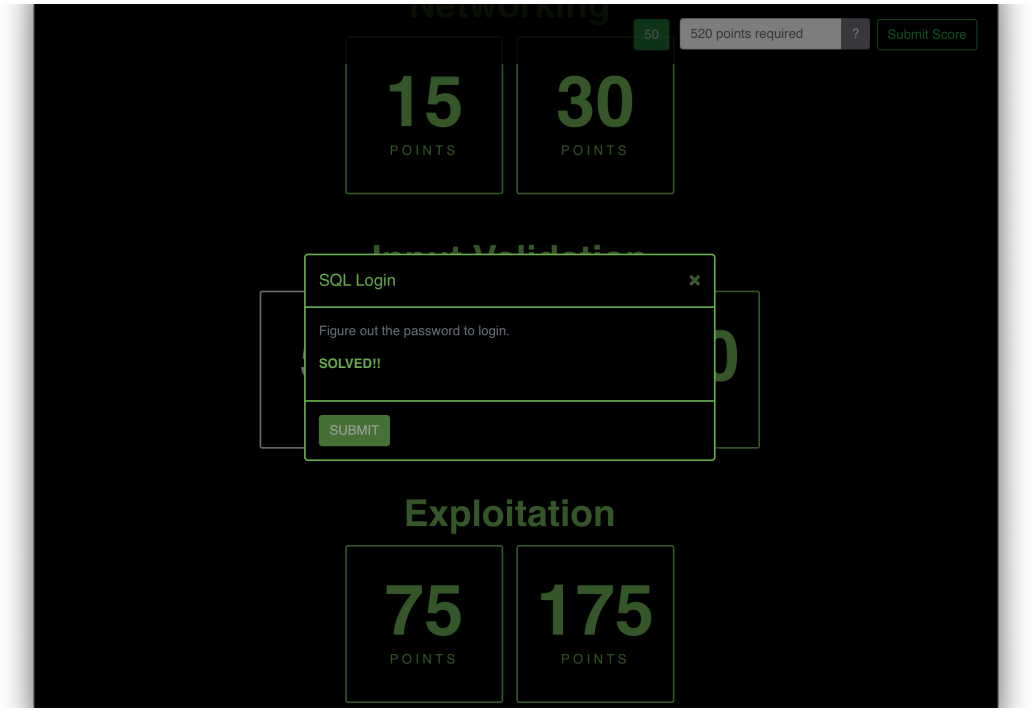
# Input Validation

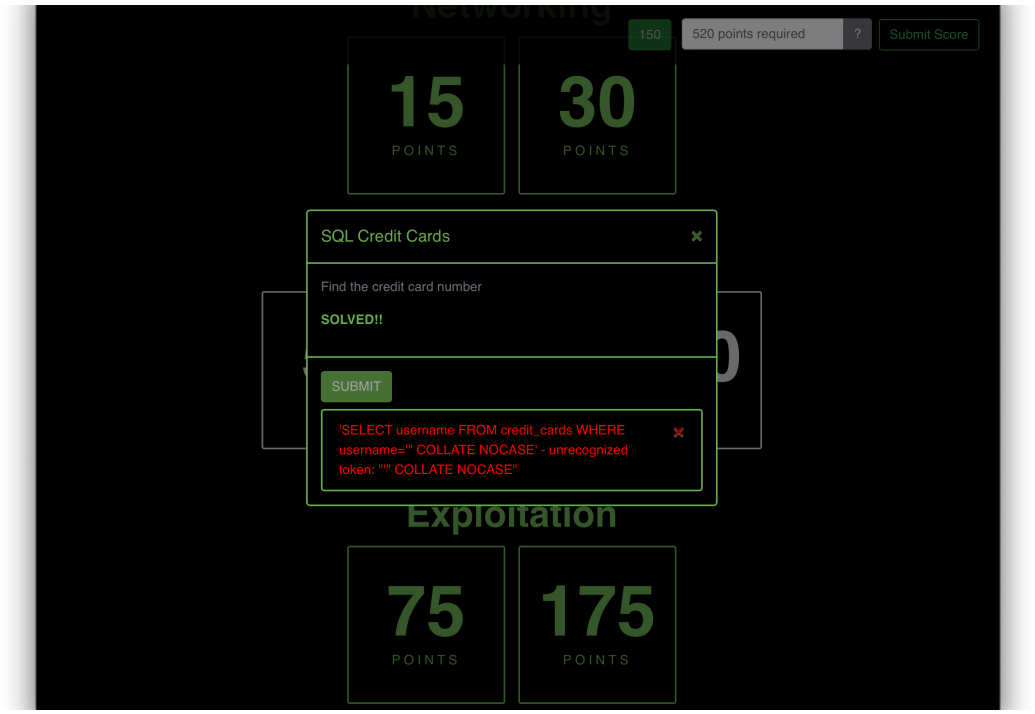
1.




