Royal University of Bhutan

འབྲུག་རྒྱལ་འཛིན་གནོན་ལག་སློབ་སྡེ།

College of Science and Technology
Rinchending: Bhutan

Continuous Practical Assignment2
Bug Bounty
SWS101
Introduction to Cybersecurity
(SS2024)

CAP{No.2} Report

Submitted By;
Student Name: Sonam Tenzin
Enrollment No.: 02230300
Programme: BESWE
Date: 31/05/2024

RUB Wheel of Academic Law: Academic Dishonesty

Section H2 of the Royal University of Bhutan's *Wheel of Academic Law* provides the following definition of academic dishonesty:

Academic dishonesty may be defined as any attempt by a student to gain an unfair advantage in any assessment. It may be demonstrated by one of the following:

1. **Collusion:** the representation of a piece of unauthorized group work as the work of a single candidate.
2. **Commissioning:** submitting an assignment done by another person as the student's own work.
3. **Duplication**: the inclusion in coursework of material identical or substantially similar to material which has already been submitted for any other assessment within the University.
4. **False declaration**: making a false declaration in order to receive special consideration by an Examination Board or to obtain extensions to deadlines or exemption from work.
5. **Falsification of data**: presentation of data in laboratory reports, projects, etc., based on work purported to have been carried out by the student, which has been invented, altered or copied by the student.
6. **Plagiarism**: the unacknowledged use of another's work as if it were one's own.

Examples are:

- verbatim copying of another's work without acknowledgement.
- paraphrasing of another's work by simply changing a few words or altering the order of presentation, without acknowledgement.
- ideas or intellectual data in any form presented as one's own without acknowledging the source(s).
- making significant use of unattributed digital images such as graphs, tables, photographs, etc. taken from test books, articles, films, plays, handouts, internet, or any other source, whether published or unpublished.
- submission of a piece of work which has previously been assessed for a different award or module or at a different institution as if it were new work.
- use of any material without prior permission of copyright from appropriate authority or owner

of the materials used".

Table of Contents:

Royal University of Bhutan

འབྲུག་རྒྱལ་ཁབ་ཀྱི་གཙུག་ལག་སློབ་གྲྭ།

College of Science and Technology

College of Science and Technology
Rinchending: Bhutan

Engagement Contacts

Contacts		
Primary Contact	Title	Email
Sonam Tenzin	Undergraduate	02230300.cst@rub.edu.bt

Executive summary

As our assignment, Mr. Kamal Acharya deployed three websites on the GCBS WiFi network for the students to perform a security test on the host 10.3.21.141 with ports 8000,8008 and 8090 to dig out all the possible security weakness and give recommendation about each and every weakness. The evidences for exploiting the websites are uploaded in the github and the link to the github repository is provided below:

https://github.com/SonamTenzin1/SWS101CAP2.git

Approach

I performed testing under a “grey box” approach since I was given crucial information like the database being used was MongoDB and mean stack js and that there were 3 vulnerable ports from 13th May, 2024 to 30th May, 2024. The testing of vulnerabilities was done from a non-evasive standpoint. Testing was performed remotely via a host that was provisioned specifically for this assessment. The vulnerabilities that were found are documented in detail and they were manually investigated by myself. I was able to gain root access in one of the ports that was vulnerable.

Scope
The scope of this this testing was to get to root access in all the available ports.

In-Scope assets

Host/IP address	Description
10.3.21.141:8008	Web Application: Gruyere
10.3.21.141:8000	Web application: Pixi
10.3.21.141:8090	Web application:

Table 1: scope details

Assessment Overview and Recommendations

The evaluation revealed significant weaknesses in the security of the web application demanding for an immediate action for resolution. Concerning the site hosted at 10.3.21.141:8008, vulnerabilities like elevation of privilege, cookie manipulation, and AJAX vulnerabilities were identified. Remediation strategies should concentrate on strengthening authentication and authorization process, fortifying cookie management with secure and HTTP only flags and deploying measures like rate limiting and request validation to thwart AJAX-based Denial of Service (DoS) attacks. Continuous monitoring of server logs and network traffic is also advised to swiftly detect and respond to potential security breaches. In summary, addressing these vulnerabilities is imperative for bolstering the security stance of the web applications and safeguarding sensitive data against unauthorized access and misuse. I suggest prioritizing remediation based on vulnerability and severity and implementing robust security protocols to mitigate future risks.

Website Penetration Test Assessment Summary

URL 1: 10.3.21.141:8008

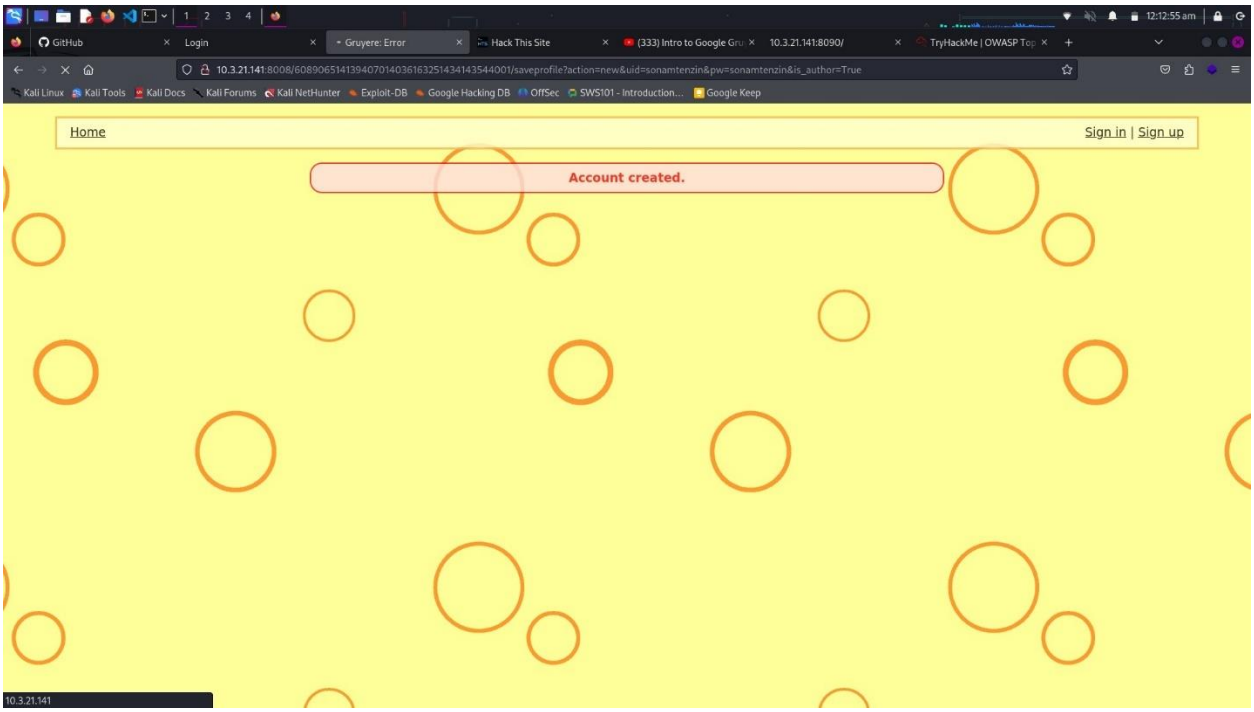
Vulnerabilities:

