

Name :- Ronak Kumar

Roll No :- 2015080

FCS Assignment 2

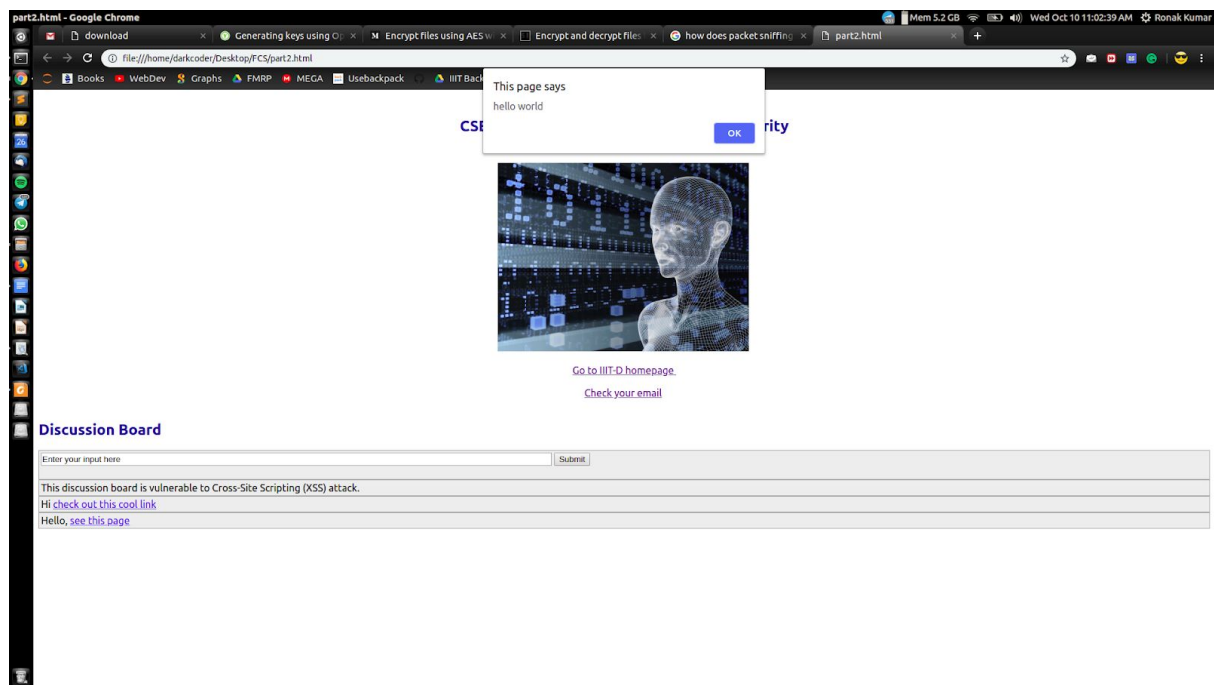
Part I

- 1) IIITD follows username, password authentication technique for successful user login. Any access to the IIITD Wireless Network would be established by the username, password followed by the BSSID (MAC Address) of the user. Whenever, a user wants to login into the network, he needs to input his username and password in the login page of network. The backend system checks whether the username and password is valid and except in the case of mobile phones / tablets, whether the BSSID of the device is registered in the database. Once, all the credentials are verified, the user is granted access to the network. For mobiles / tablets, no username and password is required and the system only checks whether the BSSID is registered for the device or not for successful network login.
- 2) Yes, IIIT-Delhi network is susceptible towards packet sniffing. This is because any intruder / connected user can see the intra network packet flow and sniff it using various available netmon tools like Wireshark etc.
- 3) IIIT-Delhi maintains a login session of 40 minutes per user so that once the time period gets over, he has re-login to verify its identity. In my opinion, the typical internet usage session for students would be around 60 minutes. Any adversary / attacker may sniff the packets and identify high traffic areas. Also, he might capture any sensitive information in the network. Attacker might also harm the valuable assets inside the network.

Part II

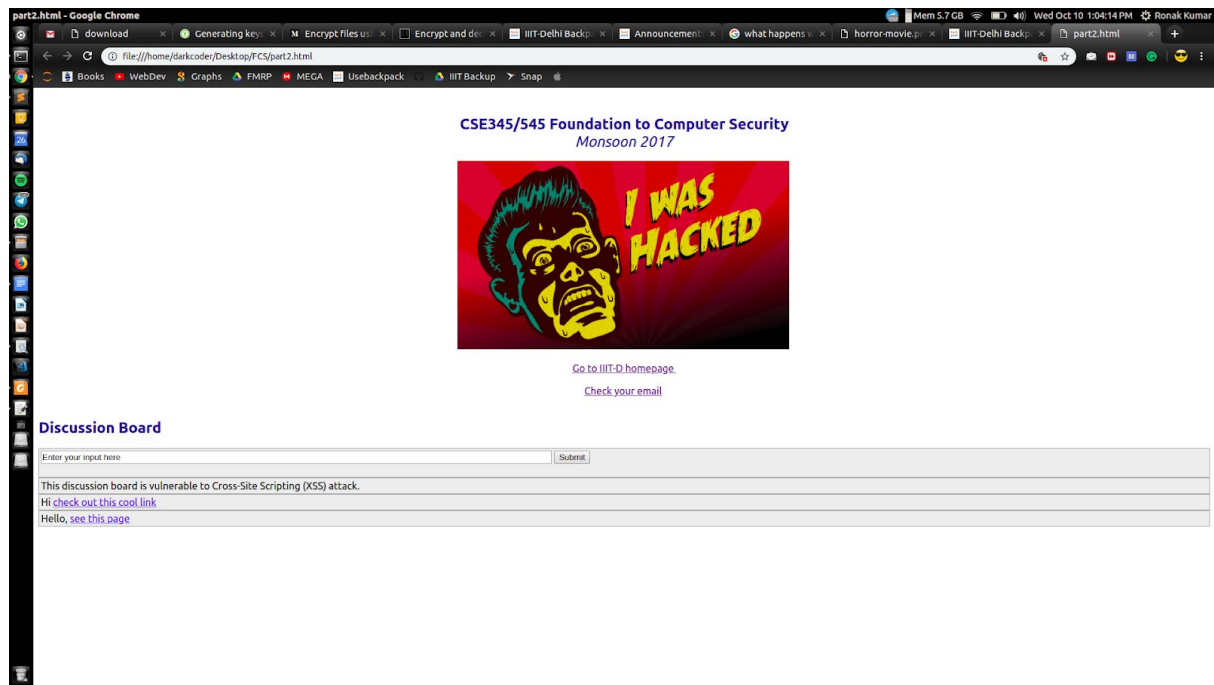
1) Whenever, we write `javascript:alert('hello world');` in the browser URL bar, an alert dialog box pops saying hello world and when we click the OK Button, the box disappears. The Javascript in the URL gets interpreted by the JavaScript Interpreter. Whenever, we have Javascript in our website's code or we write Javascript in the URL as above, the rendering of DOM Tree into a webpage stops and the Javascript is passed to the interpreter. Whenever, the execution of the script finishes, the final HTML page gets rendered.

