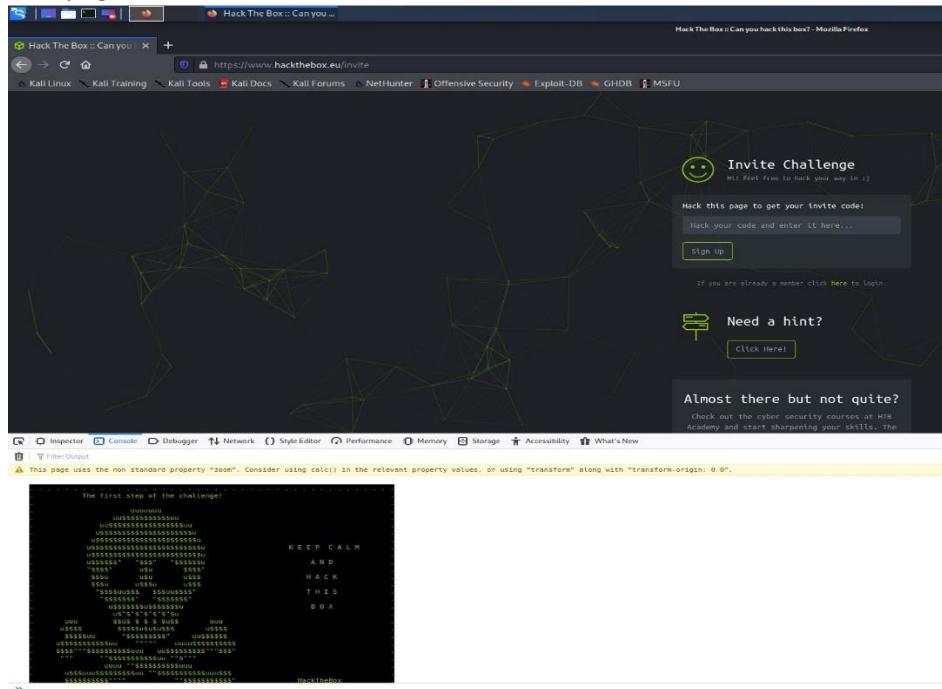


The purpose of this Final was to see that we could take what we have learned and apply it in a real-world scenario. The objective was to successfully join the website Hack the Box by gaining entry. The only way to gain entry was to procure an invite code by hacking the webpage.

1. I stated by navigating to the site and using the hint provided in the assignment, I checked the web page source code.



2. I found the first clue.

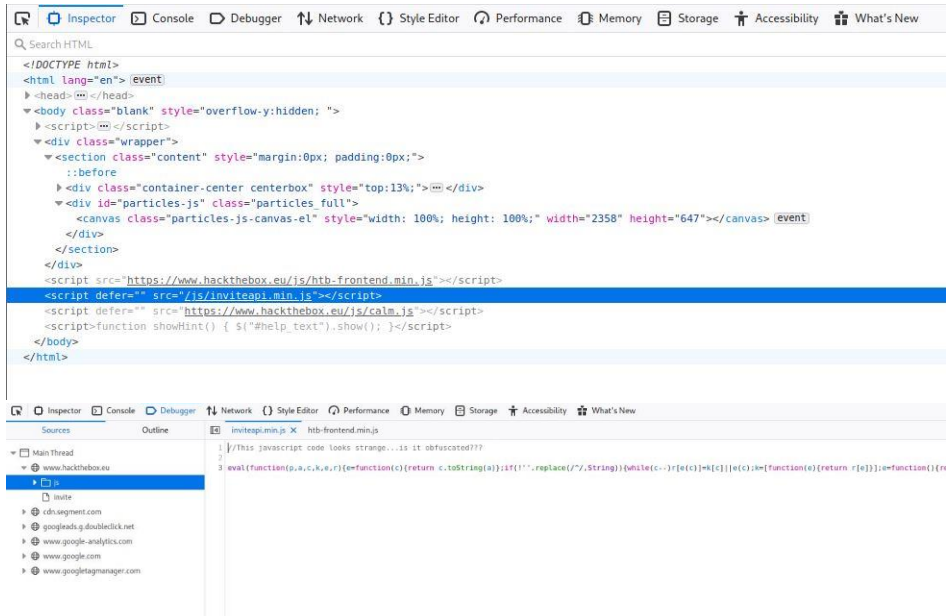


3. I went to the Inspector tab and found the first listed Java script. I opened that one in the built in Debugger tab and began going through the code. Did not find anything to indicate a next step.