**SQL LOGIN**

This problem is fairly easy to break because it's a simple login form. I test it by entering " ' ", and there's an error message shown as : 'SELECT username, password FROM users WHERE username='admin' AND password='''' - unrecognized token: "''''". Based on the given SQL information, I can figure out that we can either enter " ' or '1'='1" to break into it. Then it spits out the password for "scruffy", it is "Im_0n-br3ak". Time taken: 10 minutes.

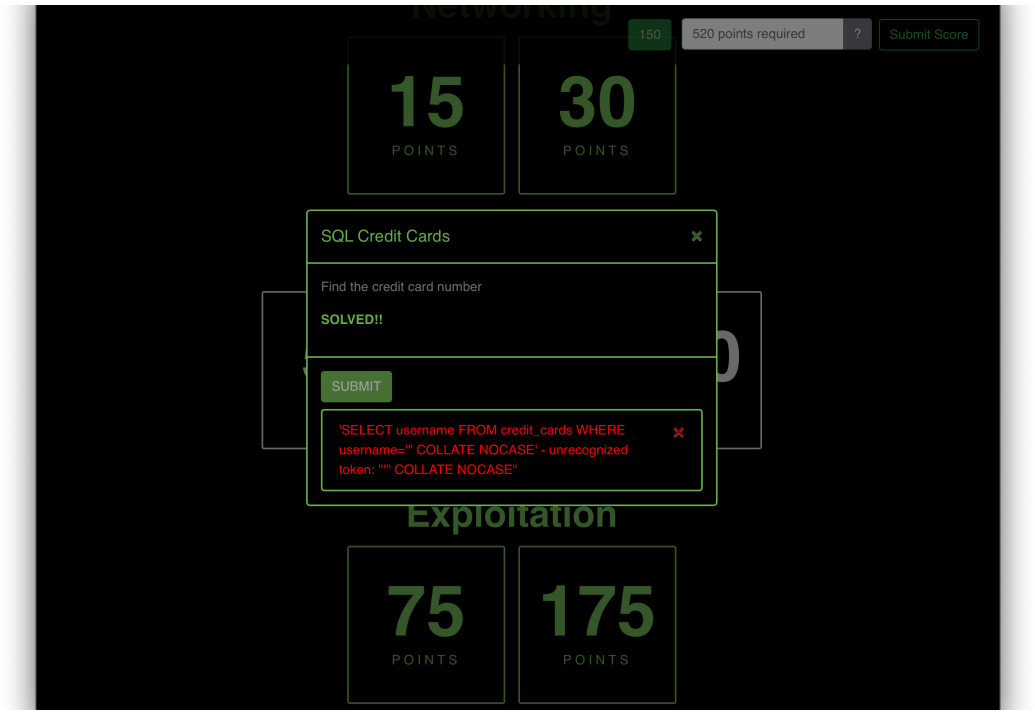
2.




**TYPE TO ENTER A CAPTION.**

It says it’s implemented a special aanti-XSS technology, so I try to test the input for URL with a basic XSS payload : <img src="http://url.to.file.which/not.exist" onerror=alert(document.cookie);> and submit. Then it displays “admin_sess_id=flag{in.stack*7hre4d.defaul7_f1ood}” , and here we have the flag. Now the image is not shown as a smiley cookie, so I “view image” and it directs me to an empty page. And I copy the image URL "https://www.ainfosec.com/%3Cimg%20src=" and here are the answers. Time taken: 30-35 minutes, researched XSS payload.