Screenshot :-



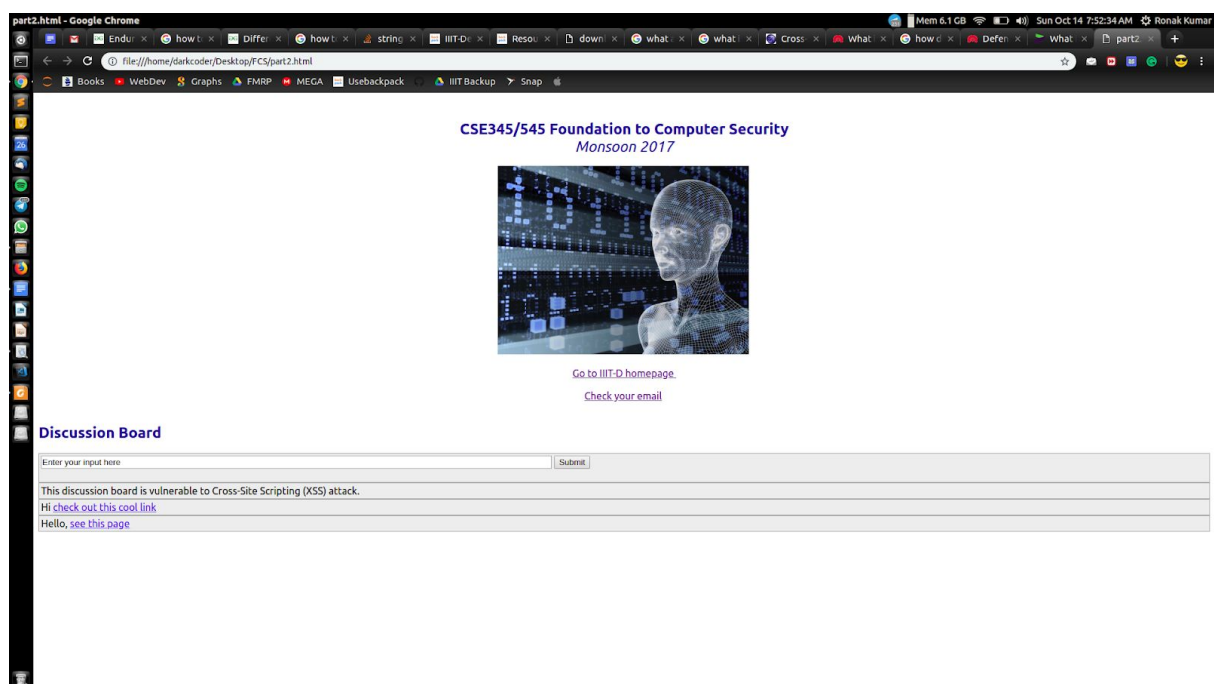
2) When I executed the Javascript snippet in the browser's address bar, the image changed and it now depicted some other image saying I am hacked. However, the previous source code and the source code after executing are the same, therefore there is no change in the source code. So, it is possible to the attacker to change the content of the website by executing any malicious Javascript on the webpage and also without affecting the source code. The script could simply point to some other image or entity which may affect the website's content.

Screenshot :-



3) When I click on the link “check out this cool link” in the Discussion Board, nothing happens because the image gets replaced by the same image as the Javascript only replaces the centre image with itself.

Screenshot :-



The code to change the image would be :-

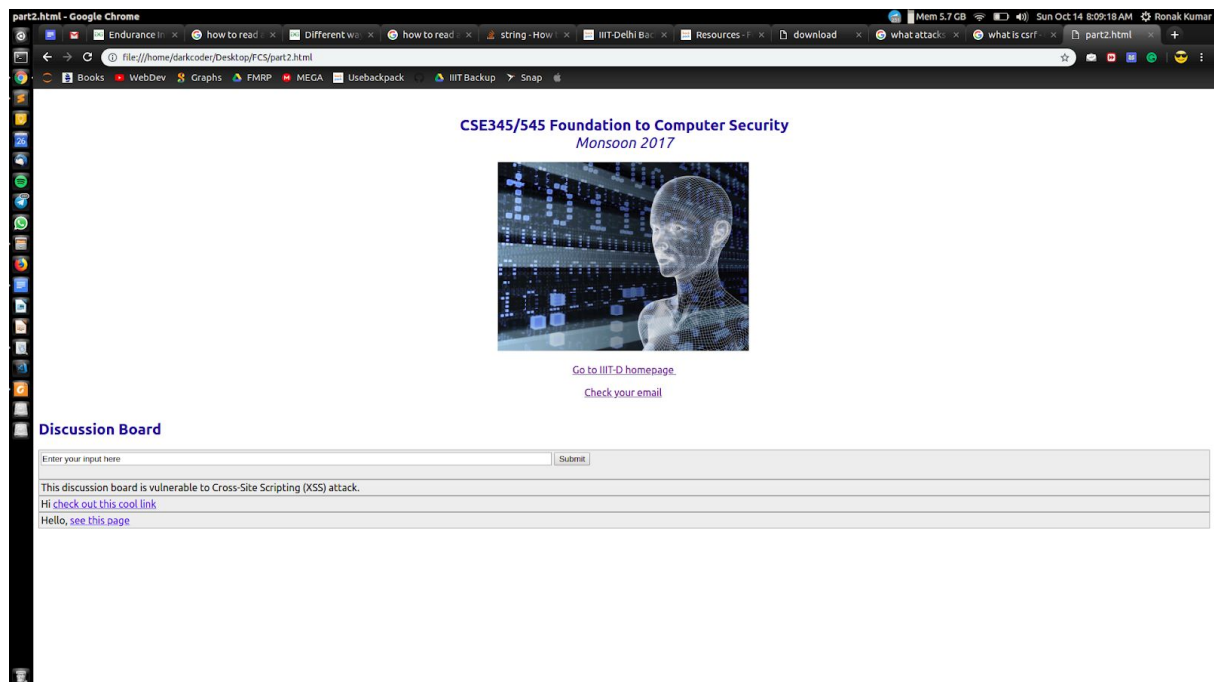
`<a`

4) So, if you want to change the image to some other image as given in the URL, I would add the following script :-

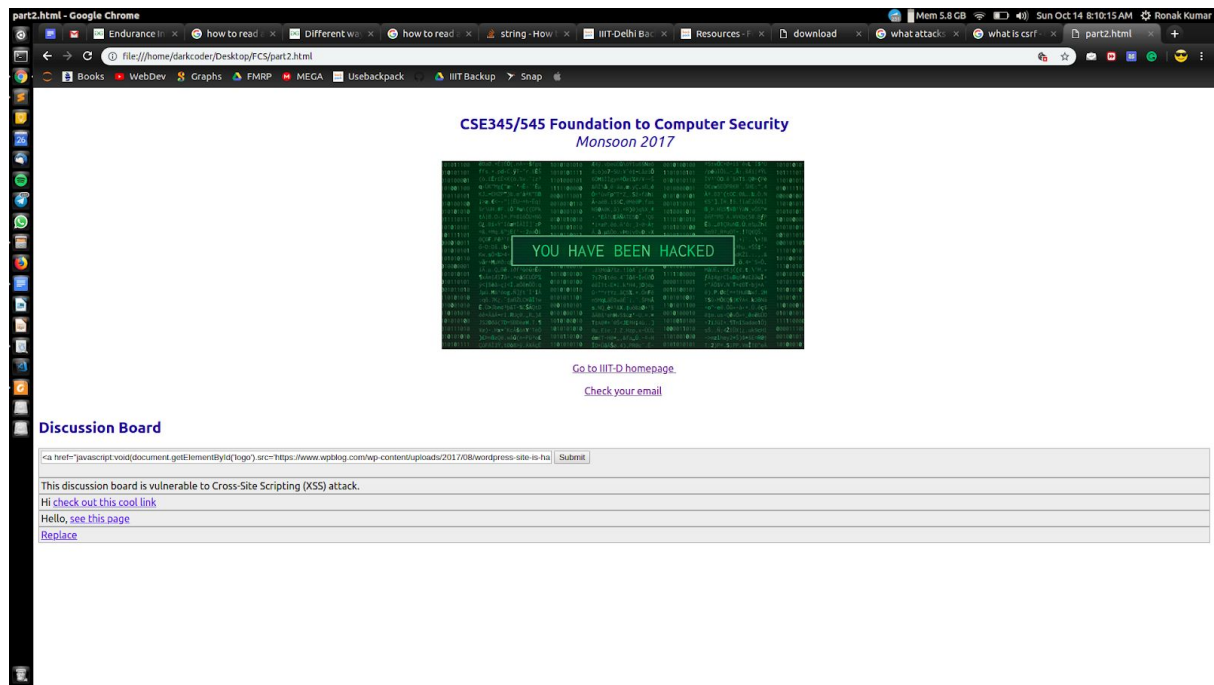
So, basically I have added a URL named Replace which executes the above Javascript. So, it gets the logo using the getElementById Method and then replaces the logo with the image given in the URL.