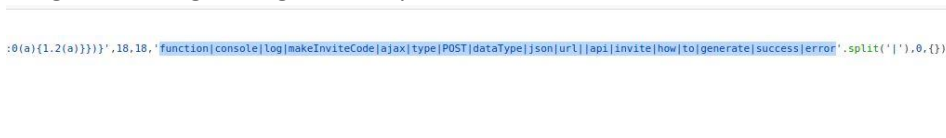




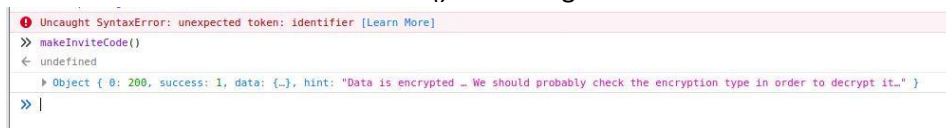
- I went back to the Inspector tab and opened the next Java script and found an indication that this was the one I was looking for.



- I began combing through the script and found the next clue.



- Seeing the string starting with function and entry for *makeInviteCode*, I knew that I should be able to enter that command in as a function in the console. I went back to the Console tab and ended the function *makeInviteCode()* and was given the next clue.

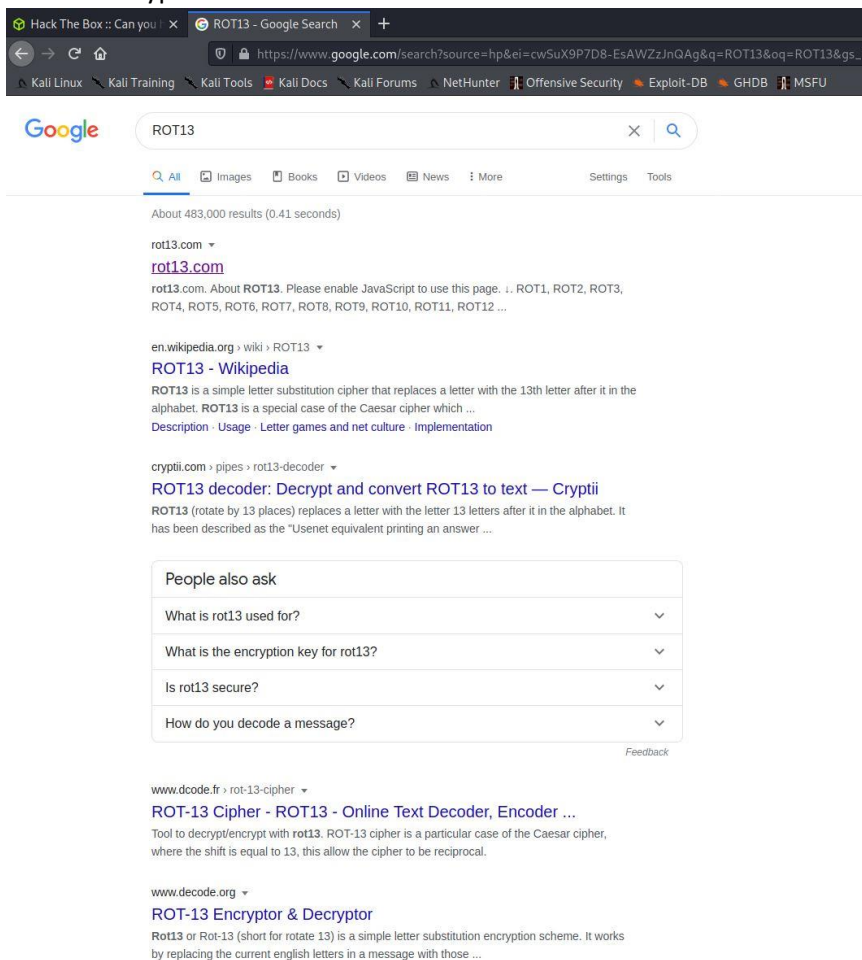


- I followed the instructions in the clue and checked for the encryption type on the data. I found that its encryption type was listed as ROT13.