1.Privilege escalation through URL

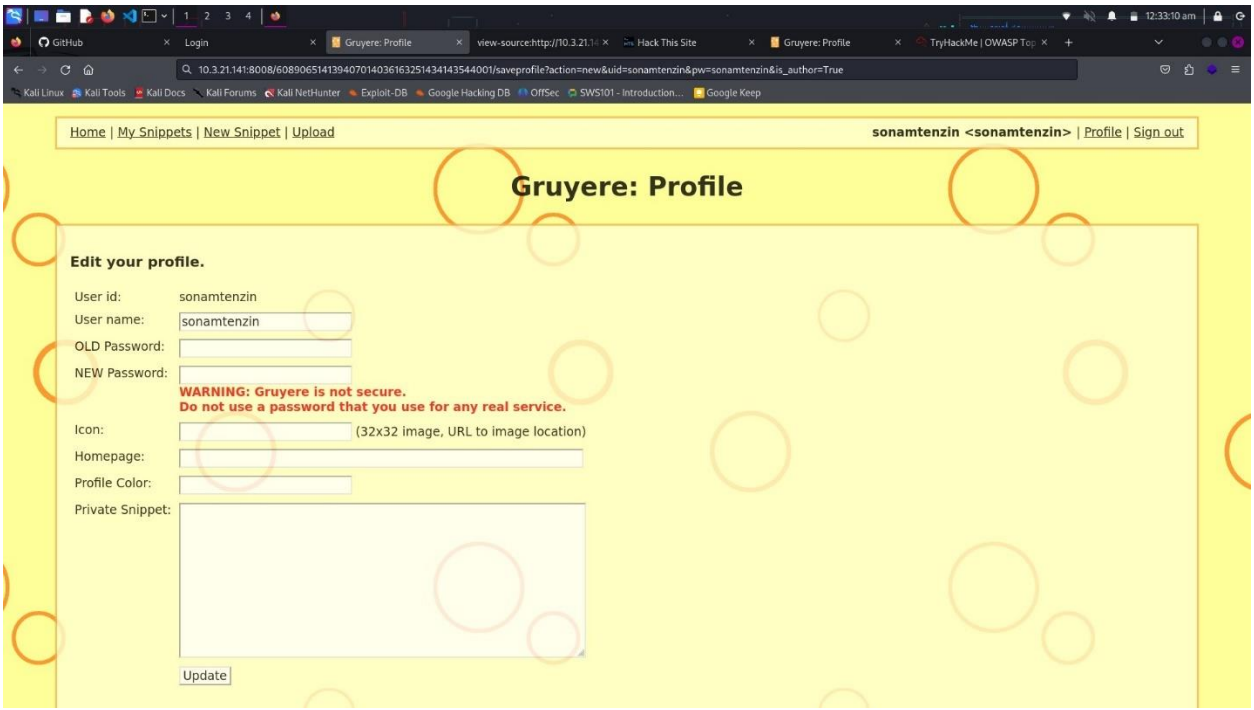
Privilege escalation is a process of gaining more control of the system than the access they are granted. The consequences of this action risk the security and integrity of data and data theft. To prevent this from happening, it is important to imply strong security.

Steps to reproduce

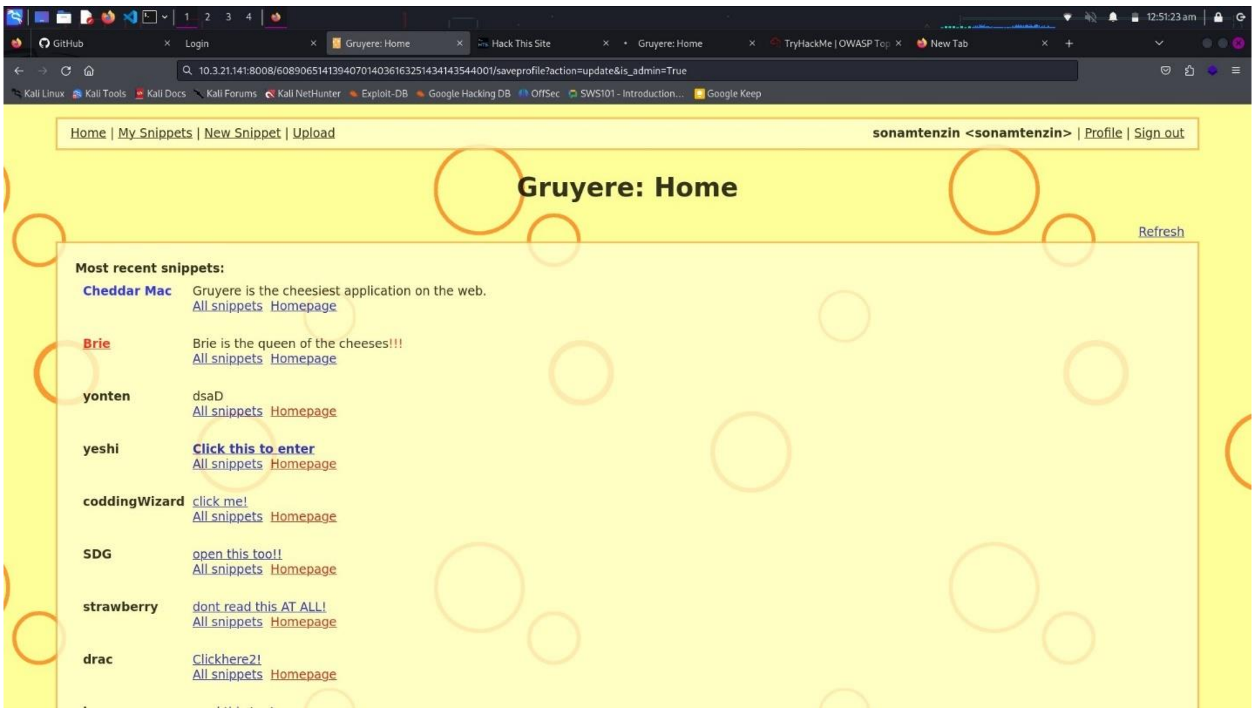
I started by making an account on the gruyere website.



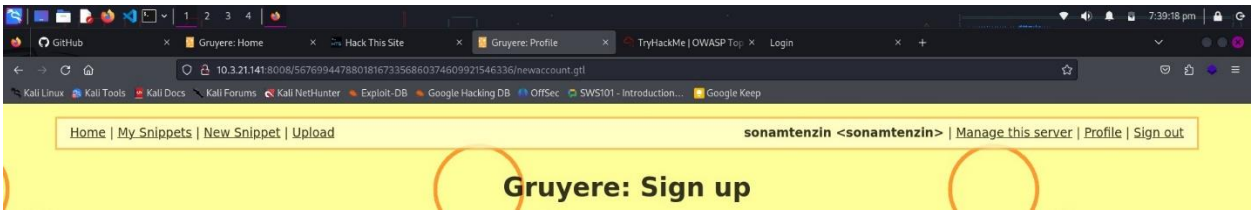
As of now, I don’t have the admin privilege granted to me.



However, if I use the query, saveprofile?action=update&is_admin=true in the URL, I can get the admin privilege since the website is constantly checking if the "is_admin" is true.



As you can see, after the running the query, I am granted the admin privilege and manage the server.



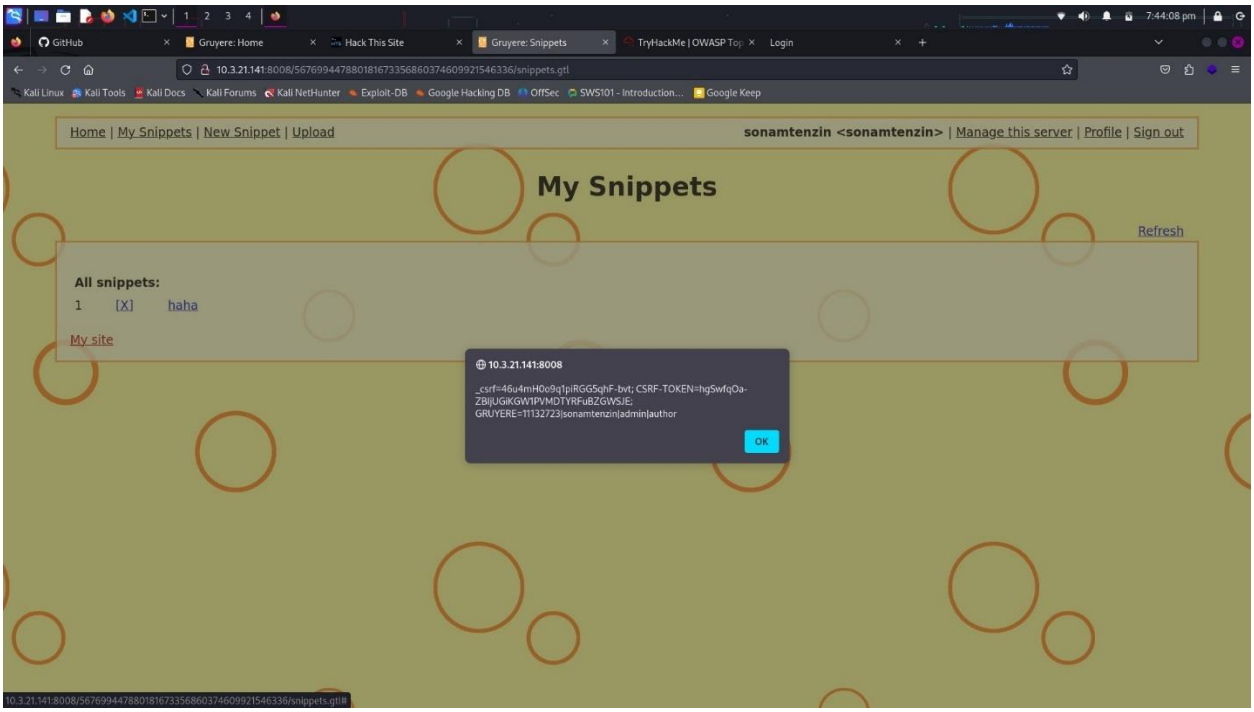
2. Cross-Site Scripting (XSS)