Screenshots :-

Before Executing the Script :-

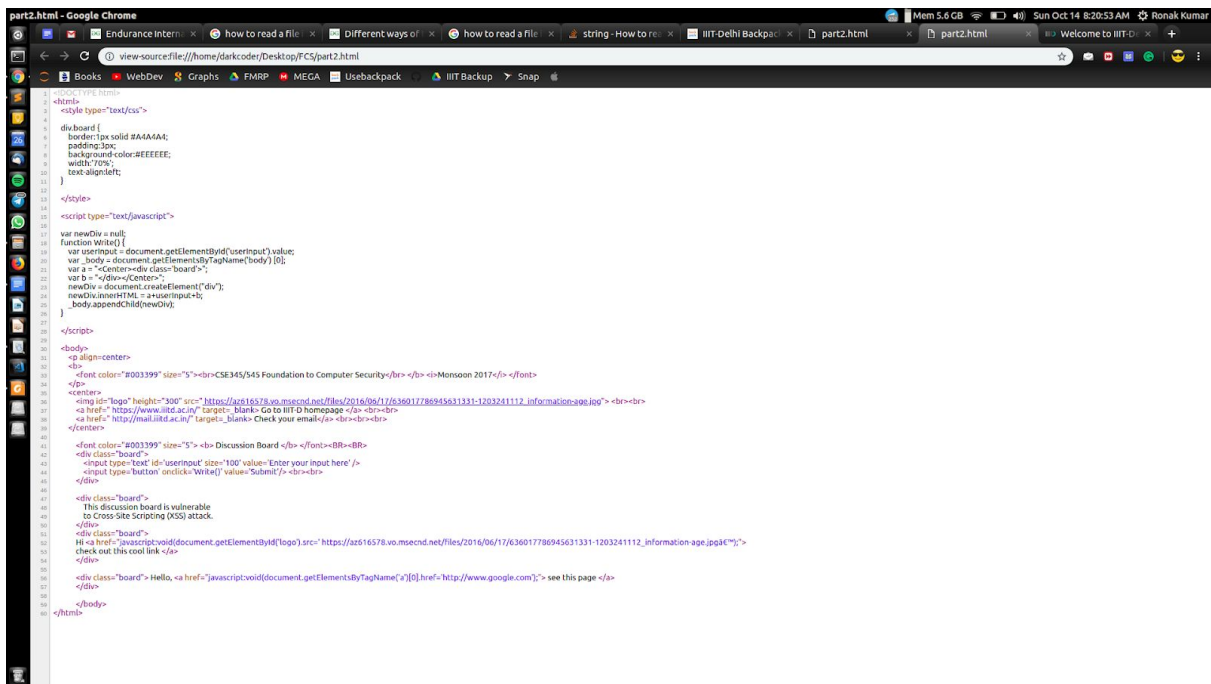
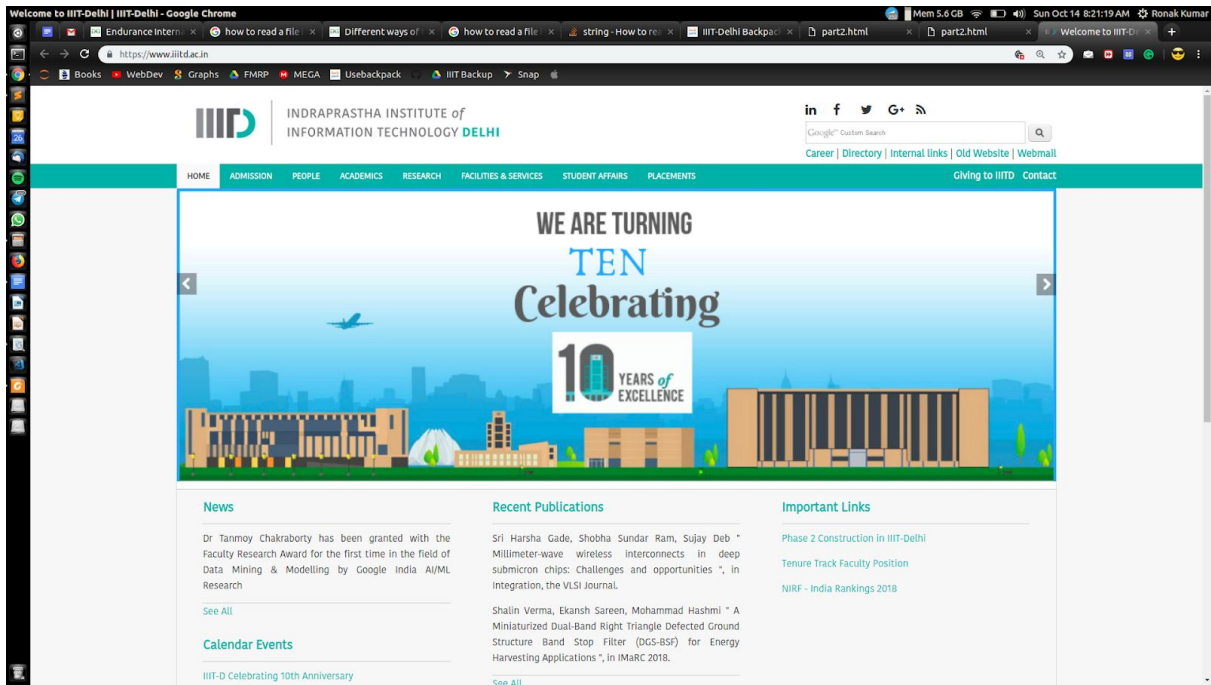


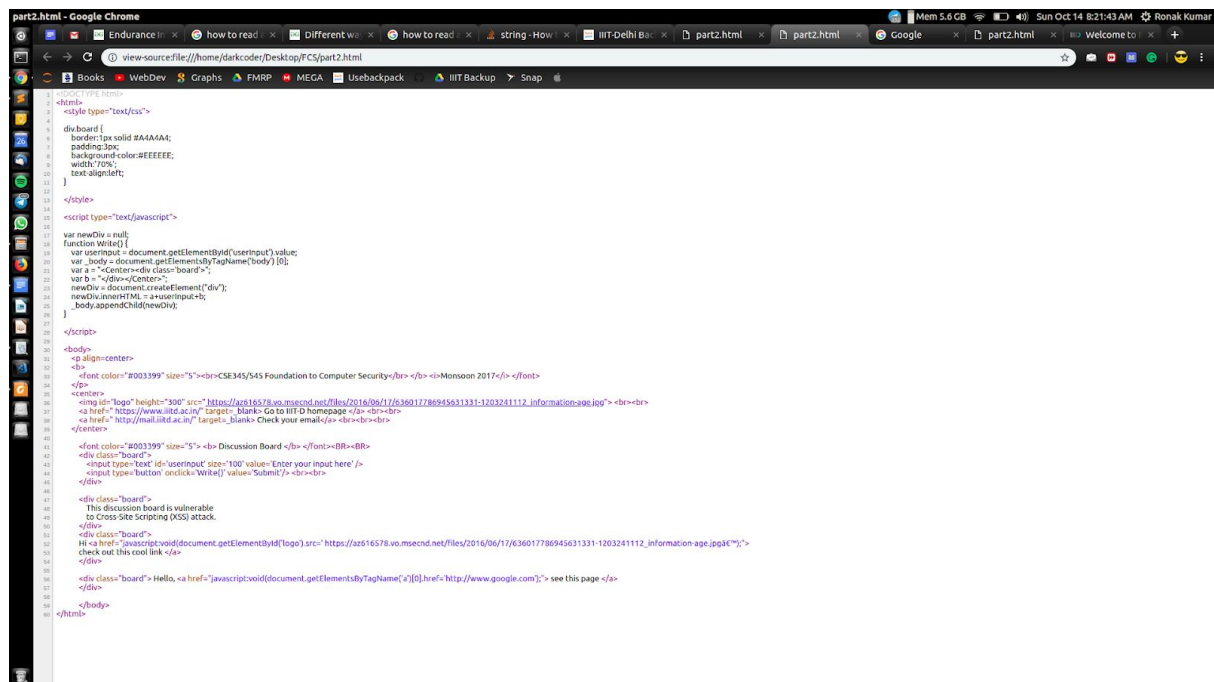
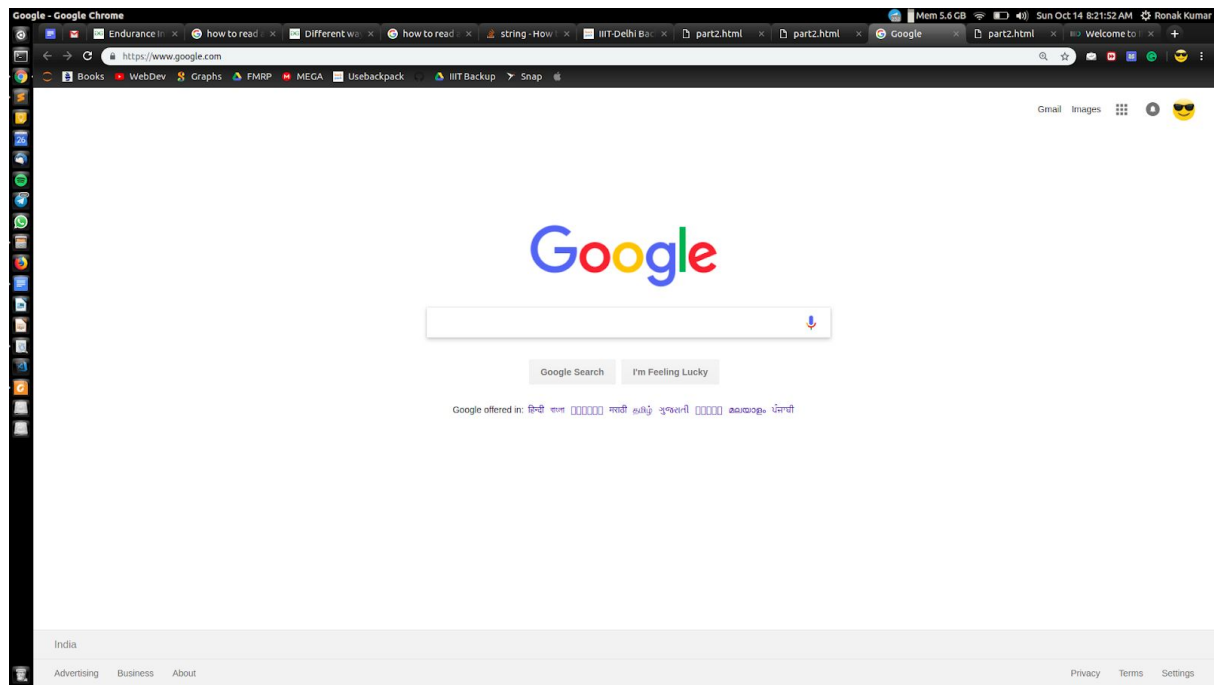
After Executing the Script & Clicking the link :-