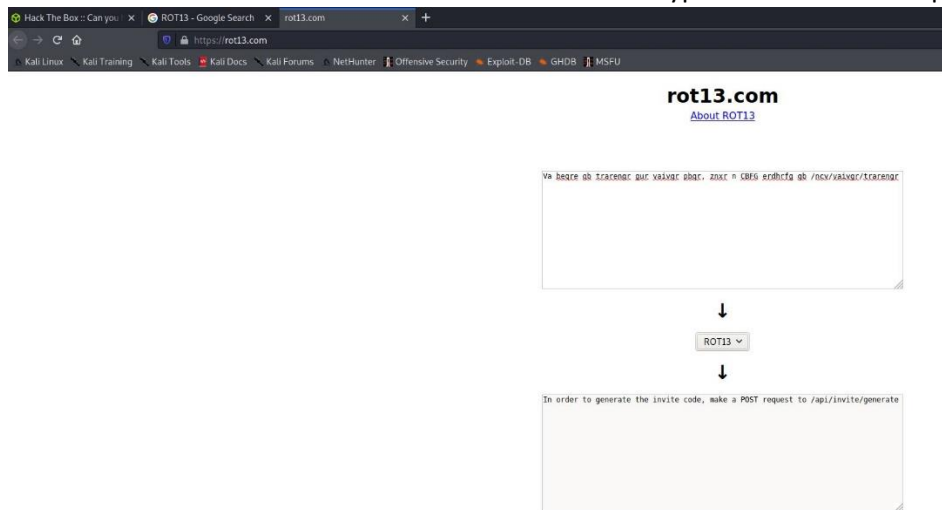
The screenshot shows a web browser's developer console with the following content:

- Uncaught SyntaxError: unexpected token: identifier** [Learn More]
- >> makeInviteCode()**
- <- undefined**
- [-]**
- 0: 200**
- data: {-}**
  - data: "Va beqre gb trarengr gur vaivgr pbqr, znxr n CBFg erdhrfg gb /ncv/vaivgr/trarengr"**
  - entype: "ROT13"**
  - <prototype: {[-]}**
    - \_defineGetter\_: function \_defineGetter\_()**
    - \_defineSetter\_: function \_defineSetter\_()**
    - \_lookupGetter\_: function \_lookupGetter\_()**
    - \_lookupSetter\_: function \_lookupSetter\_()**
    - \_proto\_: >>**
  - constructor: function Object()**
  - hasOwnProperty: function hasOwnProperty()**
  - isPrototypeOf: function isPrototypeOf()**
  - propertyIsEnumerable: function propertyIsEnumerable()**
  - toLocaleString: function toLocaleString()**
  - toString: function toString()**
  - valueOf: function valueOf()**
  - <get \_\_proto\_\_: function \_\_proto\_\_()**
  - <set \_\_proto\_\_: function \_\_proto\_\_()**
- hint: "Data is encrypted - We should probably check the encryption type in order to decrypt it-"**
- success: 1**
- <prototype: {[-]}**
  - \_defineGetter\_: function \_defineGetter\_()**
  - \_defineSetter\_: function \_defineSetter\_()**
  - \_lookupGetter\_: function \_lookupGetter\_()**
  - \_lookupSetter\_: function \_lookupSetter\_()**
  - \_proto\_: >>**
- constructor: function Object()**
- hasOwnProperty: function hasOwnProperty()**
- isPrototypeOf: function isPrototypeOf()**
- propertyIsEnumerable: function propertyIsEnumerable()**
- toLocaleString: function toLocaleString()**
- toString: function toString()**
- valueOf: function valueOf()**
- <get \_\_proto\_\_: function \_\_proto\_\_()**
- <set \_\_proto\_\_: function \_\_proto\_\_()**

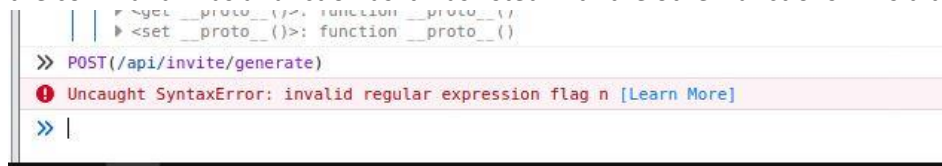
8. From there I did a Google search on ROT13 and got back results for decoders that worked with ROT13 encryption.



