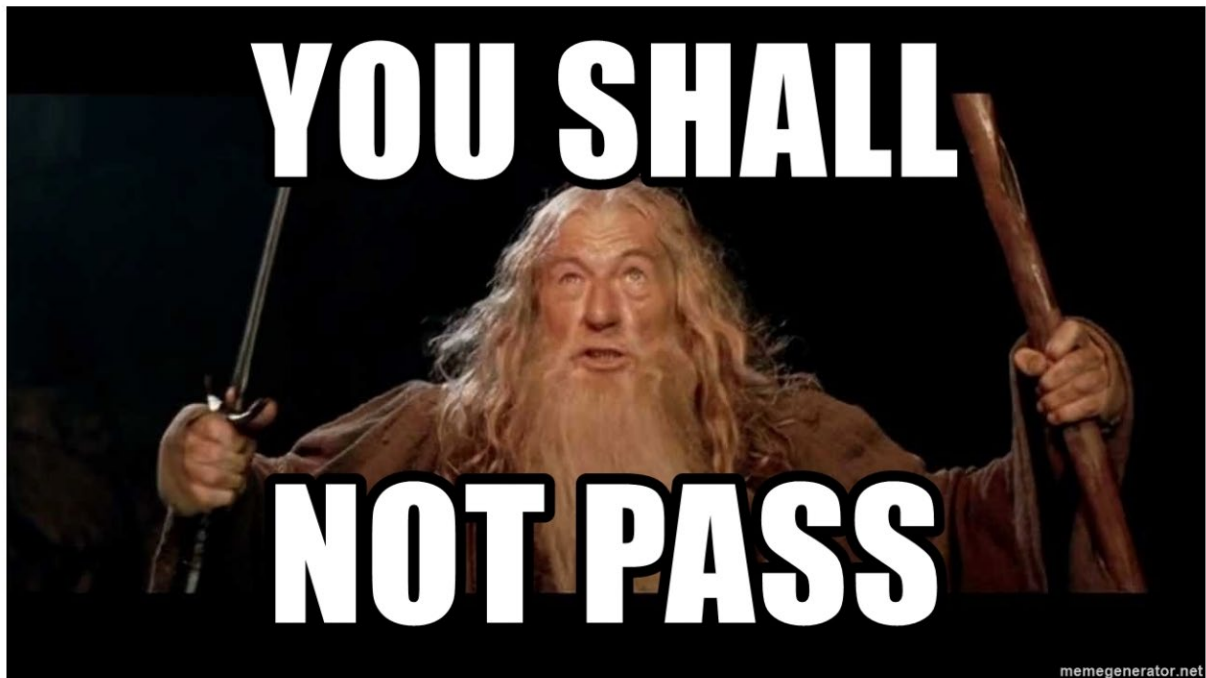
The injected string makes it –

```
SELECT * FROM users WHERE username = ' OR 1 = 1 LIMIT 1 -- ' AND password = abc
```

Thus, resulting in a True return for the query as 1=1 is always true. Thus, we bypass this level also.


Level 3: Steganography

1. We are redirected to an html page with a picture and a hint that Gandalf is hiding the treasure.



Oh no! GANDALF is protecting and hiding the treasure. What are you going to do?

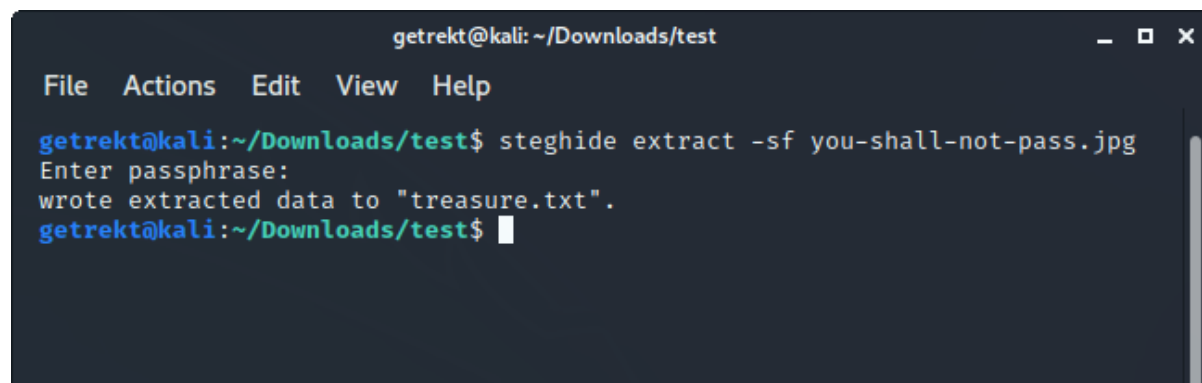
2. On initial inspection of the website source code as before we can conclude that we have only the image and the hint to go on.
3. We download the image and inspect its meta-data. But still we are unable to find anything.