Cross-site scripting occurs when an attacker injects malicious script into web pages viewed by other users. These scripts are typically in Javascript and can execute various such as stealing session cookies and redirecting users to malicious sites or modifying page content. However, in this web application, I used HTML.

The consequences of XSS are severe. Attackers can exploit XSS to steal sensitive information, such as login credentials or personal data, from users who unknowingly interact with compromised pages. They can also perform actions on behalf of the user, such as making unauthorized transactions or manipulating account settings. Furthermore, XSS can be used to deface websites, distribute malware or launch attacks against other users through phishing or social engineering tactics.

Step to reproduce

Firstly, I uploaded a new snippet with the content “<a onmouseover’alert(document.cookie)’ href=”#”>haha and uploaded it. What this snippet will do is that it will show cookies when hover I over “haha”.



3. Cookie Manipulation

Cookie manipulation is a security threat where an attacker exploits vulnerabilities in web applications to gain unauthorized access to modify their content. Cookies are small pieces of data stored on a user's browser by websites they visit, often used for session management, authentication and tracking user preferences.

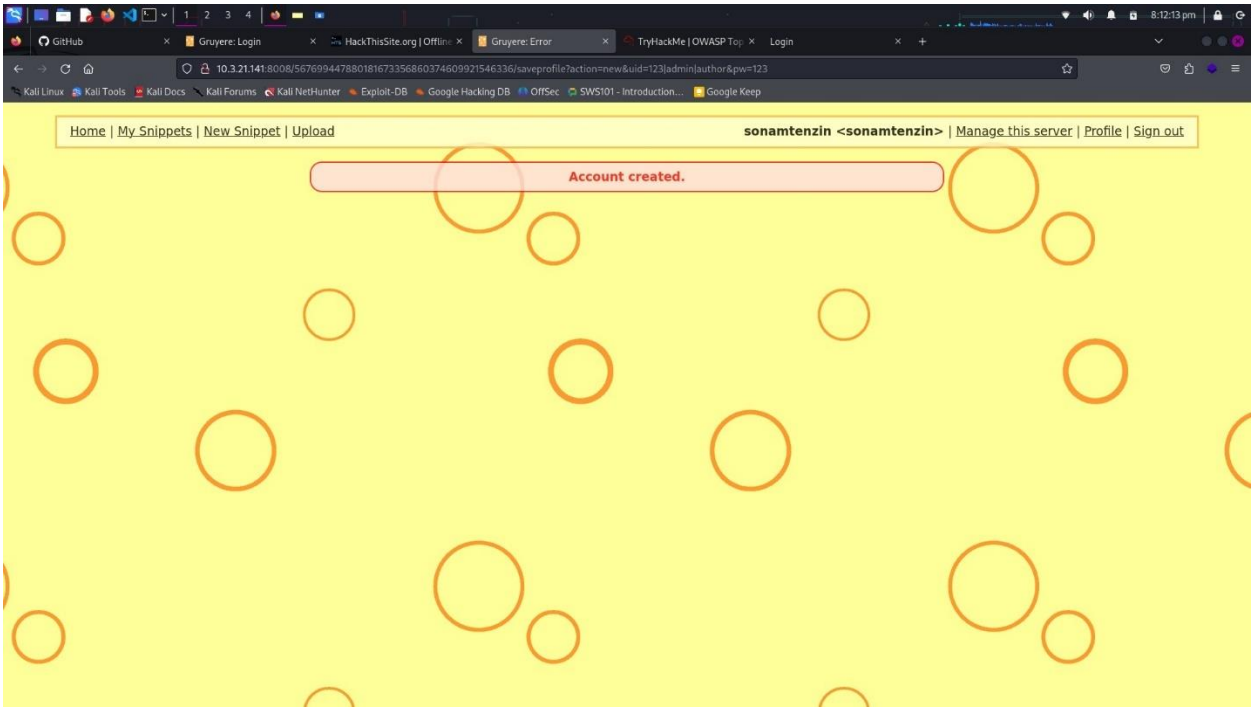
Through cookie manipulation, attackers can alter the content of cookies to impersonate users, bypass authentication mechanisms or escalate their privileges within the application. For example, they may modify a session cookie to gain access to another user's account or inject malicious data into cookies to execute further attacks.

The consequences of cookie manipulation are severe, leading to unauthorized access to the sensitive information, unauthorized transactions or compromise of user accounts. To mitigate this threat, web developers should implement secure coding practices such as encrypting sensitive information stored in cookies secure flags using HTTPOnly and Secure to prevent access by malicious scripts and validating and sanitizing cookie data on the server side. Additionally, regularly security audits and monitoring of cookies can help detect and respond to potential manipulation attempts promptly.

Gruyere remembers the cookie in the format [cookie]|username|admin|author in the URL section. In this case it is easy to locate the cookie and it is easy to manipulate it.

Steps to reproduce

Firstly, since the account that I am in a is an admin account, I used the cookie of that account to create another admin account. The username in of that account is 123 with password 123.

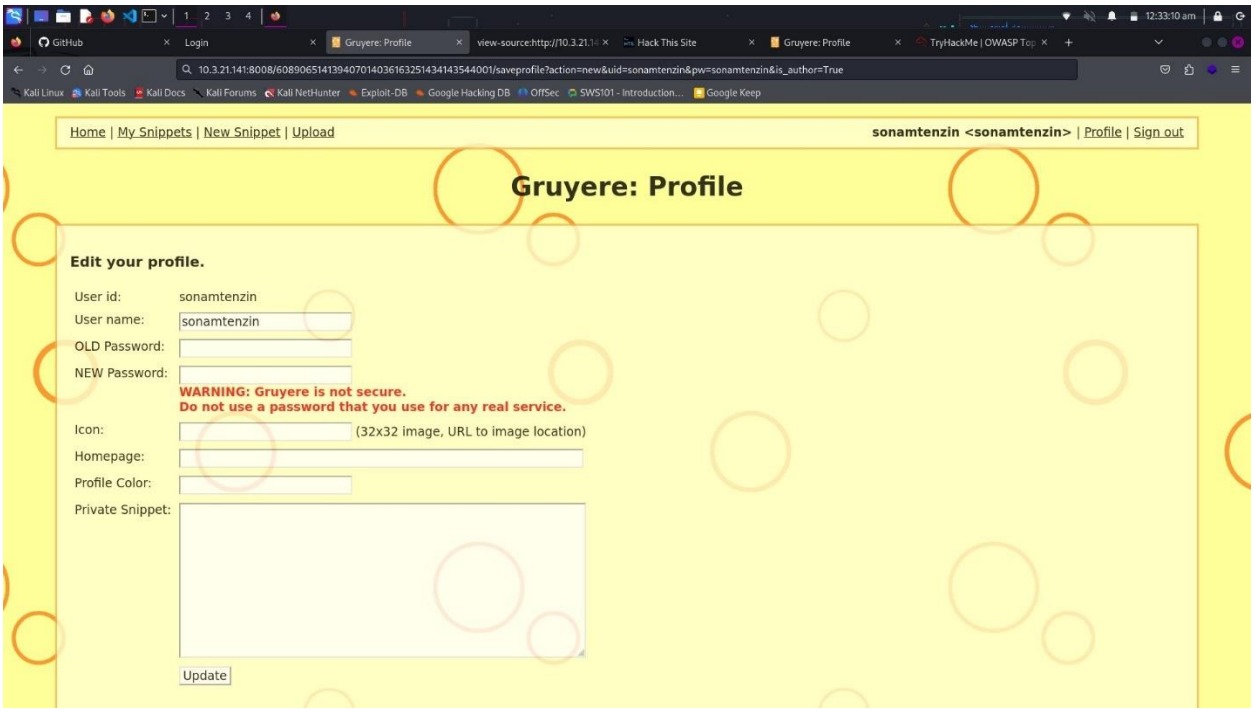


4. URL Parameter leakage

This vulnerability occurs when sensitive information such as usernames and passwords are included directly in the URL typically as query parameters. This poses a significant risk because the URLs are often logged in various places such as browser history, server logs and monitoring tools. If an attacker gains access to these logs, they can easily extract the sensitive information contained in the URL, compromising user accounts and potentially leading to unauthorized access.