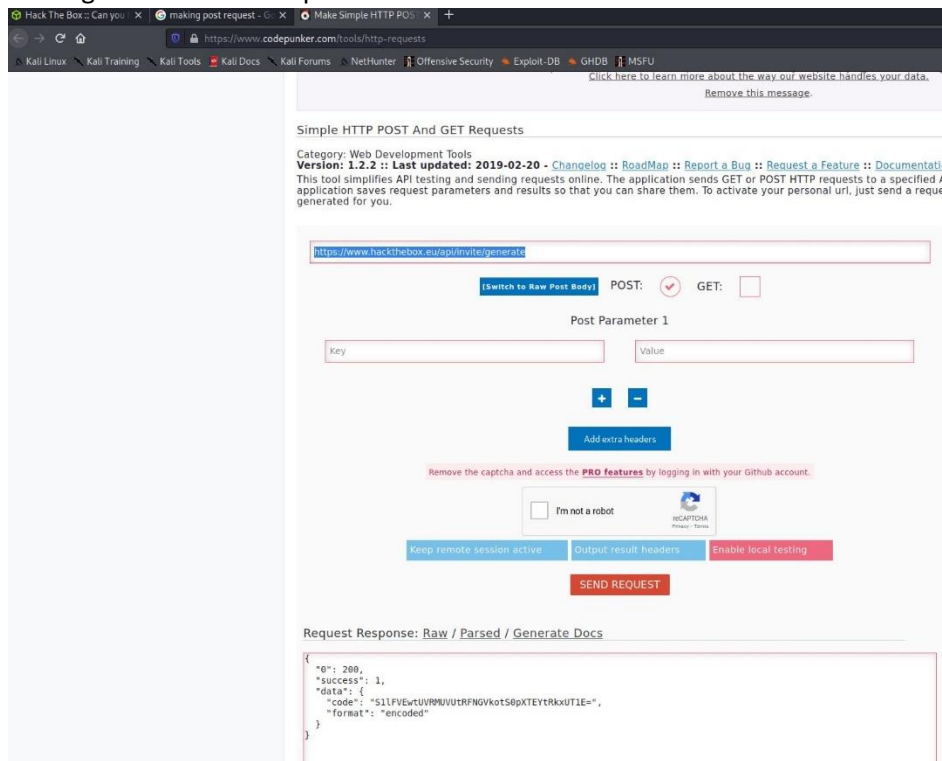
9. I selected one of the listed decoders and used it to decrypt the data from the previous clue.



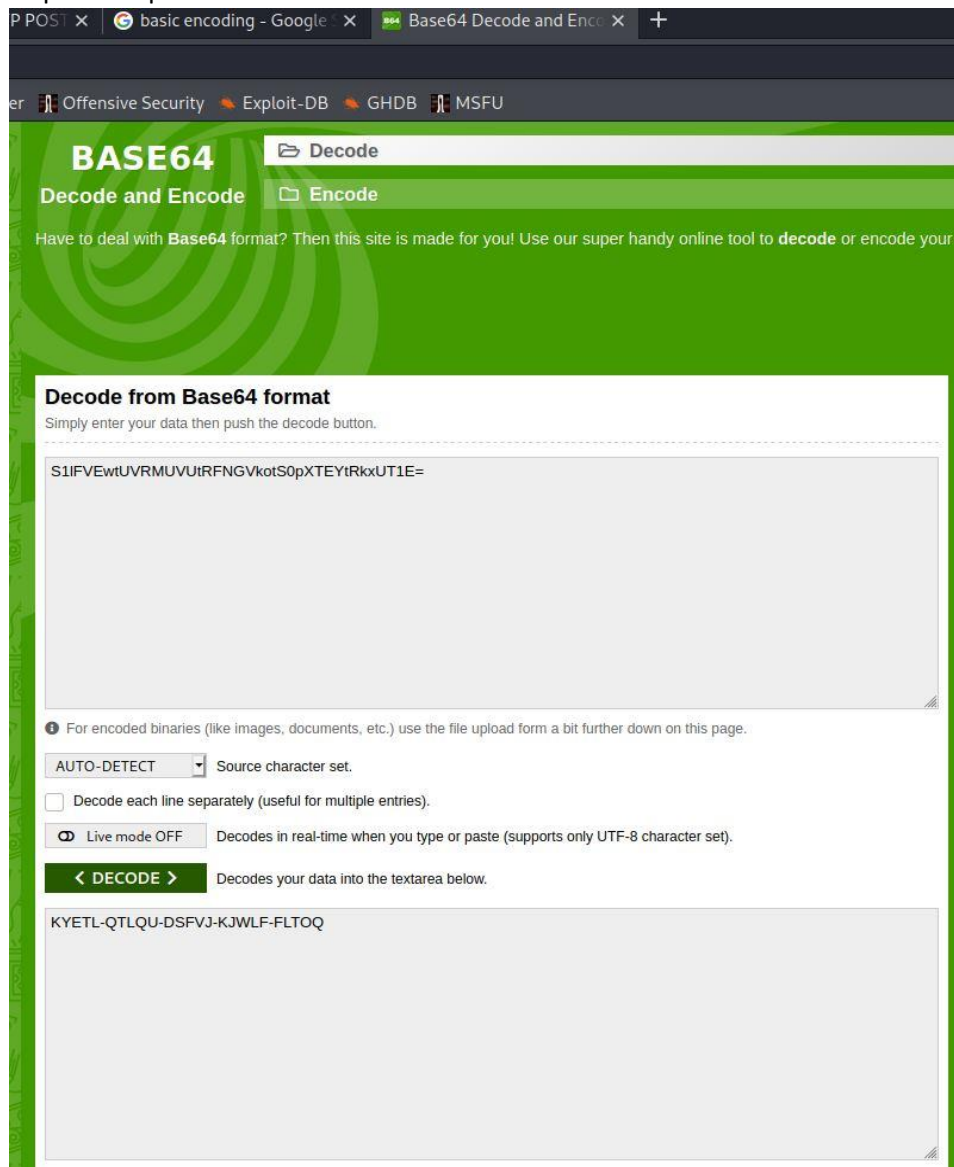
10. Following the instructions in the decoded data I went back to the Console tab and tried entering the command in as a function as it was listed with the other functions. This did not work.