5) When, I clicked the link **Go to IIIT-D homepage** for the first time, it redirects me to www.iiitd.ac.in in the next tab. After that if I come to the same page, then click the link **Hello, see this page** and again open the same link **Go to IIIT-D homepage**, it redirects me to www.google.com. This means that the attacker can easily trick the users since on clicking just a single URL, the redirection URL gets changed without changing anything in the source code. Yes, both of the source code matches as there is no difference b/w them before and after clicking on the Hello Link. So, the hacker may easily put some Javascript inside the website which would redirect the user to a fake website that he wants the user to go.

Screenshots :-





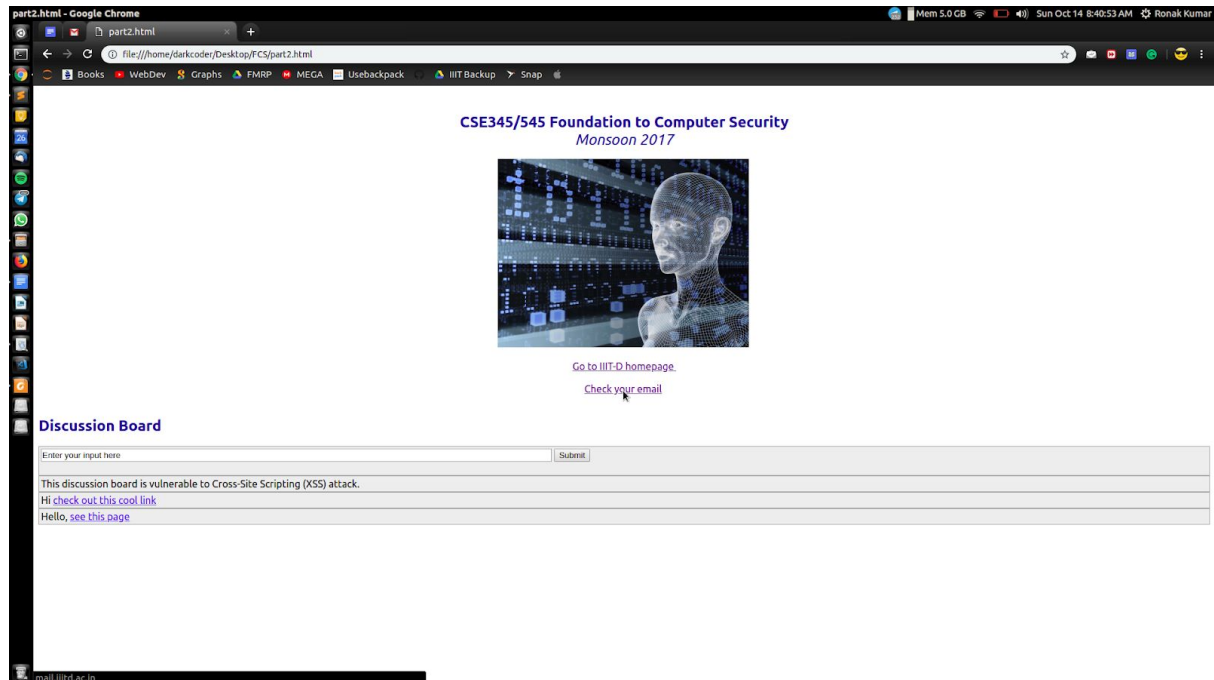
The source code remains the same as in the image.

6) Using this approach of creating a link which apparently would change the URL which was previously used for checking Email, a hacker can easily can create a fake website which matches to the genuine website and get away with his login credentials. In this way, he would conduct any malicious activity by stealing away the login information. I used the following script for changing for doing the required task