Moreover, if the website doesn't use encryption, the credentials are transmitted in plaintext, making them vulnerable to interception by attackers monitoring network traffic.

To mitigate this vulnerability, it's crucial to avoid sensitive information especially credentials directly in the URL. Instead, use secure methods for authentication such as forms submitted vias HTTPS requests or authentication tokens managed securely on the server side.



Summary of Findings

During the course of testing, I discovered that 17 ports were open in total. However, I could manage to gain root access through 4 ports only. The table below shows the severity of the ports:

Finding severity			
High	Medium	Low	Total
4	0	0	4

Table 2: severity summary

Below is the severity of all the ports through which I had gained root access:

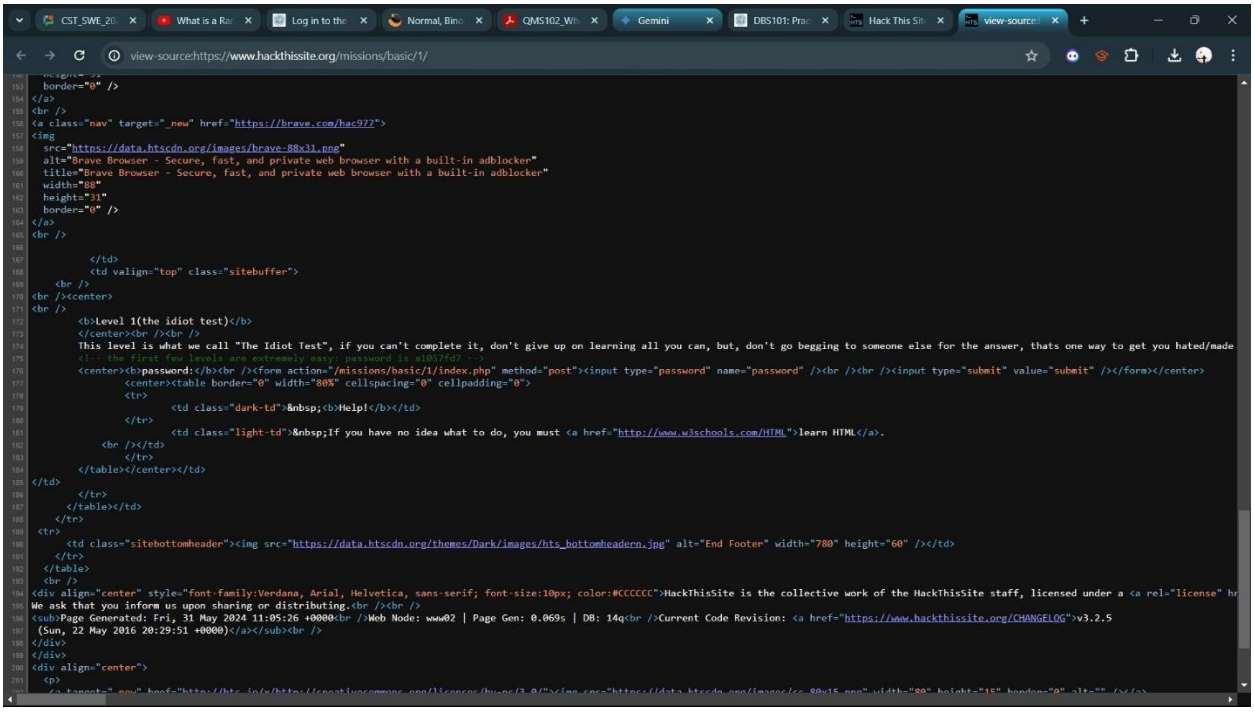
Sl. No.	Severity	Vulnerability
1	HIGH	Privilege Escalation
2	HIGH	Cookie Manipulation
3	HIGH	XSS
4	HIGH	URL parameter leakage

Table 3: finding list

2. hackthissite.org

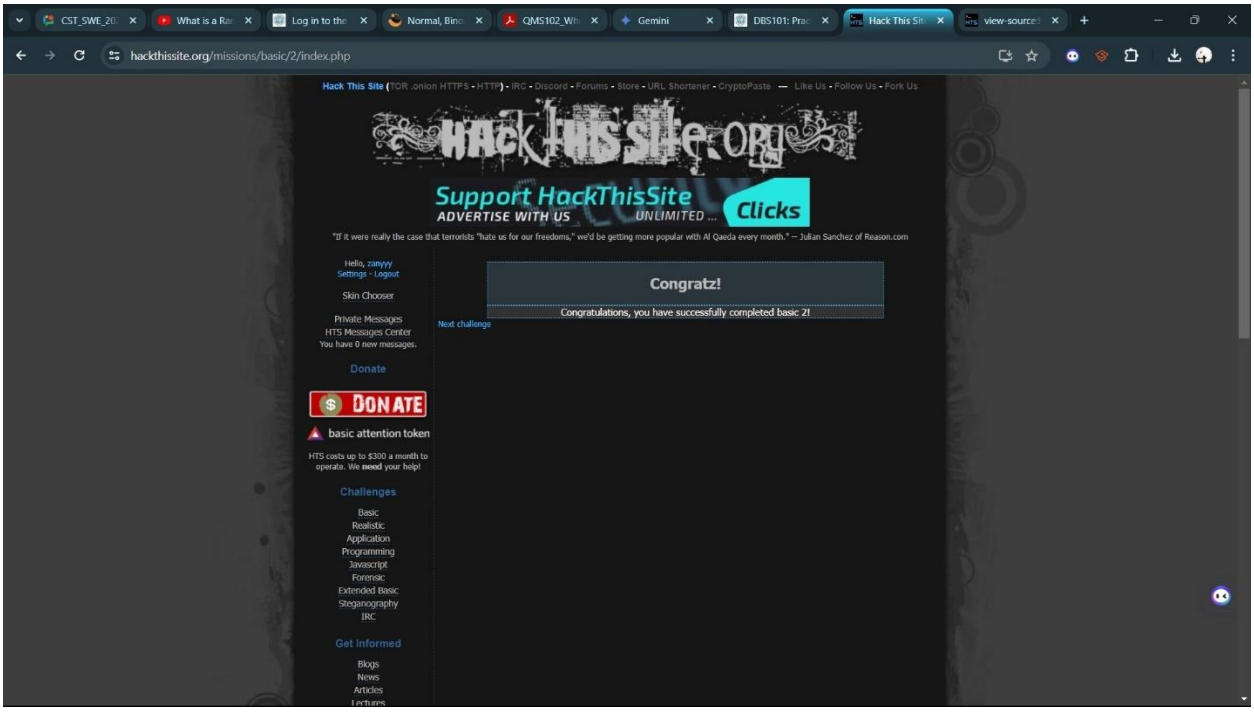
Level 1

In this level, the password is already given in the html comment.