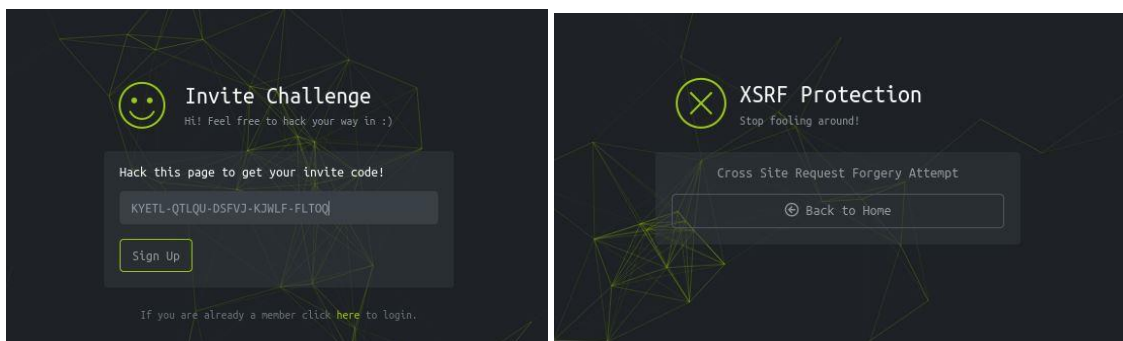
11. I then ran a Google search for making post requests and got back results for online tools that would generate POST requests. I selected one and entered the url and request information, making sure that the output was set to POST.



12. I then took that POST request output and saw that it was encoded. I did a Google search on basic encoding and got back results for Base64 decoders. I picked one and entered my POST request output data.

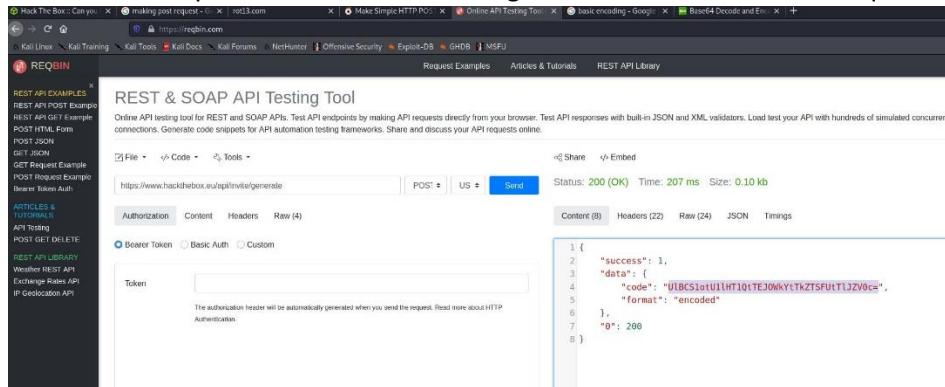


13. The output of that encoded data appeared to be the invite code I was looking for, so I took that code and tried it on the Hack the Box site. It did not work.