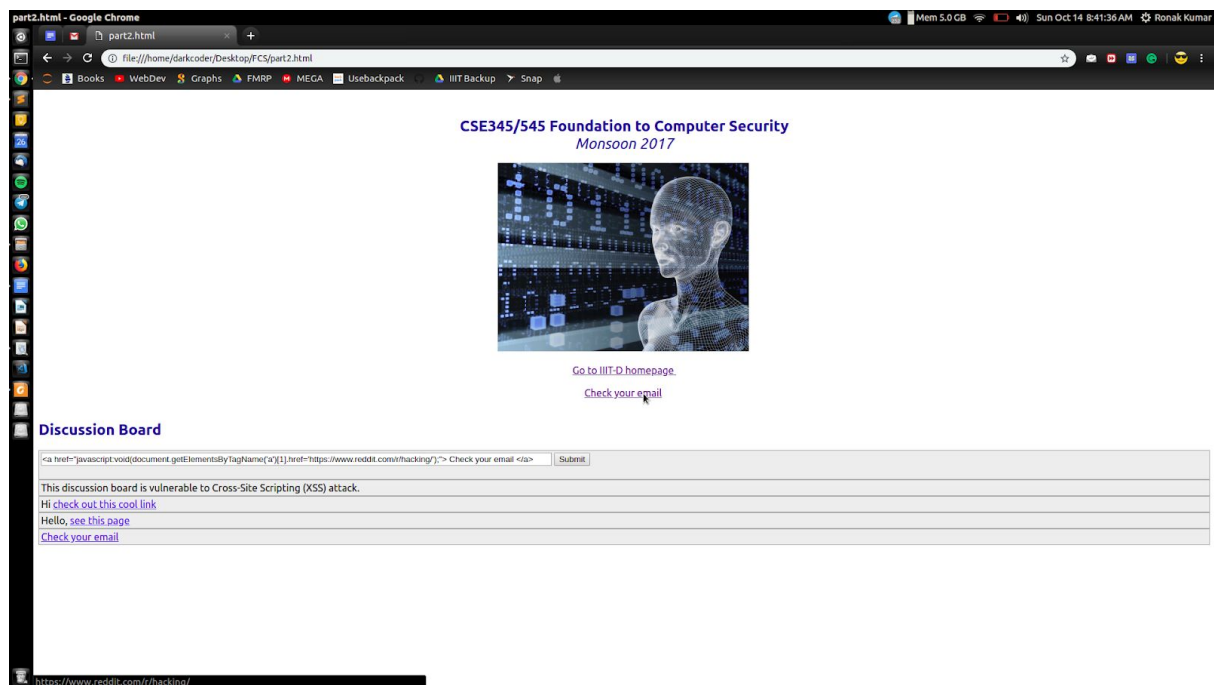
` Check your email `

Screenshots :-

Before making link :-

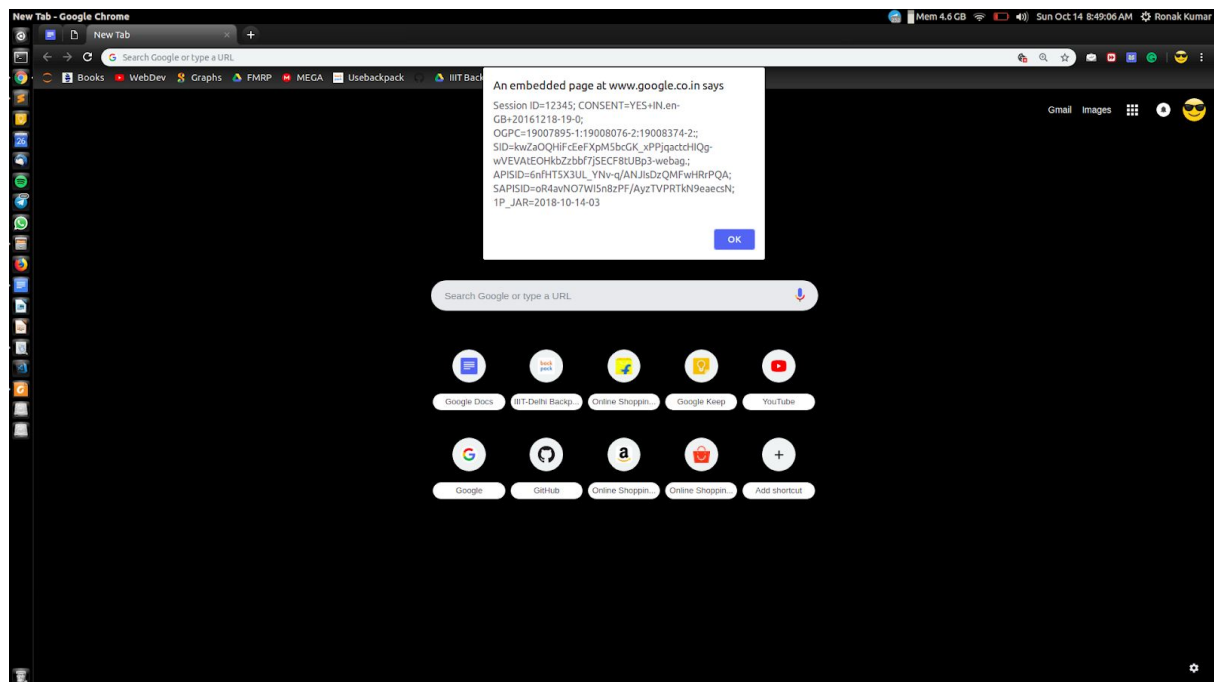


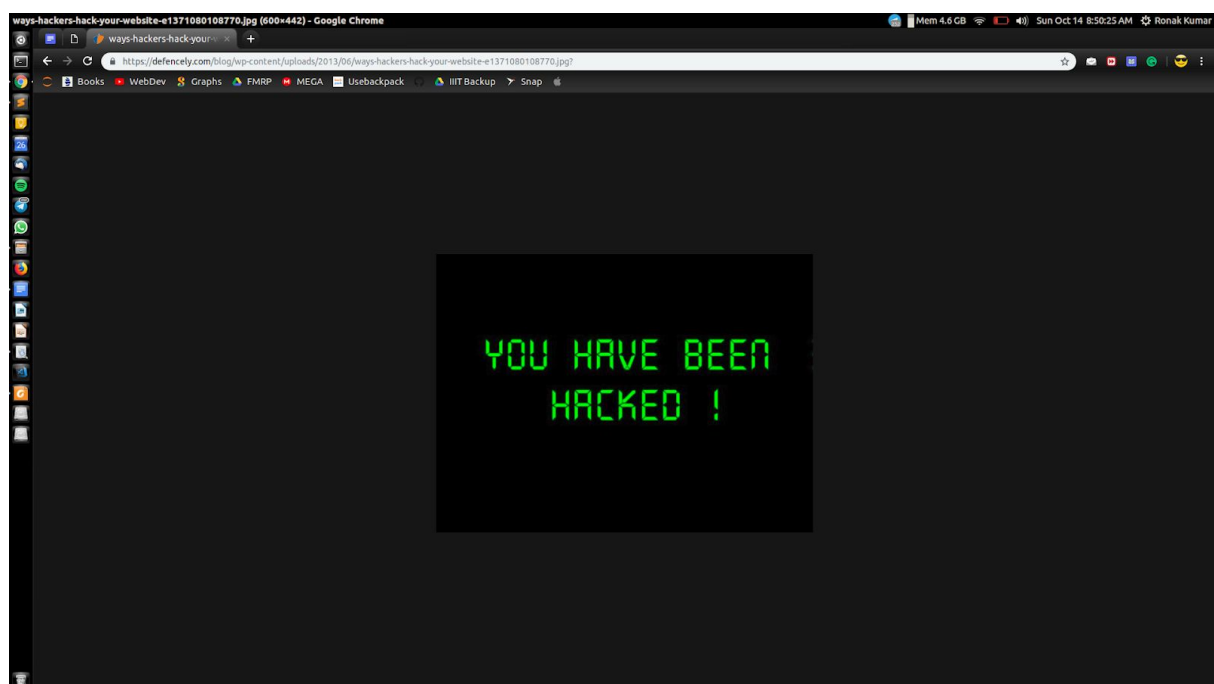
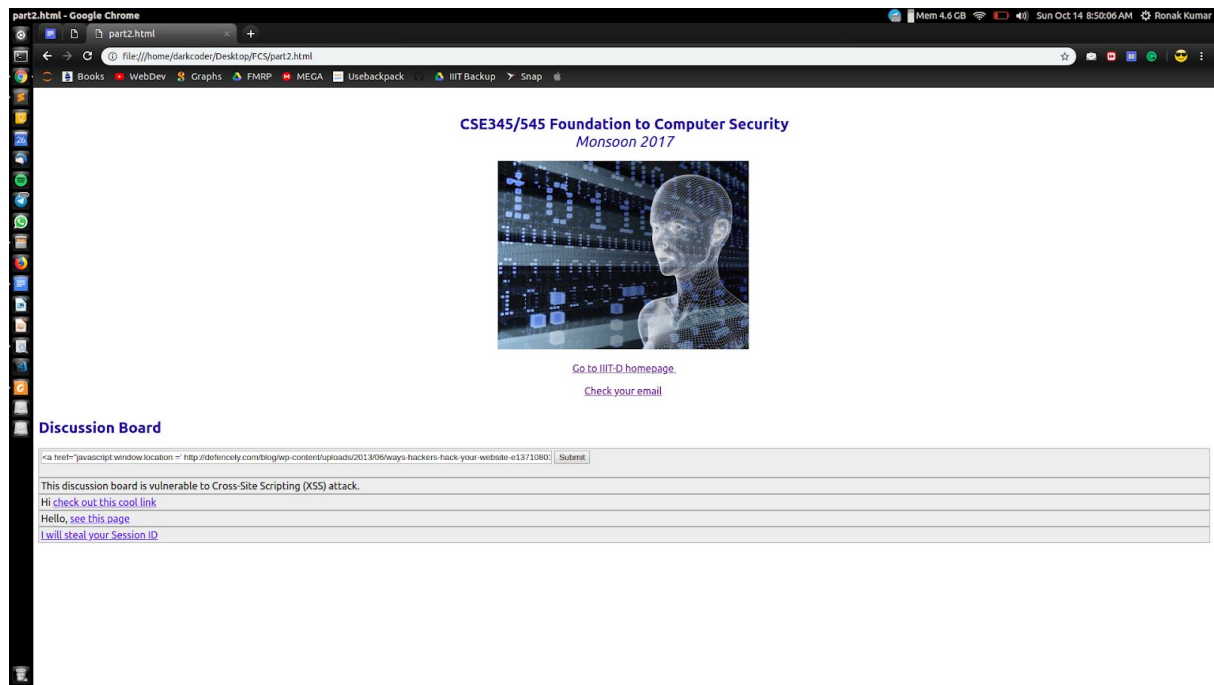
After making link and clicking on the new link :-



7) When, I click on the new link I am redirected to an image saying “YOU HAVE BEEN HACKED !”. So, by this attackers may steal away the session ID of a user with techniques like predicting session token or getting access to session token by running the document.cookie alert function of Javascript. Since, session token is required for the identification of every user’s connections, stealing it can prove to be fatal. Attacker may steal the session ID since it is just comprises of a string of variables which are being used in the URL’s as a cookie. Also, the first screenshot depicts the same.