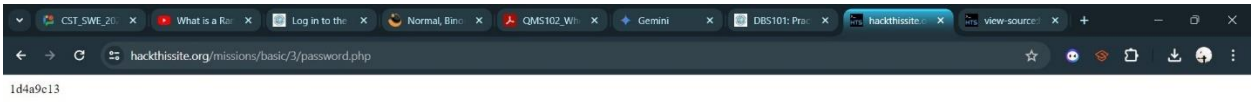
Level 2

Since Sam has forgotten to upload the password file, I directly got access without having to write a password.



Level 3

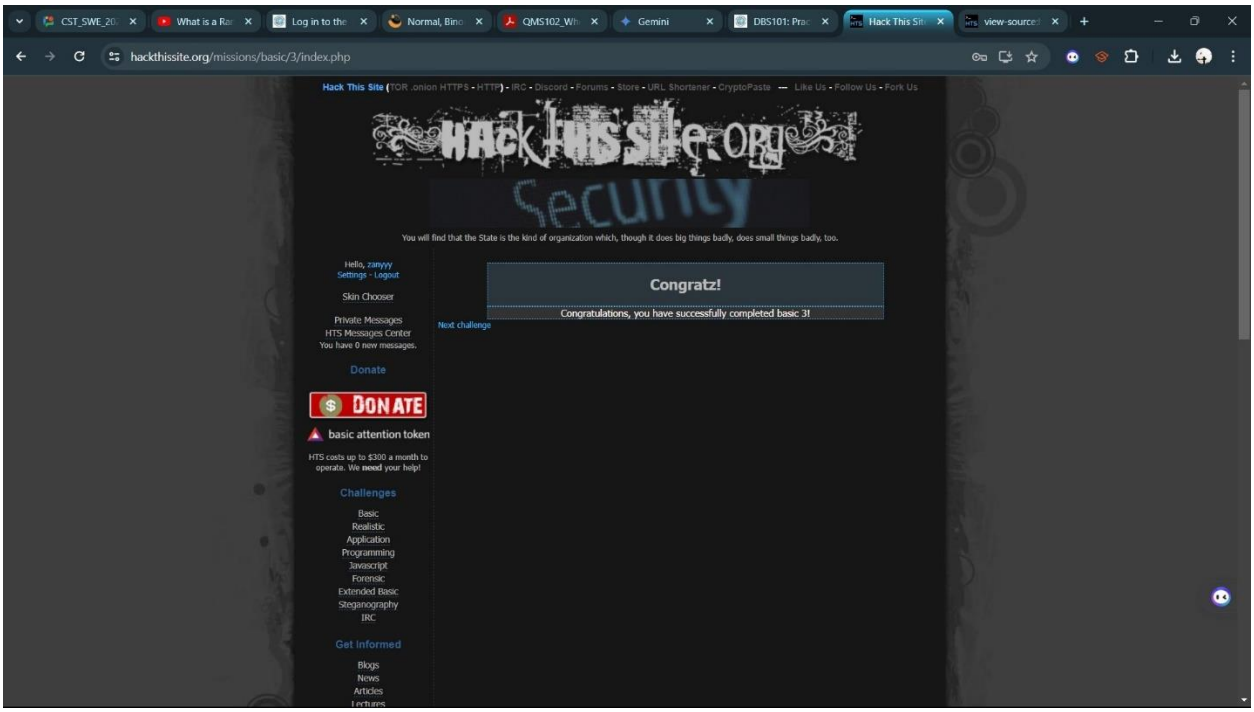
In this level, if we take a look a at the html source, we can find that there's a hidden directory called "index.php". After going to that directory we get the password for level 3.



Royal University of Bhutan

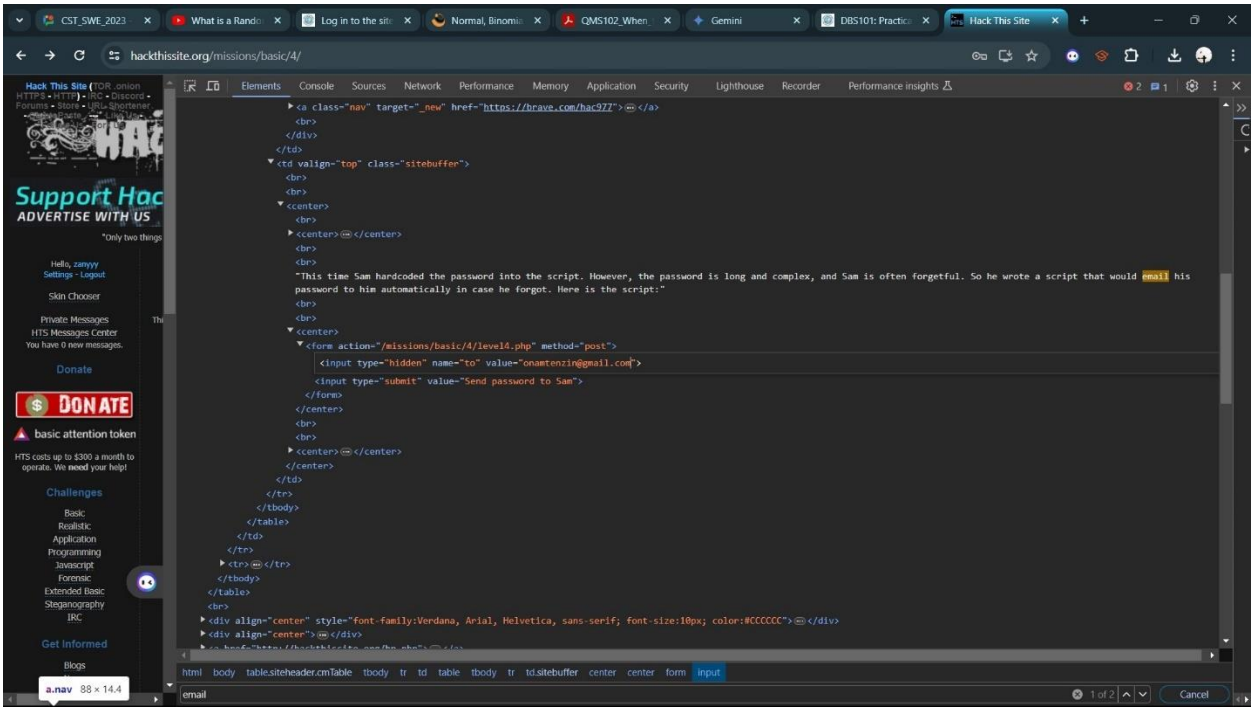
འབྲུག་རྒྱལ་འཛིན་གཞུག་ལག་སློབ་སྡེ།

College of Science and Technology
Rinchending: Bhutan



Level 4

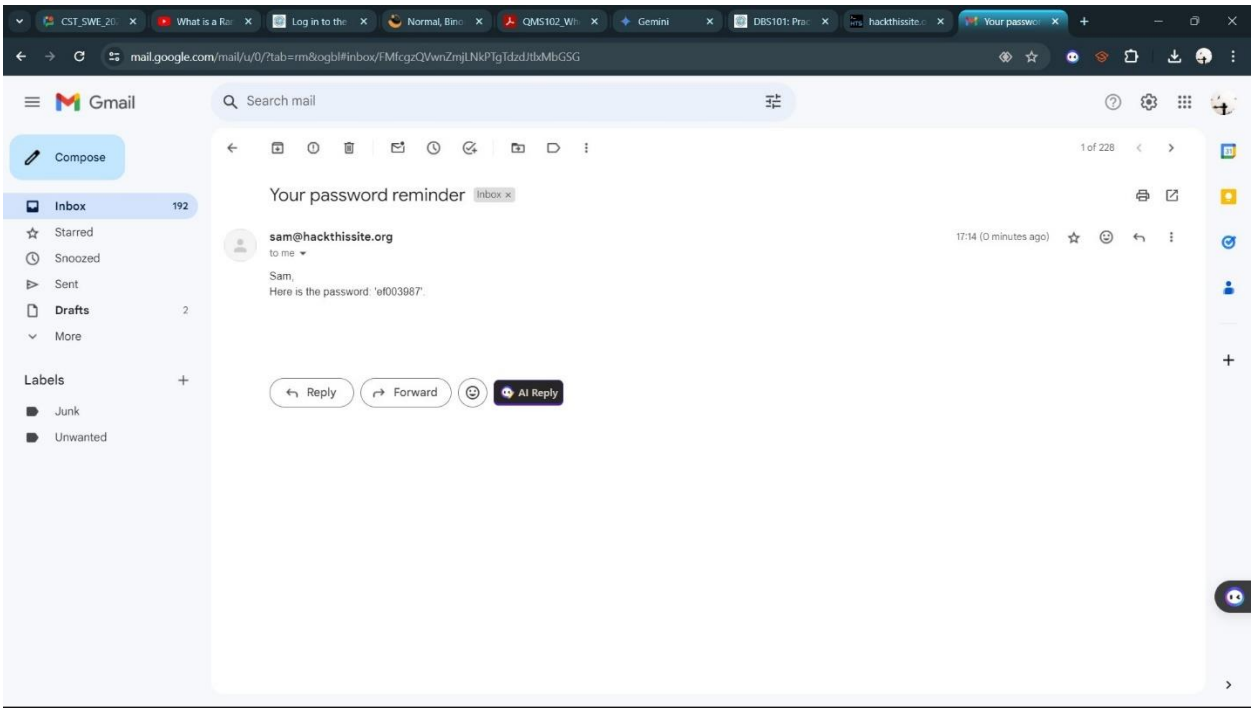
This level teaches us how to manipulate client-side scripts. In this level I changed the value field with my personal email account with Sam's email account.