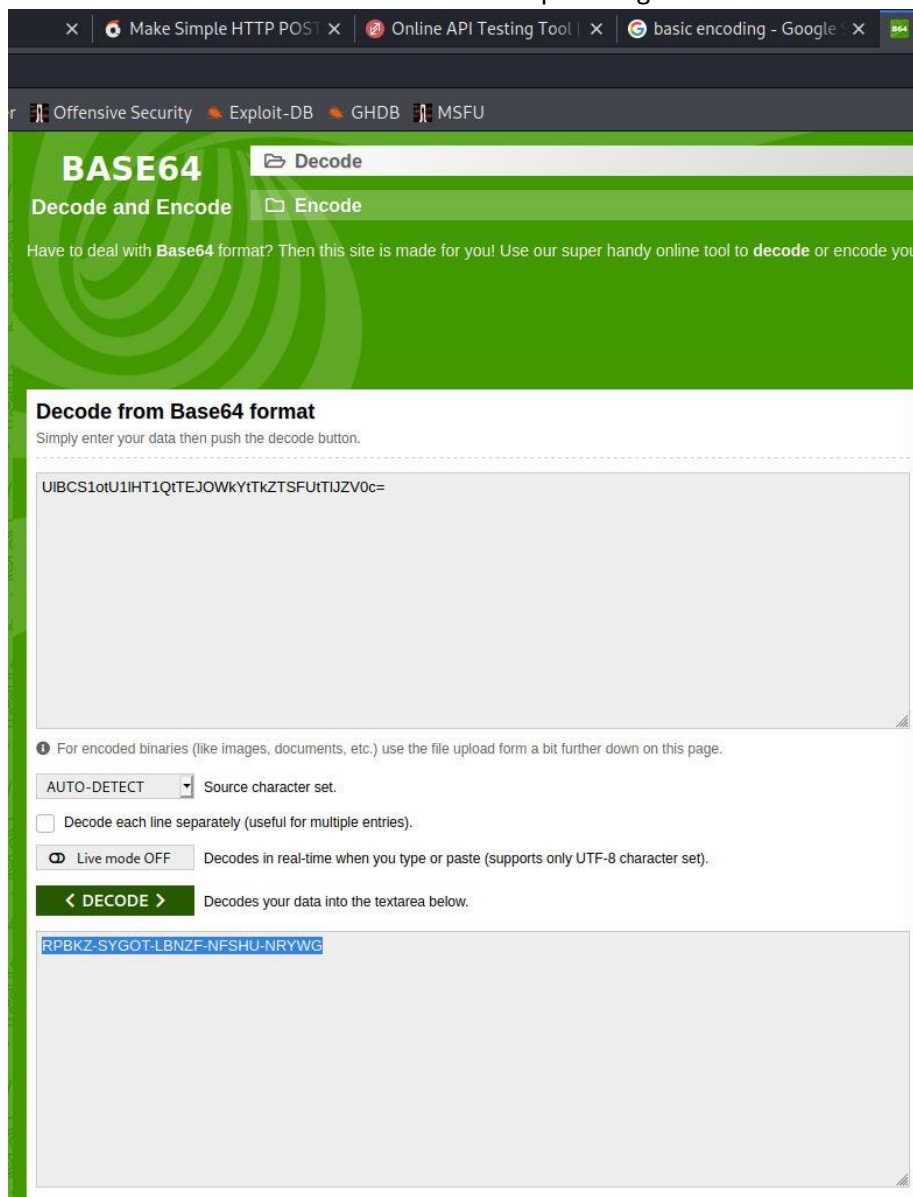




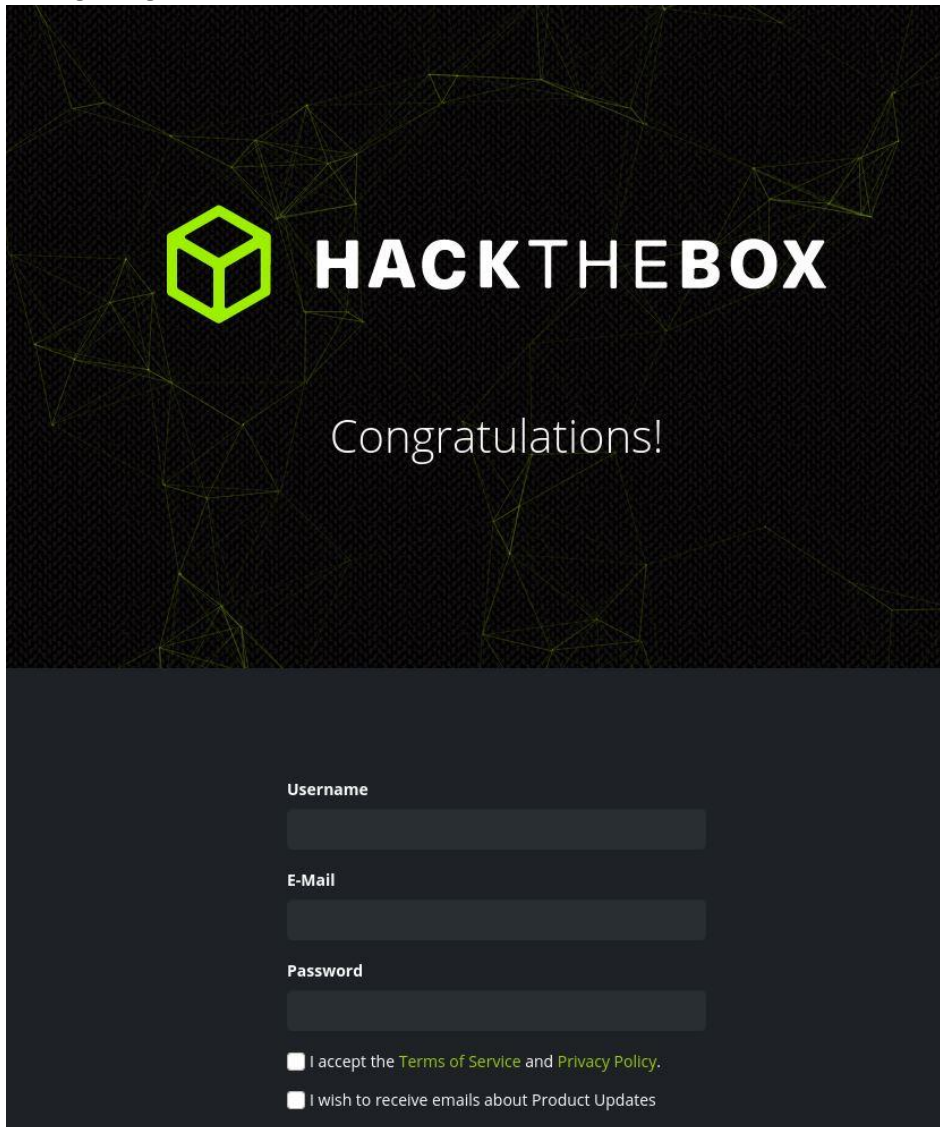
14. I went back to the steps just after I started decrypting and searched for a different POST request tool. I found reqbin.com and used that to generate a new HTTP POST request.




15. From there I went back and decoded the output using the Base64 decoder I used last time.



16. After getting the new invite code I tried this one and it worked.

A registration form for Hack The Box. The top section has a dark background with a green wireframe cube logo and the text "HACKTHEBOX" in white. Below this, the word "Congratulations!" is displayed in white. The bottom section is a dark grey form with three input fields labeled "Username", "E-Mail", and "Password". Below the input fields are two checkboxes: "I accept the Terms of Service and Privacy Policy." and "I wish to receive emails about Product Updates".

 **HACKTHEBOX**

Congratulations!