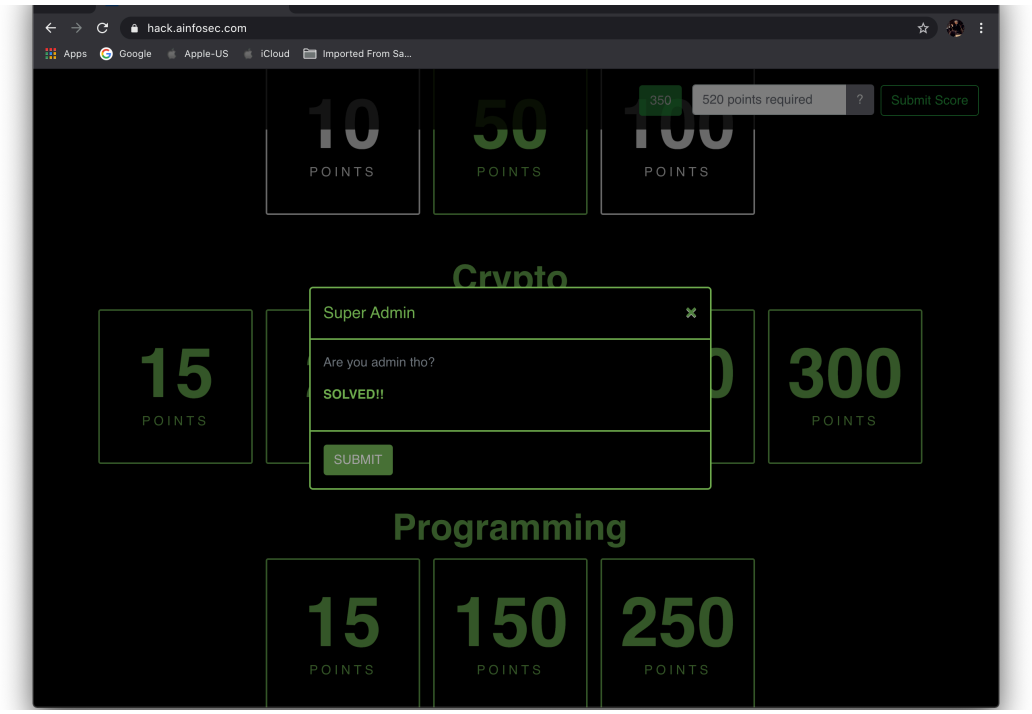
3.




**SQL CREDIT CARDS**

Again, I test the input by entering ' and it displays an error message 'SELECT username FROM credit_cards WHERE username='' COLLATE NOCASE' - unrecognized token: "'' COLLATE NOCASE". Based on the given SQL information, I figure out that a we have to forge the query. As I tried many times to find out the credit card number field name, finally it turns out to be “card”. And it gets easier, just simply enter “ ' and FALSE union SELECT card FROM credit_cards WHERE username=‘admin” and then the credit card number is displayed on the screen. This method works because the SQL entered can prompt the system to display the credit card number by searching for its username. Time taken: 1 hour 40 minutes

