

Web Bug Bounty

Part 1

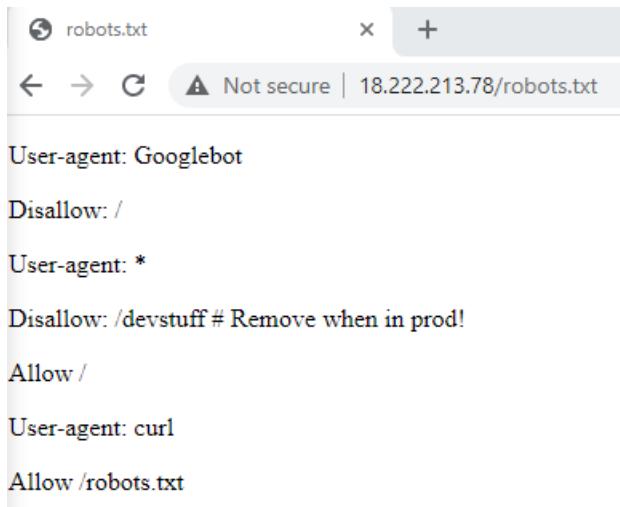
For the first part, I tried to append /login to the URL



Not Found

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

Then I appended /robots.txt



Following <http://18.222.213.78//devstuff>

The screenshot shows a web browser window titled "DEVSTUFF". The address bar indicates the URL is "Not secure | 18.222.213.78//devstuff". The main content area displays the title "Bugs, Inc. Developer login page". Below the title, there is a message "p1: CMSC388U{r0b0t_g1v3_4w4y}" followed by a form with fields for "Username" and "Password", and a "submit" button.

p1: CMSC388U{r0b0t_g1v3_4w4y}

Username

Password

The first flag is CMSC388U{ r0b0t_g1v3_4w4y }

Part 2

For the second part, i first tried default credentials

admin

admin

The screenshot shows a web browser window with the same URL as the previous one. The main content area displays the message "Bad login info!" in large bold letters, followed by "Refresh your page to try again" in a smaller font.

← → C Not secure | 18.222.213.78/devstuff

Bad login info!

Refresh your page to try again

Next i tried to do an SQL injection

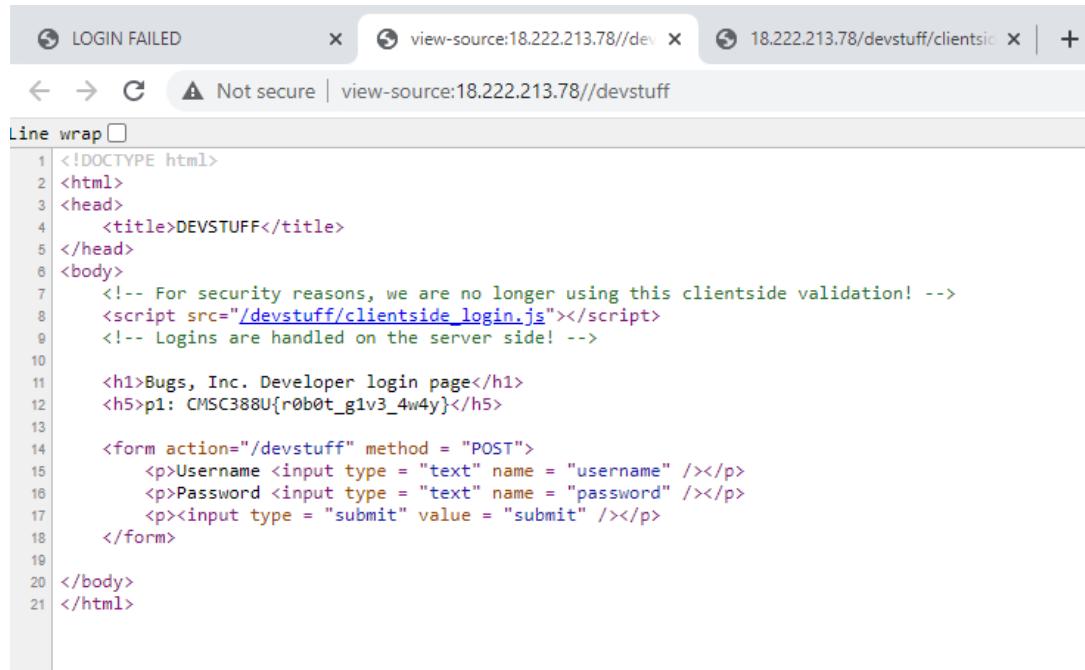
admin') 1==1 --

Password

Bad login info!