Royal University of Bhutan

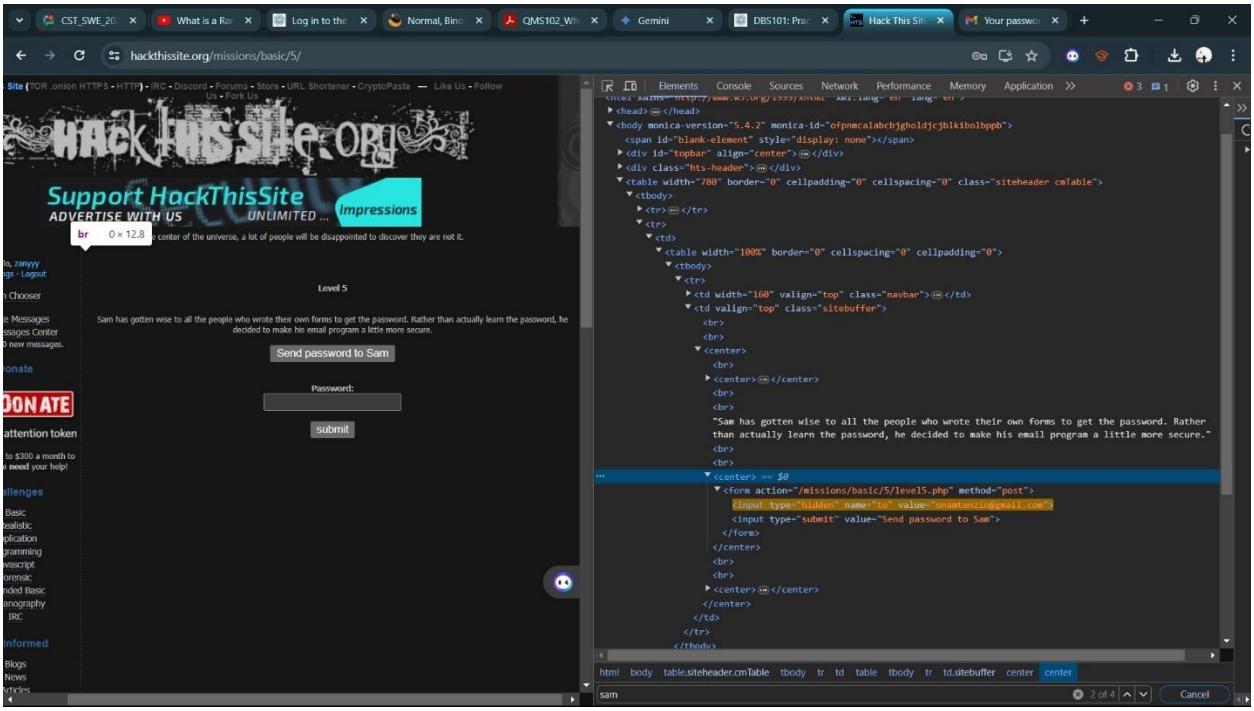
འབྲུག་རྒྱལ་འཛིན་གཞུང་ལག་སློབ་གྲྭ།

College of Science and Technology
Rinchending: Bhutan



Level 5

In this level, the way to get through this level is the same except that this level requires DOM manipulation. I modified the email address again with my own email account.

