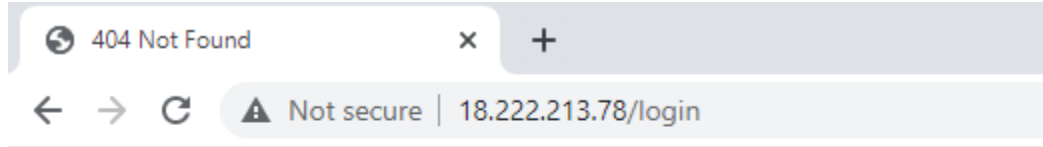


Web Bug Bounty

Part 1

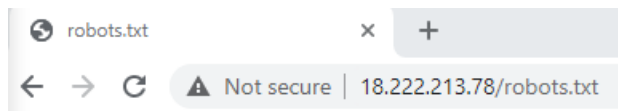
For the first part, I tried to append /login the the URL



Not Found

The requested URL was not found on the server. If you entered the URL manually

Then I appended /robots.txt



User-agent: Googlebot

Disallow: /

User-agent: *

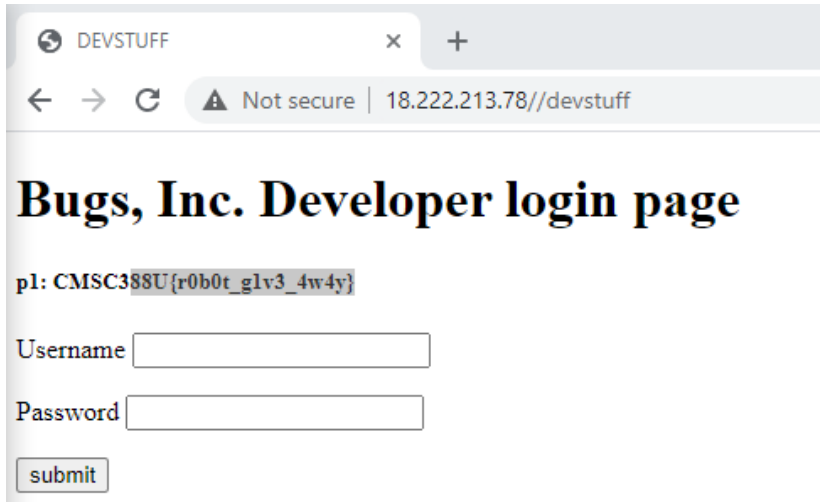
Disallow: /devstuff # Remove when in prod!

Allow /

User-agent: curl

Allow /robots.txt

Following <http://18.222.213.78/devstuff>



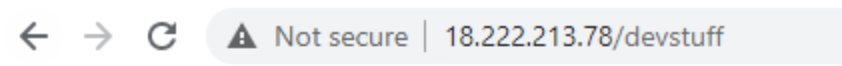
The first flag is CMSC388U{r0b0t_g1v3_4w4y}

Part 2

For the second part, i first tried default credentials

admin

admin



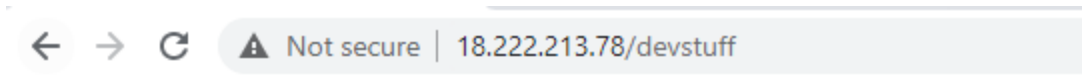
Bad login info!