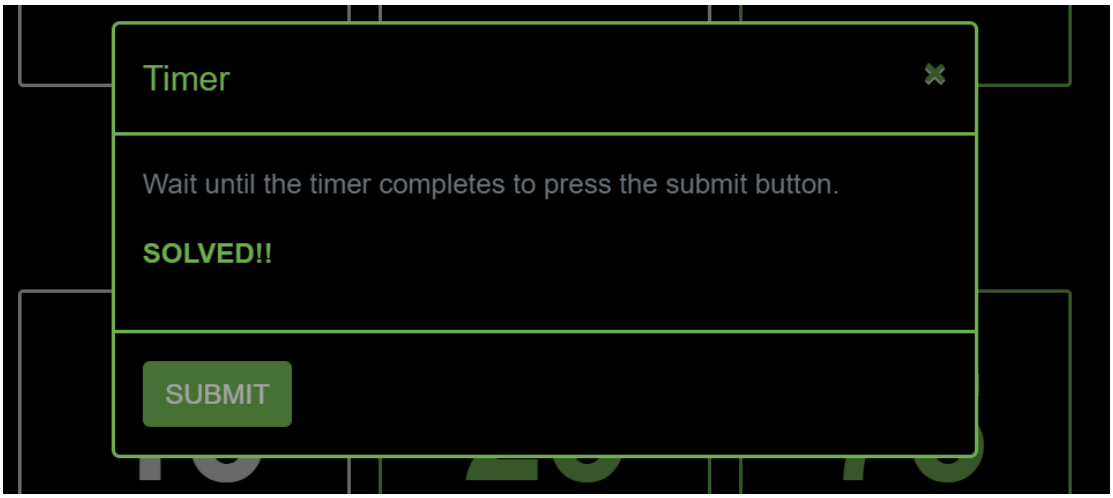
# Client Side Protections




**SUPER ADMIN**

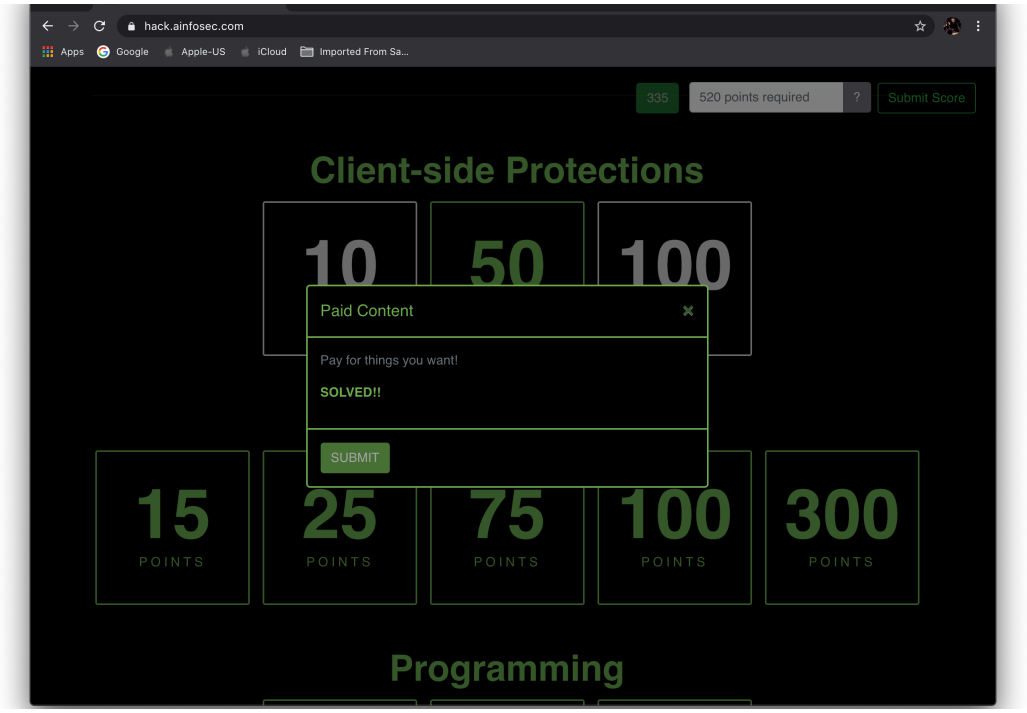
1. This is the easiest of all I believe. First I try to submit, and it shows an error message “Javascript validation failed. You are not super admin. is_super_admin = false”. Based on the last sentence, it seems like if we change “false” to “true” it might work. So I open up console and enter “is_super_admin=true”, click Submit again and it works now. Time taken: 5 minutes.
2. The task for us is to change the timer so it runs out of time fast so we can finish the challenge. First I try to inspect the elements on the page but I found out nothing helpful. Then I got an idea maybe it’s because of the Submit button, maybe we can modify something with it and get it to work. So I click Inspect on Submit button, and I click Submit, I see

there's an error message displayed, then I expand that information to find out more about it. And based on the information, I go back and select Break on Subtree Modification. After that, I go to sources and pause the script execution, so I am able to modify something. There's a JavaScript file called HackerChallenge.js, it opened up after I paused the script execution, and a line of code display.text(second) is highlighted. So what if I change the value of "seconds" to 0? It doesn't work because the value keeps decreasing by 1, and a value of "-1" doesn't make sense. So I change it to 1, which should go down to 0 in a second. So I type "seconds = 1" in console and resume the script execution, then click submit. It works! The idea behind the process is basically find out the function and variable that are responsible for the calculation of the timer, and change the variable to a smaller value so it runs out of time fast. Time spent: 3.5 hours+.




3. Like the Timer challenge, I use the dev tools to help me through the problem. I open up the Network tab and click Submit hoping it would give some useful feedback. And the error message I got was a "400 Status" for a "POST method". It basically means the server cannot process the