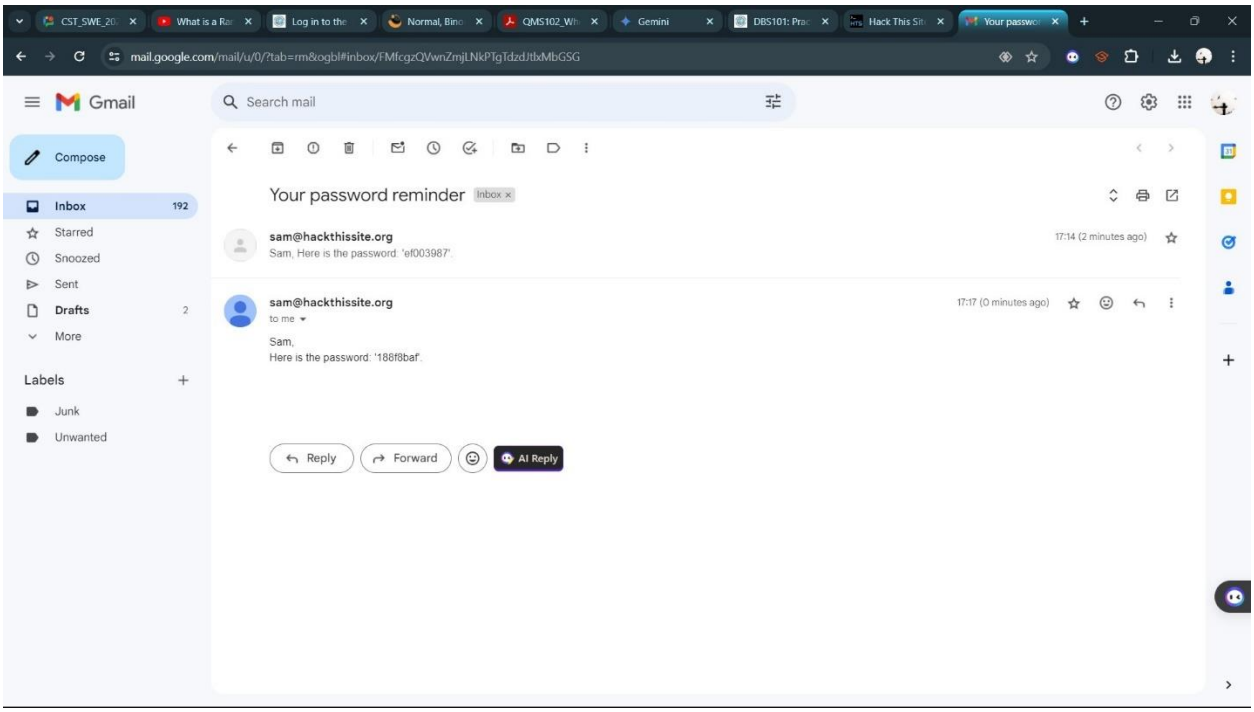


Royal University of Bhutan

འབྲུག་རྒྱལ་འཛིན་གཞུང་ལག་སློབ་གྲྭ།

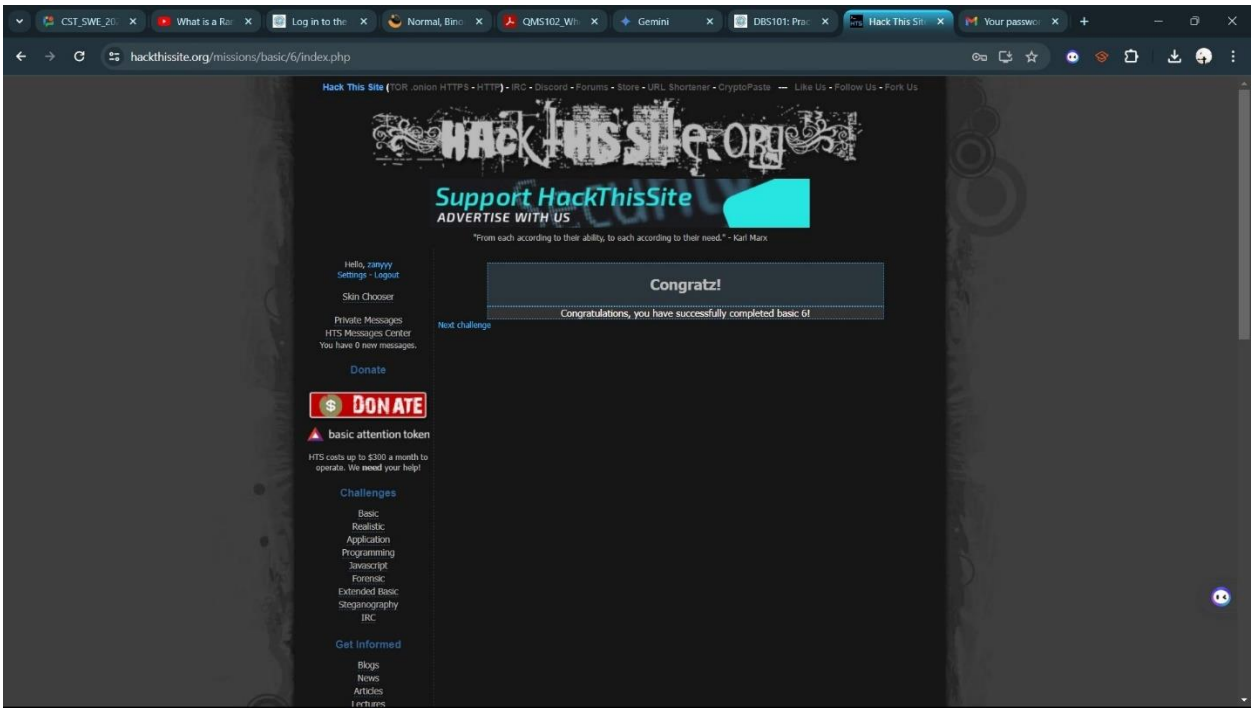
College of Science and Technology

College of Science and Technology
Rinchending: Bhutan



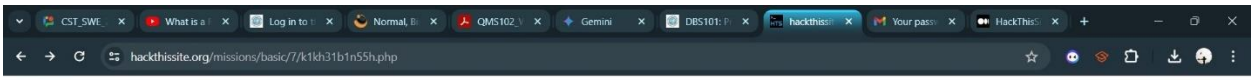
Level 6

In this level we're provided with an encrypted password which we have to decrypt. The encrypted password that I got was a1ffg578. When I decrypted the password, the password turned out to be a0dcc011.



Level 7

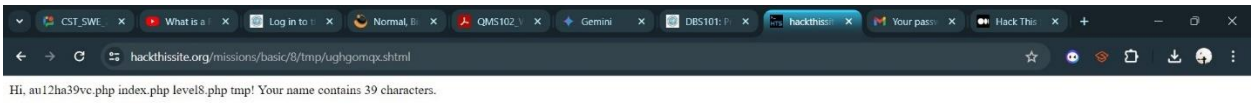
In this level we have to use the “ls” command from the UNIX OS which lists everything in a directory. After inputting a year with ls (ex. 2012l; ls) we are given a calendar along with the directory. After directing to k1kh31b1n55h.php, we get the password to pass the level.

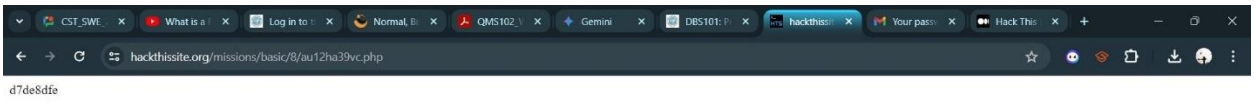


101cabe5

Level 8

In this level we're expected to use to SQl injection. We have to submit the command <!--#exec cmd="ls .." --> and then we are redirected to a page which tells us the next directory au12ha39vc.php and while we visit this directory, we get the password.

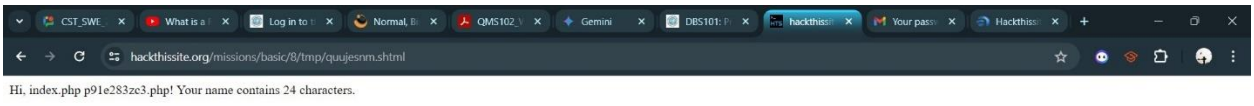


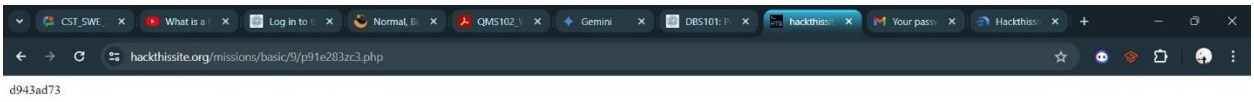


d7de8dfe

Level 9

This level is similar to the level 8 however to complete this level, we have to revisit the previous level and use a similar comment which is <!--#exec cmd="ls../../9" --> and we are given two directories. Visiting the second directory, we get the password for the level.

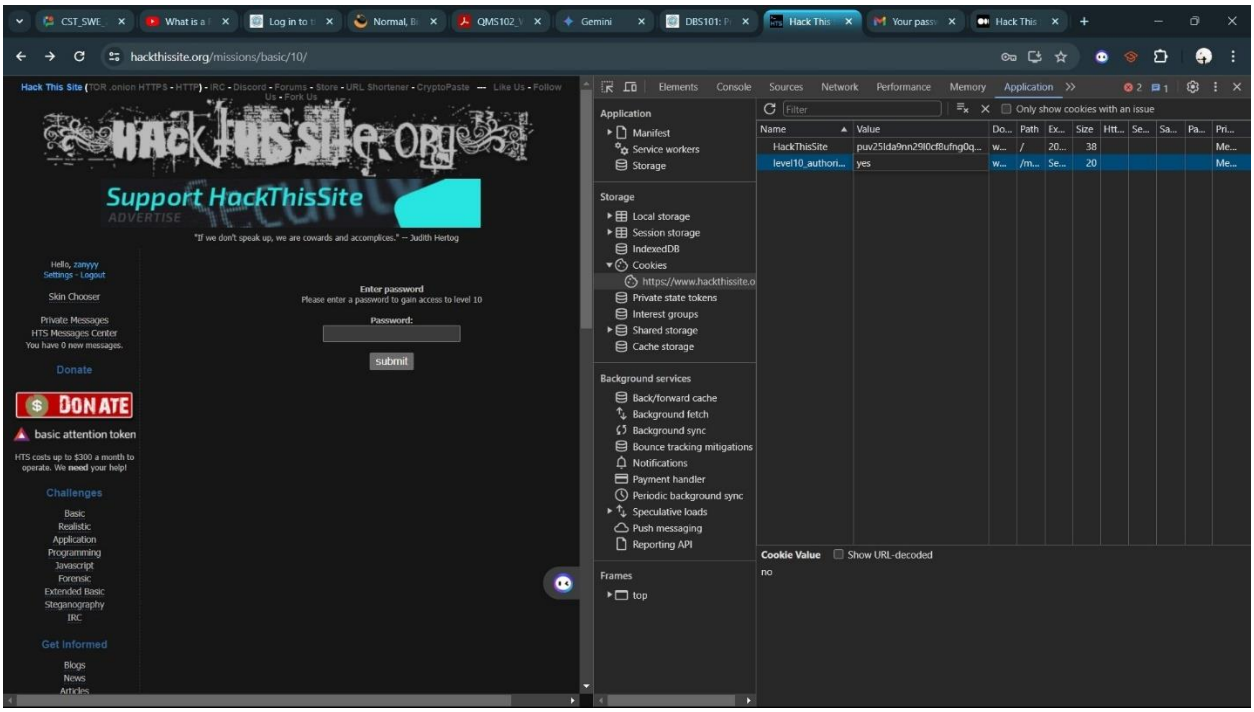




d943ad73

Level 10

In this level we make use of cookies. When we take a look at the cookies, we notice that the value for level10_authorization is no but if we change the value to yes, we get the authorization for level 10. After changing the value, even without a password we are able complete the level.