File Name	you-shall-not-pass.jpg
File Size	171 kB
File Type	JPEG
File Type Extension	jpg
Mime Type	image/jpeg
Jfif Version	1.01
Resolution Unit	inches 
X Resolution	96
Y Resolution	96
Image Width	1280
Image Height	720
Encoding Process	Baseline DCT, Huffman coding
Bits Per Sample	8
Color Components	3
Y Cb Cr Sub Sampling	YCbCr4:2:0 (2 2)
Image Size	1280x720
Megapixels	0.922
Category	image
Raw Header	FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 00 60 00 60 00 00 FF DB 00 43 00 01 FF DB 00 43 01

The data above is all the metadata we could automatically extract from your file. It may be neither complete nor adequate. Metadata could have been changed or deleted in the past. Please be aware that

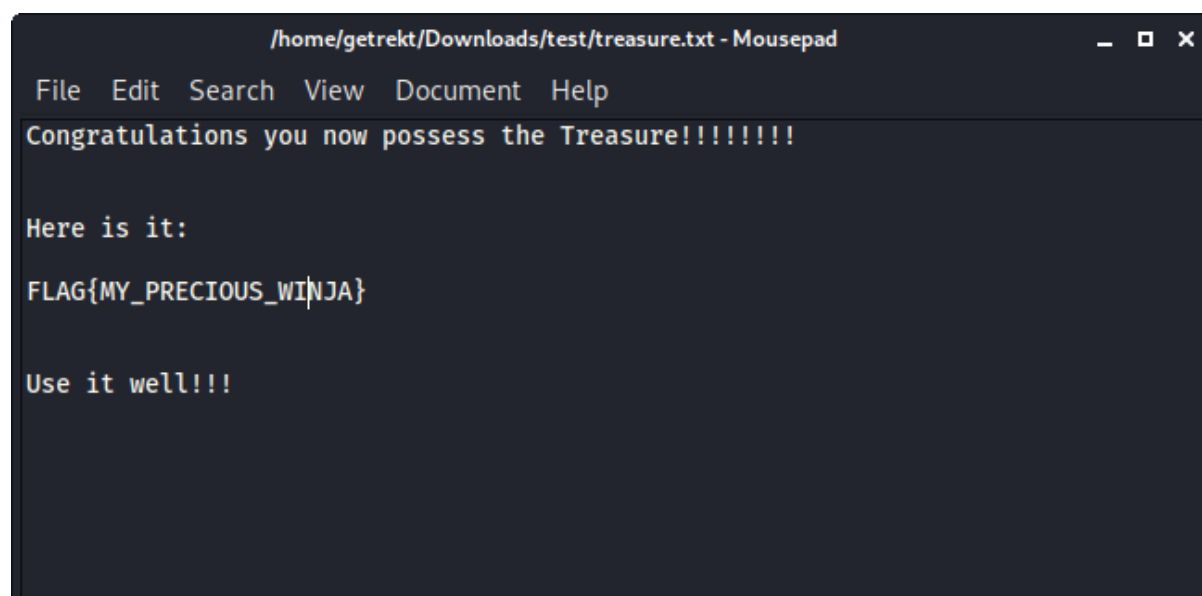
4. Focusing on the hint, we are told that Gandalf is "hiding" the treasure. Thus, we try extracting data using reverse steganography using the popular tool [steghide](#).

We use the command `steghide extract -s you-shall-not-pass.jpg`. When asked for the passphrase, we use the biggest hint in the level – Gandalf.

A terminal window titled 'getrekt@kali: ~/Downloads/test' with a menu bar (File, Actions, Edit, View, Help). The command 'steghide extract -sf you-shall-not-pass.jpg' is entered. The prompt 'Enter passphrase:' appears, followed by the message 'wrote extracted data to "treasure.txt".' The prompt returns to 'getrekt@kali:~/Downloads/test\$' with a cursor.

```
getrekt@kali: ~/Downloads/test
File  Actions  Edit  View  Help

getrekt@kali:~/Downloads/test$ steghide extract -sf you-shall-not-pass.jpg
Enter passphrase:
wrote extracted data to "treasure.txt".
getrekt@kali:~/Downloads/test$
```

A Mousepad window titled '/home/getrekt/Downloads/test/treasure.txt - Mousepad' with a menu bar (File, Edit, Search, View, Document, Help). The text inside the window reads: 'Congratulations you now possess the Treasure!!!!!!', 'Here is it:', 'FLAG{MY_PRECIOUS_WI\NJA}', and 'Use it well!!!'.

```
/home/getrekt/Downloads/test/treasure.txt - Mousepad
File  Edit  Search  View  Document  Help

Congratulations you now possess the Treasure!!!!!!

Here is it:

FLAG{MY_PRECIOUS_WI\NJA}

Use it well!!!
```

5. Voila! All challenges completed and the precious flag is given to us in the treasure.txt file. Hope Gandalf learns that he shouldn't use his own name for the secret passphrase!