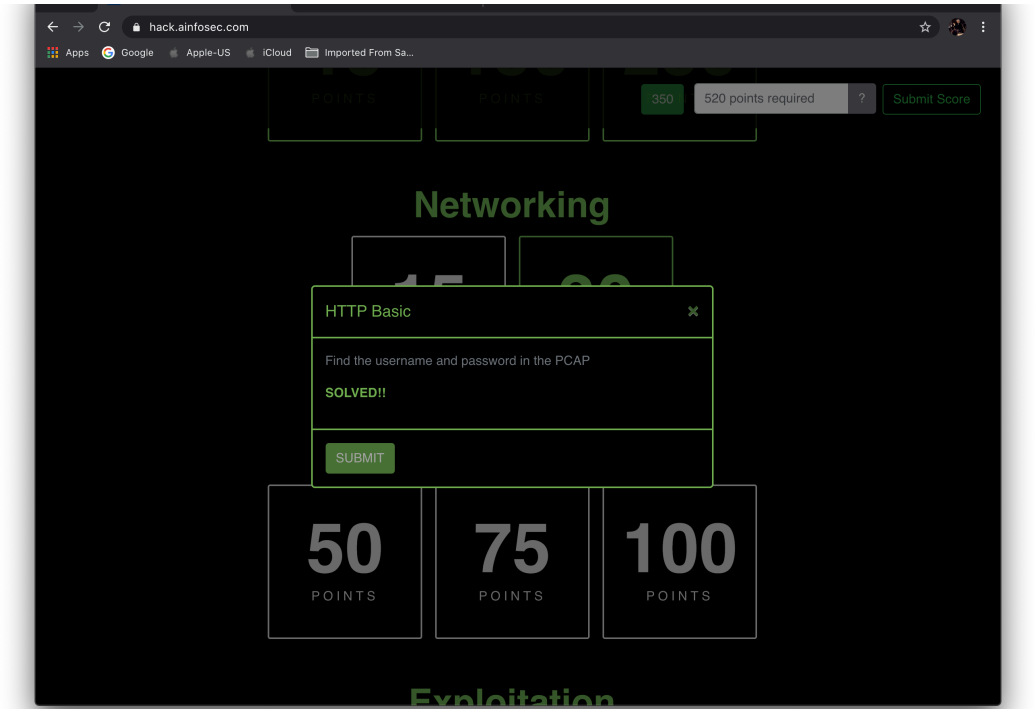
requests due to something that is perceived to be a client error, like malformed request syntax, invalid request message framing or deceptive request routing. And in console, I expand the feed back “XHR POST https://hack.ainfosec.com/challenge/submit-answer/ [HTTP/2.0 400 Bad Request” and click on “Stack Trace” to find the function and the file. Then I found out “hackerchallenge.js” is the file we are looking for. The code is not so much so I decided to look through it, how ever at line 53, there’s a “return $.ajax({“ to execute the AJAX call. Then I thought maybe set a breakpoint at that line of code. Another notable thing in the code is that I found out the variable “answer” could be the solution. So I set a breakpoint at line 53 and click submit. When the execution stopped, I tried typing “answer = answer.replace('"paid":false', ‘“paid":true')" in console and it should be the solution since I replaced the bool value of “paid” from “false” to “true”. After I resume the execution, it works. Time spent: 6 hours, researched JavaScript.




**PAID CONTENT**

# Networking

1.