**Username**

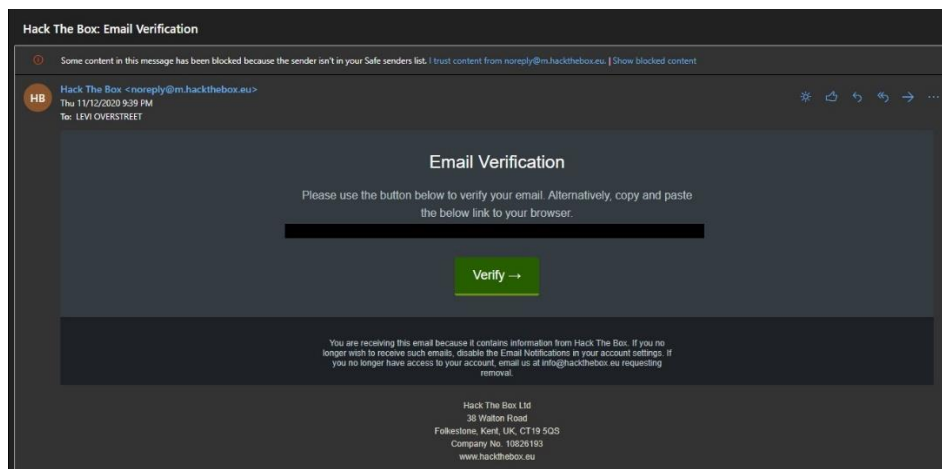
**E-Mail**

**Password**

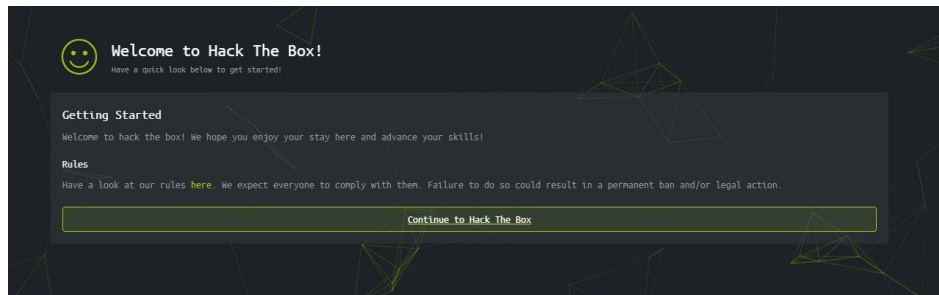
☐ I accept the [Terms of Service](#) and [Privacy Policy](#).

☐ I wish to receive emails about Product Updates

17. From there I entered my registration information and waited for my verification email to be sent.



18. Once I verified my email I logged into [www.hackthebox.com](http://www.hackthebox.com) for the first time to confirm that I had an active account.



## Logs

- 11/12/20 18:05 - Power on Kali VM and logged in
- 11/12/20 18:10 - In web browser, navigate to <https://www.hackthebox.eu/invite>
- 11/12/20 18:13 - Right clicked on webpage and selected option *Inspect Element* and go to *Console* tab
- 11/12/20 18:16 - Found the first clue, need to check the Java scripts that are loaded on page
- 11/12/20 18:17 - Go to *Inspector* tab and right click on first listed Java script, open in *Debugger* tab
- 11/12/20 18:18 - Checked through first Java script, not finding anything out of the ordinary
- 11/12/20 18:24 - Go back to *Inspector* tab and right click on second Java script, open in *Debugger* tab
- 11/12/20 18:25 - Checked through second Java script, found second clue
- 11/12/20 18:30 - Combed through second Java script looking for something out of place, found the next clue
- 11/12/20 18:31 - Using the information from the third clue, go back to *Console* tab
- 11/12/20 18:33 - Type in the function ***makeInviteCode()*** into the Console
- 11/12/20 18:34 - Found the next clue, need to decrypt the data
- 11/12/20 18:36 - Google search on ROT13, selected a decoder
- 11/12/20 18:39 - Copy encrypted data and paste it into the decoder
- 11/12/20 18:41 - Follow instructions in decoded message
- 11/12/20 18:43 - Go back to *Console* tab, type in the function ***POST(/api/invite/generate)*** into the console
- 11/12/20 18:44 - That failed
- 11/12/20 18:46 - Google search on making POST request



11/12/20 18:49 - Found online tool Make simple HTTP POST and GET Requests

11/12/20 18:52 - Input the url and request info, <https://www.hackthebox.eu/api/invite/generate>, making sure the output is set for POST

11/12/20 18:54 - Using the output from the POST request, need to decode the generated code

11/12/20 18:55 - Google search on basic encoding

11/12/20 18:57 - Found a decoder for basic encoding

11/12/20 18:58 - Copied over encoded string into the Base64 decoder

11/12/20 19:00 - After getting the output, it appears to be an input code, copy that code and trying it on the signup page

11/12/20 19:01 - That failed

11/12/20 19:02 - Google search on making POST request, looking for different HTTP POST request online tool

11/12/20 19:08 - Found reqbin.com, input the url and request info, <https://www.hackthebox.eu/api/invite/generate>, making sure the output is set for POST

11/12/20 19:11 - Using the output from the POST request, decoded with Base64 decoder

11/12/20 19:12 - Tried that code, it worked

11/12/20 19:15 - Input my registration info and got confirmation email

11/12/20 19:22 - Logged into hackthebox.com for first time

11/12/20 19:24 - Closed web browser

11/12/20 19:25 - Powered down Kali VM