Screenshots :-





8) Three ways to prevent XSS vulnerabilities are :-

- Escaping is the first method for preventing XSS vulnerabilities. The term, escaping data means that taking the data of an application and securing it before generating any output for the user. By escaping user input, all the key characters in a webpage which are prone for the malicious attack are prevented. However, if the page prevents the users from adding their own code, then all the HTML, URL and JS entities should be escaped.

- Validating Input is the second way for preventing XSS vulnerabilities. It means that the application is rendering doesn't render any false data which prevents any malicious attack to database, application and the current users. This is often done by using the technique of blacklisting, avoiding bad characters and by only whitelisting good characters throughout the website.
- Sanitizing is the third method for preventing XSS vulnerabilities. It is a good defense mechanism however it shouldn't be the only one. This method becomes helpful on websites that allows HTML markup to ensure that the data received cannot harm any users and also to the entire database by changing the input for any adversary to an acceptable format.

We can use input validation for preventing XSS vulnerability. In the Write() method of the Javascript, we can check the input and make changes when receiving the userInput.

The function to add would be :-

```
function transform(user_input) {
    "<" : "&lt;",
    ">" : "&gt;",
    "." : "&#46;",
    ":" : "&#44;",
    " " : "&#32;"
}
```

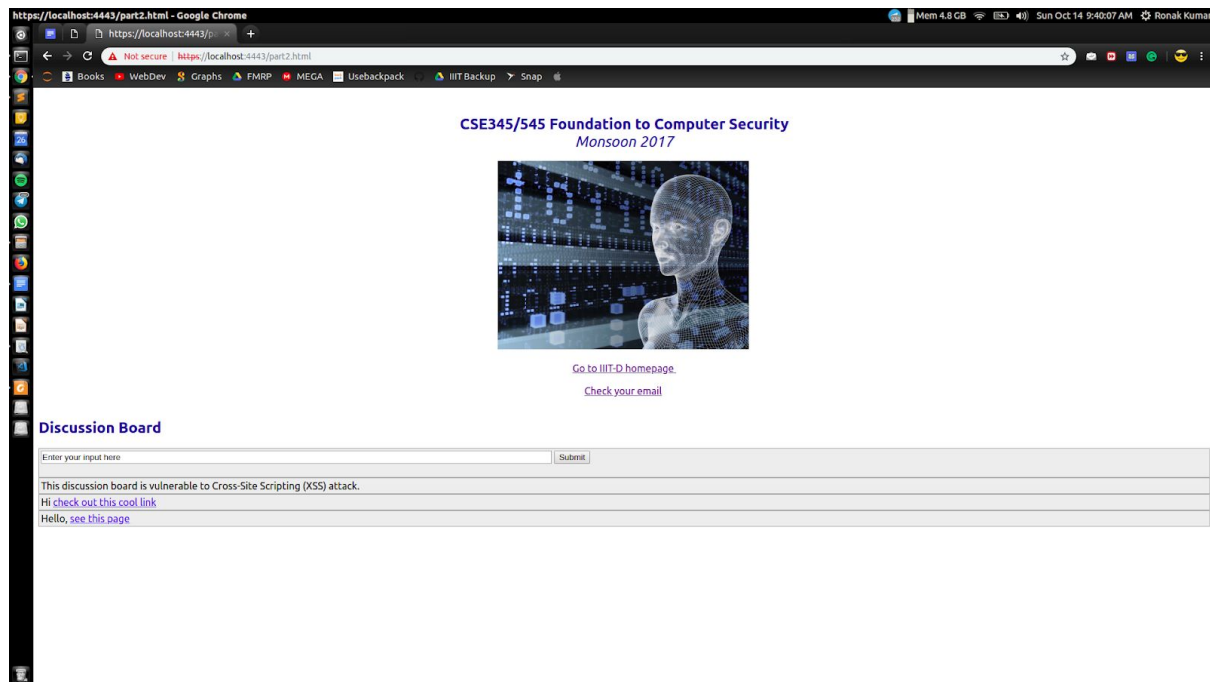
We would call this passing the userInput ID value to the userInput variable.

9) Self signed certificates using OpenSSL are created like :-

```
openssl req -new -x509 -keyout server.pem -out server.pem -days 365 -nodes
```

The code is given in the folder

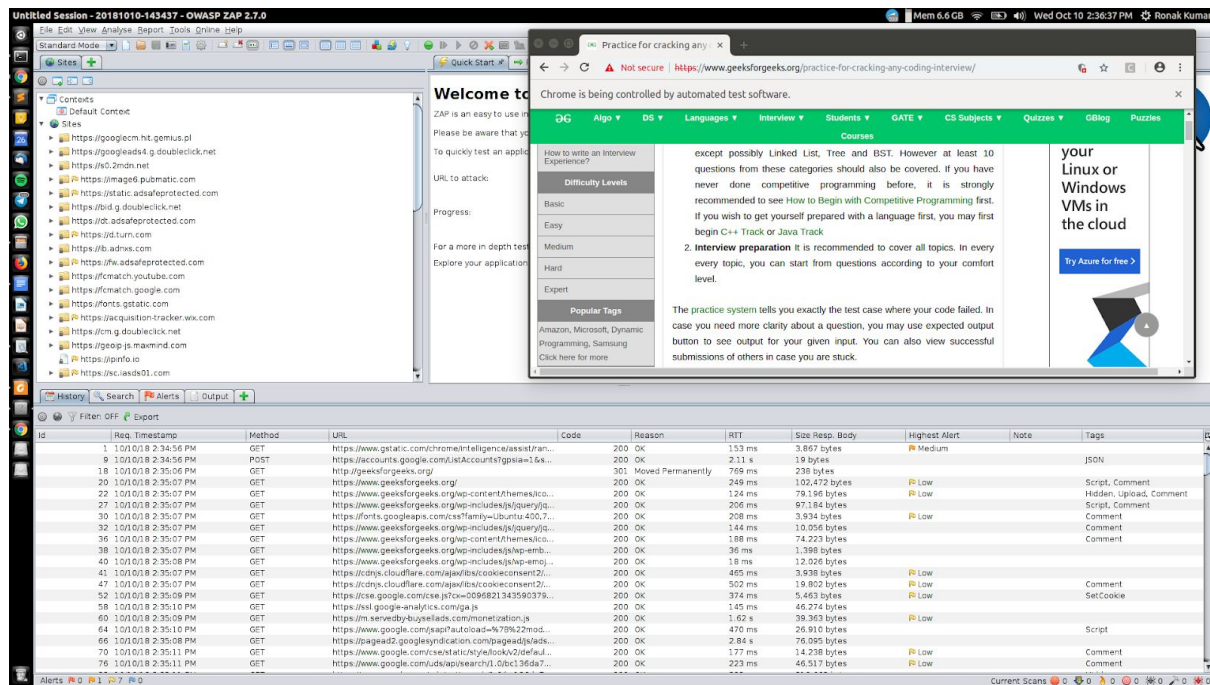
Screenshot :-



10) No, I don't think that using HTTPS alone would make the security of the system good. HTTPS only uses SSL/TLS for providing encrypted communication between the user and the server. However, it doesn't prevent any kind of code injection attacks like XSS. If the system only uses HTTPS as a security medium, then the hackers can still access confidential information and steal user's data by executing malicious scripts. HTTPS doesn't holistically secure the entire system.

Part III

1) I have used the OWASP ZAP as the web security tool for intercepting the client and server requests and replies. I visited www.geeksforgeeks.org for capturing server and my request. The kind of information that was displayed included methods used which include GET and POST. The code for the headers and the cookies was also mentioned if we search for it in the search tab. URL was also present. RTT was mentioned along with the various alert methods like Low, Medium and High. Lastly, tags were also present indicating JSON, Comment, Script etc.