Refresh your page to try again

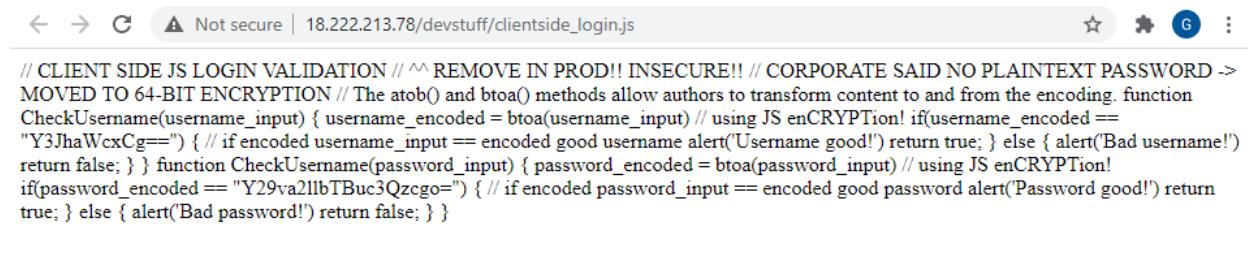
Upon inspecting the page, i found a link to a .js file



The screenshot shows a browser window with three tabs. The active tab is titled "view-source:18.222.213.78//devstuff". The source code is displayed in a monospaced font:

```
Line wrap □
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>DEVSTUFF</title>
5 </head>
6 <body>
7   <!-- For security reasons, we are no longer using this clientside validation! -->
8   <script src="/devstuff/clientside_login.js"></script>
9   <!-- Logins are handled on the server side! -->
10
11 <h1>Bugs, Inc. Developer login page</h1>
12 <h5>p1: CMSC388U{r0b0t_g1v3_4w4y}</h5>
13
14 <form action="/devstuff" method = "POST">
15   <p>Username <input type = "text" name = "username" /></p>
16   <p>Password <input type = "text" name = "password" /></p>
17   <p><input type = "submit" value = "submit" /></p>
18 </form>
19
20 </body>
21 </html>
```

Following it, we get



The screenshot shows a browser window displaying the contents of the file "clientside_login.js". The code is as follows:

```
// CLIENT SIDE JS LOGIN VALIDATION // ^ REMOVE IN PROD!! INSECURE!! // 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. function
CheckUsername(username_input) { username_encoded = btoa(username_input) // using JS enCRYPTION! if(username_encoded ==
"Y3JhaWcxCg==") { // if encoded username_input == encoded good username alert('Username good!') return true; } else { alert('Bad username!')
return false; } } function CheckPassword(password_input) { password_encoded = btoa(password_input) // using JS enCRYPTION!
if(password_encoded == "Y29va2lhbTBuc3Qzcgo=") { // if encoded password_input == encoded good password alert('Password good!') return
true; } else { alert('Bad password!') return false; } }
```

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 CheckPassword(password_input) {
15     password_encoded = btoa(password_input) // using JS enCRYPTION!
16     if (password_encoded == "Y29va211bTBuc3Qzcgo=") { // 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 browser developer tools window. At the top, there are tabs for 'index.html', 'styles.css', 'script.js' (which is currently selected), and 'WEB'. Below the tabs, the script.js code is displayed:

```

index.html  x  styles.css  x  script.js  x

1 import 'bootstrap@4.6.0'
2
3 console.log(atob("Y3JhaWcxCg=="));
4 console.log(atob("Y29va211bTBuc3Qzcgo="));
5

```

At the bottom, the 'CONSOLE' tab is active, showing the output of the script:

```

craig1
cookieM0nst3r

```

I found the credentials to be
craig1
cookiem0nste3r

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

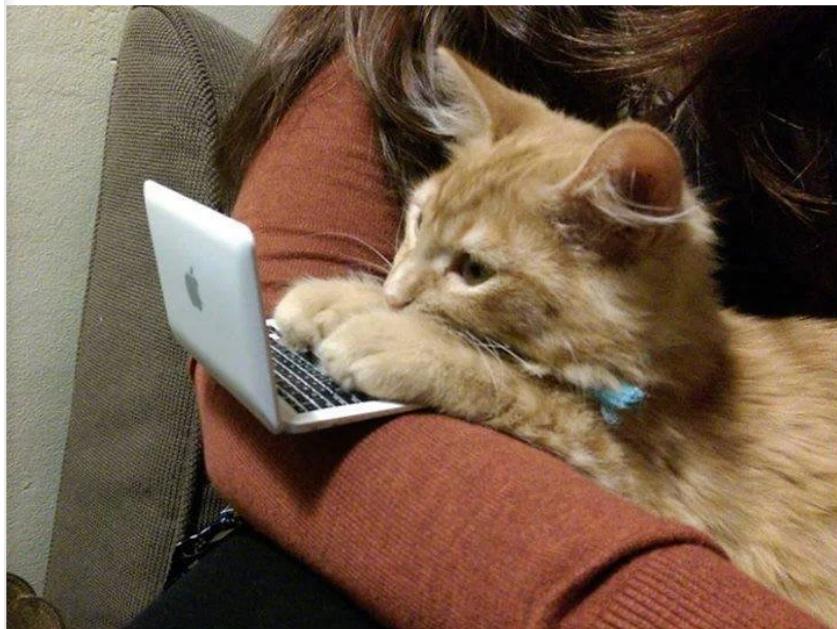


Welcome, craig1 !

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!

Not secure | 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}