Refresh your page to try again

Next i tried to do an SQL injection

admin') 1==1 --

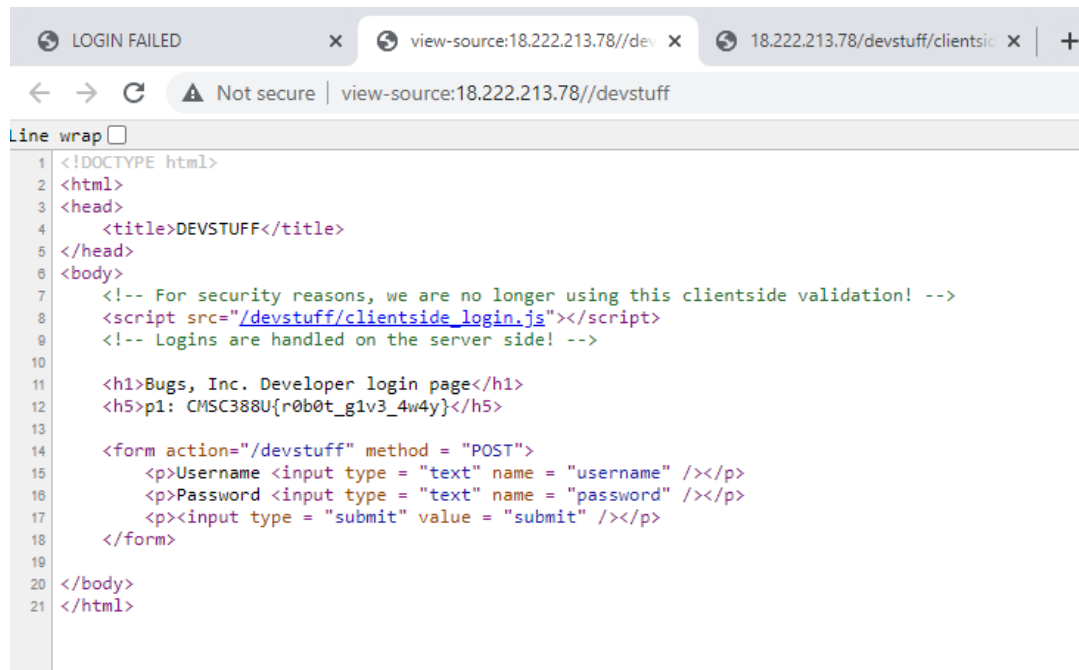
Password



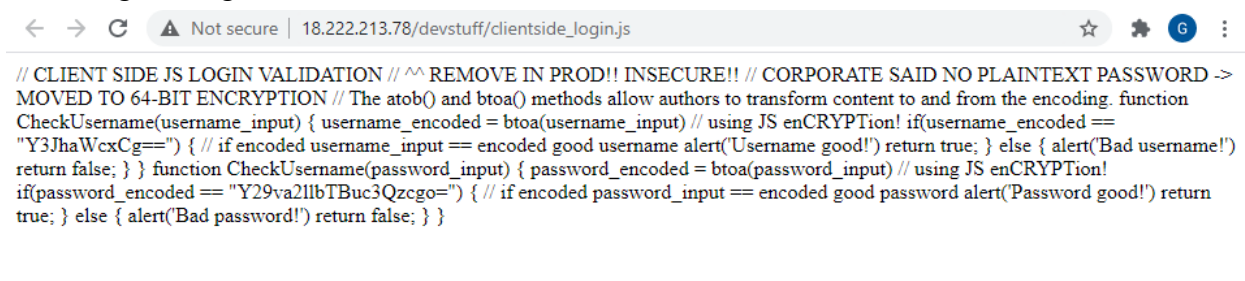
Bad login info!

Refresh your page to try again

Upon inspecting the page, i found a link to a .js file



Following it, we get



Using <https://www.10bestdesign.com/dirtymarkup/js/> to clean this up, the code looks like

```

1 // CLIENT SIDE JS LOGIN VALIDATION
2 // ^^ REMOVE IN PROD!! INSECURE!!
3 // CORPORATE SAID NO PLAINTEXT PASSWORD -> MOVED TO 64-BIT ENCRYPTION
4 // The atob() and btoa() methods allow authors to transform content to and from the encoding
5 function CheckUsername(username_input) {
6     username_encoded = btoa(username_input)
7     // using JS enCRYPTION!
8     if (username_encoded == "Y3JhaWcxCG==") { // if encoded username_input == encoded good username alert('Username good!') return true; }
9     else {
10         alert('Bad username!') return false;
11     }
12 }
13
14 function CheckUsername(password_input) {
15     password_encoded = btoa(password_input) // using JS enCRYPTION!
16     if (password_encoded == "Y29va2l1bTBuc3Qzcgo=") { // if encoded password_input == encoded good password alert('Password good!')
17         return true;
18     } else {
19         alert('Bad password!')
20         return false;
21     }
22 }

```

While the password credentials are encoded, we can see from the comments

```

// CORPORATE SAID NO PLAINTEXT PASSWORD -> MOVED TO 64-BIT ENCRYPTION
// The atob() and btoa() methods allow authors to transform content to and from the encoding

```

That they use 64 bit encryption and that we can use atob() and btoa() to decode/encode
Using an online script,

The screenshot shows a web browser's developer console with a script editor and a console output. The script editor has tabs for 'index.html', 'styles.css', and 'script.js'. The 'script.js' tab is active, showing the following code:

```

1 import 'bootstrap@4.6.0'
2
3 console.log(atob("Y3JhaWcxCG=="));
4 console.log(atob("Y29va2l1bTBuc3Qzcgo="));
5

```

The console output, located at the bottom, shows the results of the console.log statements:

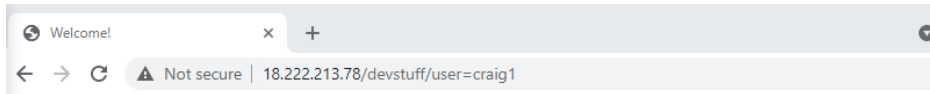
```

craig1
cookiem0nst3r

```

I found the credentials to be
craig1
cookie0nste3r

This didn't work at first, so I got confused but then I realized I had to refresh the page and I got in.



Welcome, craig1 !

p2: CMSC388U{3nc0d1ng_15_n0t_3ncrypt10n}

If you have any questions, please contact the user: "admin"

Now we know you're really crag1, enjoy this picture:



The flag is CMSC388U{3nc0d1ng_15_n0t_3ncrypt10n}

Part 3

For part 3, to login as admin, I noticed the URL for craig1 was

<http://18.222.213.78/devstuff/user=craig1>

The user=craig1 is interesting so i changed it to admin

<http://18.222.213.78/devstuff/user=admin>



Welcome, admin !

p3: CMSC388U{ID0R_TH3_W0RLD}

Great job! Here's another cat pic:



The flag is CMSC388U{ID0R_TH3_W0RLD}