2) Three vulnerabilities that may occur by exploiting the data would be :-

i) MIMA :- Since, using this tool we are aware of the type of methods being used in the communication between the user and the server, and also the level of security, we may easily use the technique of sniffing over the network and try to break the weakest links in the network.

ii) Cross Site Scripting & CSRF :- Since, we are aware of the information in the URL's and the methods that are used for communication, we can easily inject malicious JavaScript to steal particular user's information. Also, since the attacker knows the methods of communication like GET and POST and also the URL's which are requested, he may study the pattern and alter the GET request URL to steal user's information or perform some malicious information.

iii) Phishing :- Since, the tool allows us to capture information like cookies, header it would be easy for the hacker to extract personal information from them. In that case, he might just send the fake website to the user pretending to be a real one and the user may become a victim of phishing.

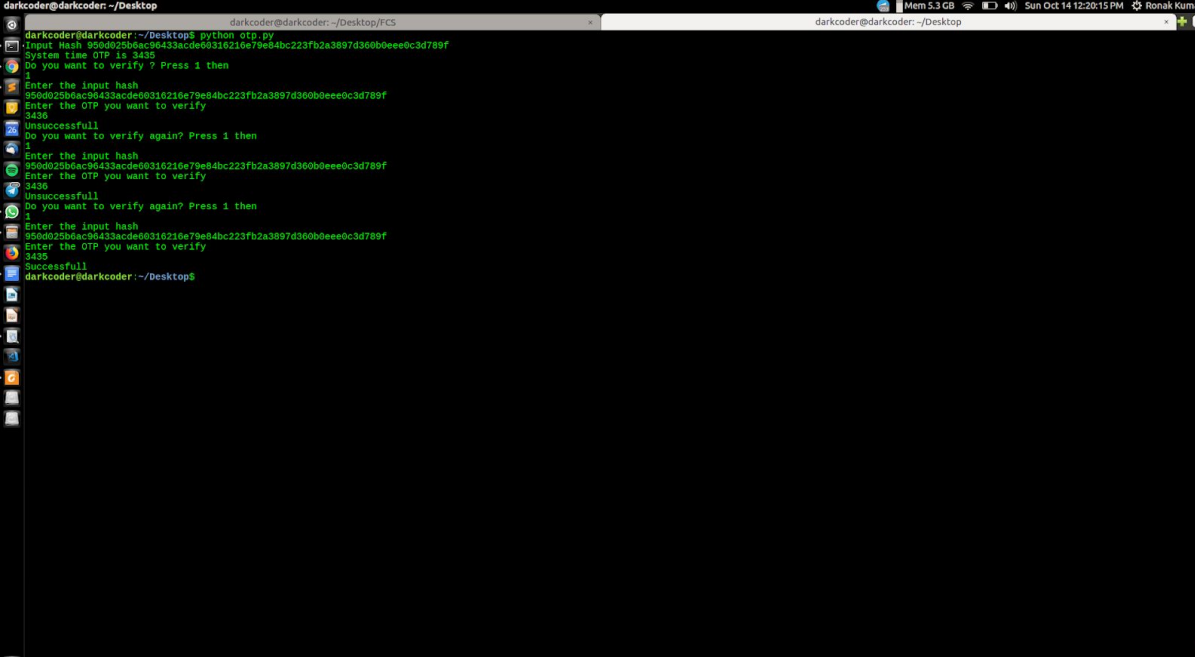
3) If I were a developer, first of all I would try to implement HTTPS in my website. It would encrypt all the information that is being transferred between the user and the server. It would support SSL/TLS and the PKI system. So, whenever the user requests for a website, SSL handshake occurs and the use of certificates would come into place. Secondly, I would also do input validation for preventing scripting attacks like XSS. Also, any state changing operation in the website, should use a secure random token with a particular TTL and also encrypted with strong encryption techniques for preventing CSRF Attacks. On the user side, phishing can be avoided

by using Firewalls whereas on the server side. Using secure communication as in the above example like of HTTPS would make the network away from phishing since the attackers cannot send any fake email containing any malicious entity. SSL/TLS would secure the handshake process so man in the middle attacks can be avoided. Escaping the characters, sanitizing and input validation would prevent XSS since any intruder cannot inject code in this way.

Part IV

- 1)
 - a) The code is given in the folder
 - b) I think it might be vulnerable to depict the OTP. Since, the SHA-256 produces a hashing of 64 bits, I used the indices 15,31,37,63 of the hash string with 0 based indexing approach. The attacker may guess the indices for the hash string and may easily generate the OTP. However, for guessing which indices to choose, it may take some time for him. I chose this type of method as it is easy to encrypt and generate OTP. If the characters present in these indices are numbers, then I simply append the number value of the character to the OTP, else I get the ASCII value of the character, subtract it with 97 (i.e ASCII value of 'a') and append the value into the final OTP.

Screenshot :-



```
darkcoder@darkcoder: ~/Desktop
darkcoder@darkcoder:~/Desktop$ python otp.py
Input Hash 950d025b0ac96433acde00310216e79e84bc223fb2a3897d360b0eee0c3d789f
System time OTP is 3435
Do you want to verify ? Press 1 then
1
Enter the input hash
950d025b0ac96433acde00310216e79e84bc223fb2a3897d360b0eee0c3d789f
Enter the OTP you want to verify
3436
Unsuccessful
Do you want to verify again? Press 1 then
1
Enter the input hash
950d025b0ac96433acde00310216e79e84bc223fb2a3897d360b0eee0c3d789f
Enter the OTP you want to verify
3435
Unsuccessful
Do you want to verify again? Press 1 then
1
Enter the input hash
950d025b0ac96433acde00310216e79e84bc223fb2a3897d360b0eee0c3d789f
Enter the OTP you want to verify
3435
Successful
darkcoder@darkcoder:~/Desktop$
```

- 3)

The commands for building the source code of TOR are :-

```
tar -xvf tor-0.3.4.8.tar.gz
```

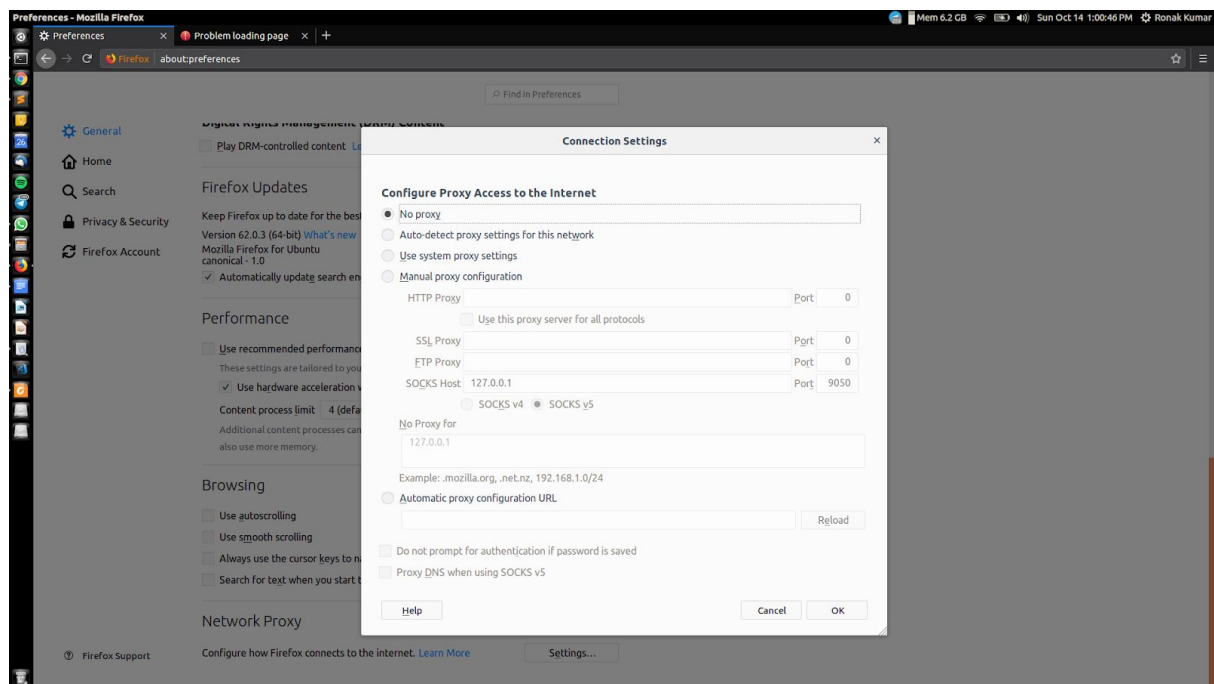
```
cd tor-0.3.4.8/
```

```
./configure
```

```
make && src/or/tor
```

Screenshots :-

Before TOR



What is My IP Address?