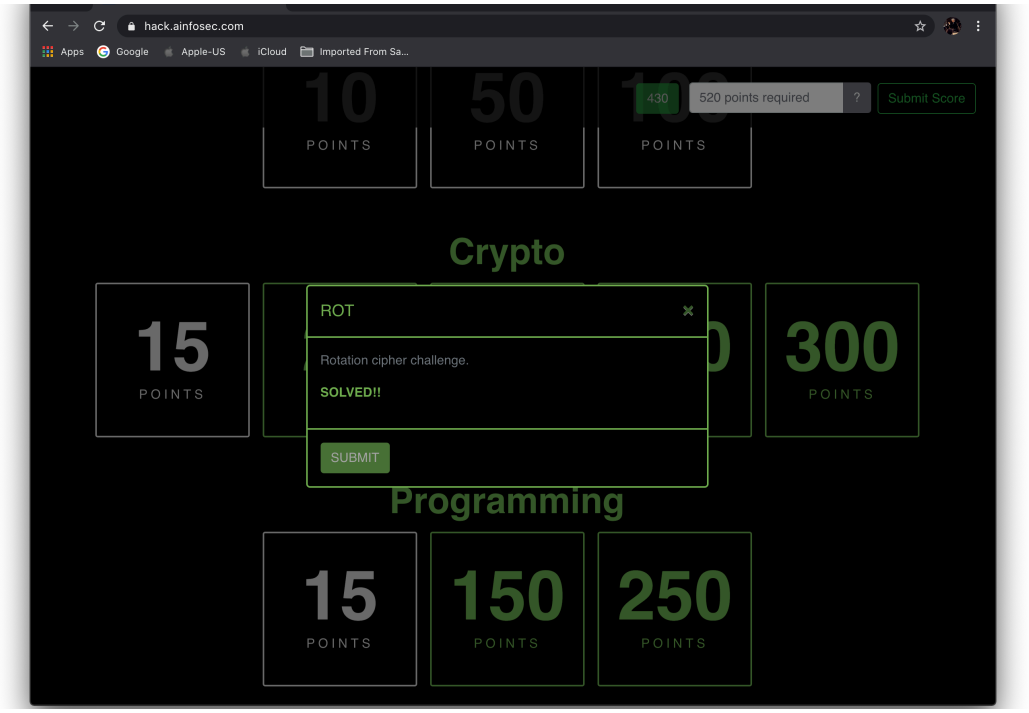



**HTTP BASIC**

It provides a link to download the file, and I see it's a ".cap" file which contains packets captured using a sniffing tool. So I think using Wireshark would probably help, so I load the file into it. But then I find out that the information it provided is not really helpful, so I enter "http.request.method == POST" in "Apply a display filter" so I can see all the forms submitted. I scroll to the bottom and I see "HTML Form URL Encoded: application/x-www-form-urlencoded" column, so I click on it and look for more. And the "name" and "pass" information are at the bottom, simply copy and paste

the 2 fields and submit. Time spent: 4.5-5 hours, researched JavaScript and “.cap” file in depth.

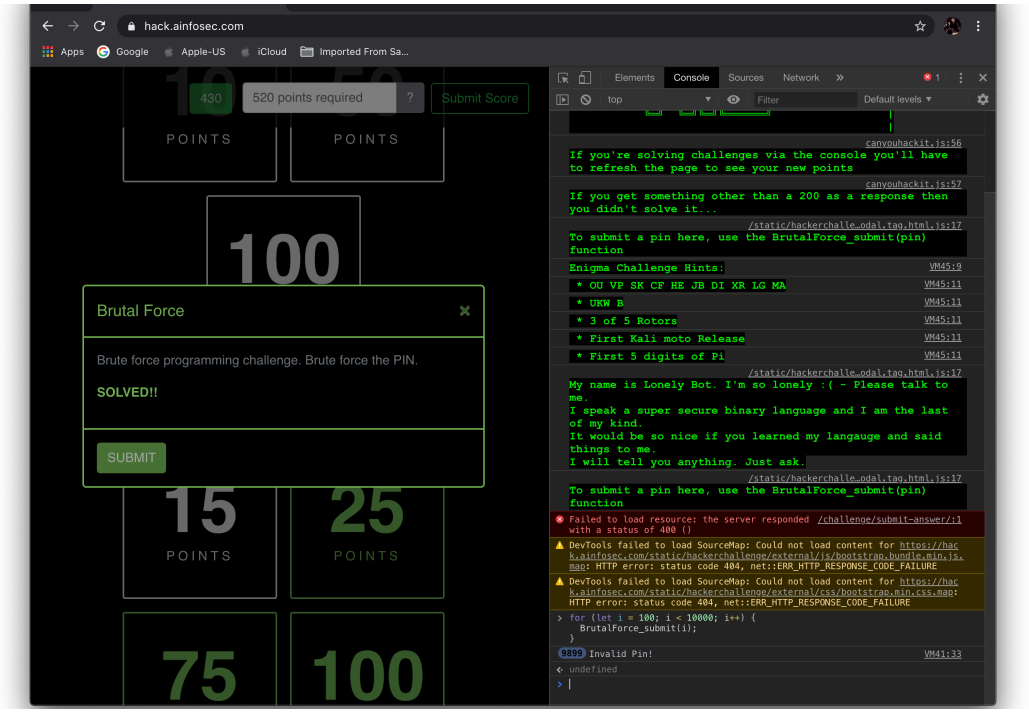
## Crypto




**ROT**

1. Since it directly tells me its ROT cipher, so I use the link “rot13.com" which is a decoder specifically designed to decode ROT ciphers. However it’s hard to know the key, so I tried it one by one and finally found out that a ROT14 is the key. Time spent: 30 minutes, researched ROT cipher.

# Programming




**BRUTE FORCE**

1. This is not hard, just write a simple “for loop” to brute force and solve the problem using BrutalForce_submit(pin) function. The code is for (let x = 100; x < 10000; i++) {BrutalForce_ssubmit(x);}. The program keeps looking for the PIN one by one and the problem is solved. Time spent: 1 hour.