Google AdSense

Monetize your site With ads you can count on

Learn More

Your IP Address is **43.231.58.170**. [Hide IP with VPN](#)

This is the public IP address of your computer. If your computer is behind a router or used a proxy server to view this page, the IP address shown is your router or proxy server.

Do you want to find an IP address of your network printer? Please read [How to find an IP of a printer](#) to find ways to obtain an IP number of your network printer.

Do you want to find IP Addresses of private network? Please read [How to find IP addresses of computing devices on the private network?](#)

Ads by Google

Check Your IP

All IP

Address Finder Map

IP Address Details

IP Address	43.231.58.170 [Hide this IP with VPN]
IP Location	Dehi, Dehi (IN) [Details]
Proxy	43.231.58.170, 198.143.34.220
Device Type	Linux
OS	Ubuntu
Browser	Firefox
User Agent	Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0
Screen Size	1920px X 1080px
Cookie	Enabled
Javascript	Enabled

SEARCH OUR WEBSITE

Google Custom Search

Search

IP TOOLS

- Trace Email Source
- Verify Email Address
- Proxy Check
- Subnet Calculator

DOMAIN TOOLS

- Who is Hosting a Website
- Alexa Traffic Rank Checker
- Domain Age Checker
- Reverse DNS Lookup
- HTTP Server Header Check

ADVERTISEMENT

YU ACE

The Big Billion Days Exclusive Offer 11-14th Oct, 2018

₹ 5,499/- onwards

Buy Now

IP ARTICLES

- What is an IP Address?
- What is an IPv6 Address?
- What is the Internet?
- What is an Intranet?
- What is a Subnet Mask?
- What is a MAC address?

After TOR

Firefox Updates

Keep Firefox up to date for the best performance and security. Version 62.0.3 (64-bit) [What's new](#) Mozilla Firefox for Ubuntu canonical - 1.0

Automatically update search engines

Performance

Use recommended performance settings. These settings are tailored to your system. Use hardware acceleration when available.

Content process limit: 4 (default). Additional content processes can also use more memory.

Browsing

Use autoscrolling

Use smooth scrolling

Always use the cursor keys to move the cursor

Search for text when you start typing

Network Proxy

Configure how Firefox connects to the internet. [Learn More](#) [Settings...](#)

Connection Settings

Configure Proxy Access to the Internet

☐ No proxy

☐ Auto-detect proxy settings for this network

☐ Use system proxy settings

☒ Manual proxy configuration

HTTP Proxy Port

☐ Use this proxy server for all protocols

SSL Proxy Port

FTP Proxy Port

SOCKS Host: 127.0.0.1 Port: 9050

☐ SOCKS v4 ☒ SOCKS v5

No Proxy for: 127.0.0.1

Example: mozilla.org, net.nz, 192.168.1.0/24

☐ Automatic proxy configuration URL [Reload](#)

☐ Do not prompt for authentication if password is saved

☐ Proxy DNS when using SOCKS v5

[Help](#) [Cancel](#) [OK](#)

What is my IP Address?

Your IP Address is **171.25.193.77**. [Hide IP with VPN](#)

This is the public IP address of your computer. If your computer is behind a router or used a proxy server to view this page, the IP address shown is your router or proxy server.

Do you want to find an IP address of your network printer? Please read [How to find an IP of a printer](#) to find ways to obtain an IP number of your network printer.

Do you want to find IP Addresses of private network? Please read [How to find IP addresses of computing devices on the private network](#)

As by Google [Check Your IP](#) [Get Software](#) [Company of Software](#)

IP Address Details

IP Address	171.25.193.77 [Hide this IP with VPN]
IP Location	Copenhagen, Hovedstaden (DK) [Details]
Host Name	for-exit-readme.dti.se
Proxy	171.25.193.77, 198.143.34.220
Device Type	Linux
OS	Ubuntu
Browser	Firefox
User Agent	Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0
Screen Size	1920px X 1080px
Cookie	Enabled
Javascript	Enabled

SEARCH OUR WEBSITE

Google Custom Search

IP TOOLS

- Trace Email Source
- Verify Email Address
- Proxy Check
- Subnet Calculator

DOMAIN TOOLS

- Who is Hosting a Website
- Alexa Traffic Rank Checker
- Domain Age Checker
- Reverse DNS Lookup
- HTTP Server Header Check

ADVERTISEMENT

WAN Optimization, DDoS Mitigation, Cloud Managed, Firewall, Load Balancing, GET A DEMO, Trusted by 5M+ Sites Globally, #1 DDoS Attack Protection

IP ARTICLES

- What is an IP Address?
- What is an IPv6 Address?
- What is the Internet?
- What is an Intranet?
- What is a Subnet Mask?
- What is a MAC address?
- What is an Ethernet?
- What is a TCP/IP?

This shows that TOR perfectly works. The website shows the change in IP Address with the change in location from New Delhi to Copenhagen

4)

Screenshots :-

Stream Cipher RC4

```

darkcoder@darkcoder: ~/Desktop/FC5
darkcoder@darkcoder:~/Desktop/FC5$ time openssl enc -rc4 -in large_file.txt -out rc4_encryption.txt.enc
enter rc4 encryption password:
Verifying - enter rc4 encryption password:

real    0m2.698s
user    0m0.103s
sys     0m0.040s
darkcoder@darkcoder:~/Desktop/FC5$

```

AES-256

```
darkcoder@darkcoder: ~/Desktop/FCS$
darkcoder@darkcoder:~/Desktop/FCS$ time openssl enc -aes-256-cbc -in large_file.txt -out aes256_encryption.txt.enc
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
real    0m0.901s
user    0m0.978s
sys     0m0.041s
darkcoder@darkcoder:~/Desktop/FCS$
```

RSA

```
darkcoder@darkcoder:~/Desktop/FCS/44/44$
darkcoder@darkcoder:~/Desktop/FCS/44/44$ time openssl genrsa -out private.pem 8196
Generating RSA private key, 8196 bit long modulus
.....++
e is 65537 (0x10001)
real    0m0.209s
user    0m0.194s
sys     0m0.028s
darkcoder@darkcoder:~/Desktop/FCS/44/44$ time openssl rsa -in private.pem -out public.pem -outform PEM -pubout
writing RSA key
real    0m0.014s
user    0m0.014s
sys     0m0.000s
darkcoder@darkcoder:~/Desktop/FCS/44/44$ time openssl rand -base64 32 | cut -c1-31 > aes_key.txt
real    0m0.016s
user    0m0.015s
sys     0m0.019s
darkcoder@darkcoder:~/Desktop/FCS/44/44$ time openssl enc -aes-256-cbc -salt -in large_file.txt -out large_file_output.enc -pass file:./aes_key.txt
real    0m0.106s
user    0m0.097s
sys     0m0.039s
darkcoder@darkcoder:~/Desktop/FCS/44/44$ time openssl rsauti -encrypt -inkey public.pem -pubin -in aes_key.txt -out aes_key.txt.crypted
real    0m0.007s
user    0m0.003s
sys     0m0.004s
darkcoder@darkcoder:~/Desktop/FCS/44/44$ ls
aes_key.txt  aes_key.txt.crypted  large_file_output.enc  large_file.txt  private.pem  public.pem
darkcoder@darkcoder:~/Desktop/FCS/44/44$
```

So, if we compare all the three encryption standards as in the screenshots we see that the stream cipher RC4 and the symmetric cipher AES-256 approximately takes equal time. Whereas, when the encryption is being done using RSA, it is incredibly fast. This less time consumption of time except for generating the first RSA key is because RSA itself is incapable of encrypting large files. It uses the AES algorithm for encryption which being a symmetric algorithm makes the RSA encryption fast. Although, the RSA algorithm is an asymmetric algorithm which uses two different keys, but doesn't require both of them to be shared. However, this increases the overall time complexity but since both of the keys need not be shared, this would be much secure for information exchange.

Reference taken from this <https://gist.github.com/carlj/6509821>