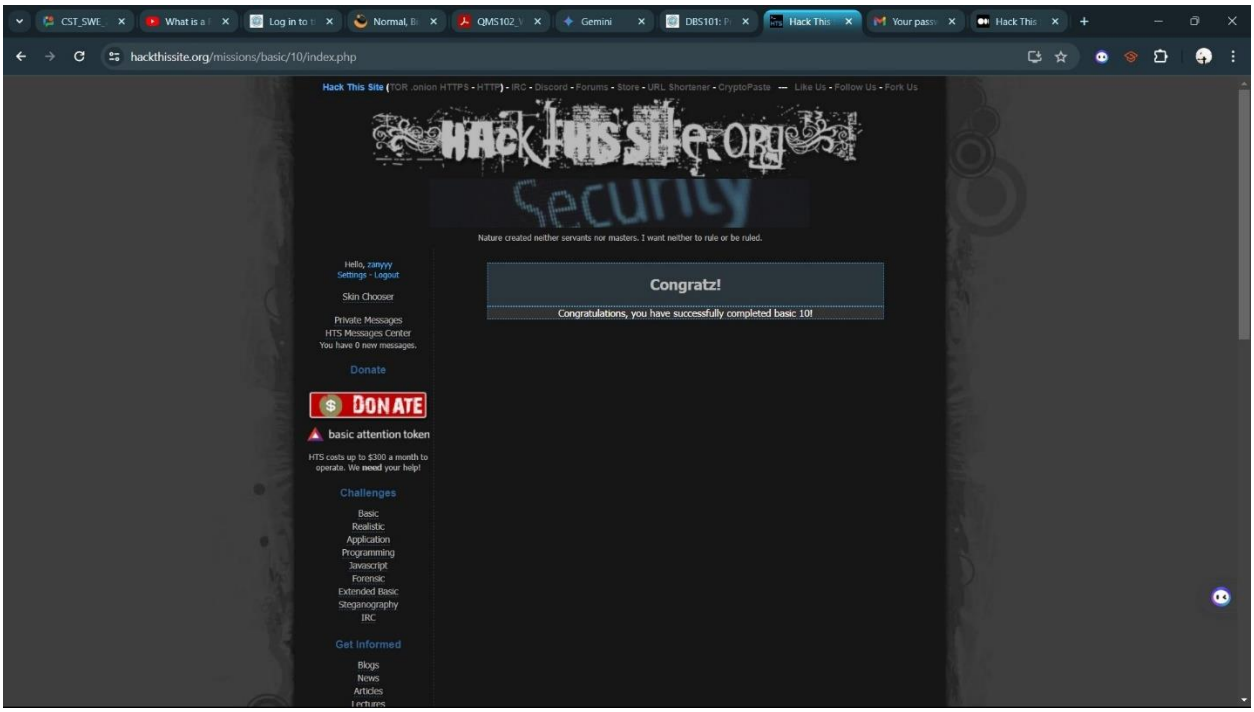


Royal University of Bhutan

འབྲུག་རྒྱལ་ཁབ་ཀྱི་གཙུག་ལག་སློབ་གྲྭ།

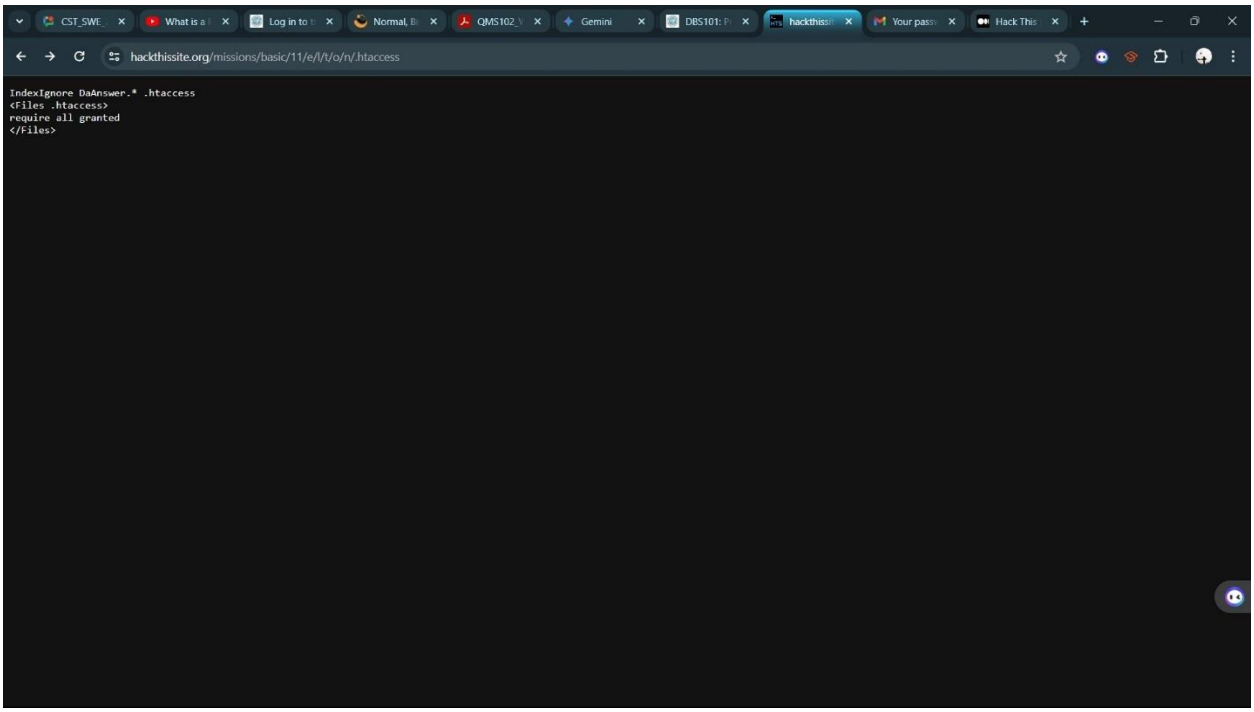
College of Science and Technology

College of Science and Technology
Rinchending: Bhutan

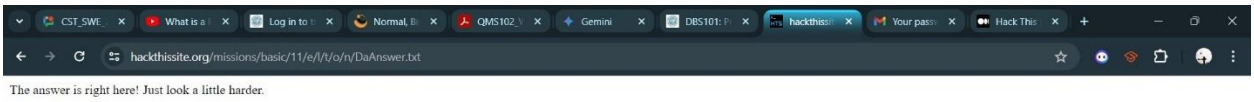


Level 11

We have a login page and there is no hint given. When scan for directories, we have a /e directory. While continuing to scan, we have e/l/t/o/n/. Since Sam is using Apache in the website, we can't rule out the possibility that he used .htacess. In this page, DaAnswer seems interesting so while trying to query it, it redirected a page.



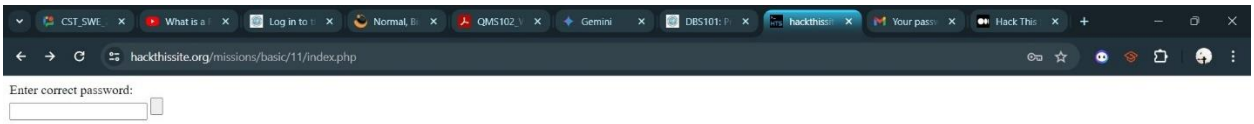
It seems that the answer for this level is “right here”



Royal University of Bhutan

འབྲུག་རྒྱལ་ཁབ་ཀྱི་གཙུག་ལག་སློབ་གྲྭ།

College of Science and Technology
Rinchending: Bhutan



Conclusion

As I reflect on completing the basic challenges on HackThisSite, I feel a profound sense of accomplishment and growth. These challenges have not only honed my technical skills but have also taught me invaluable lessons about cybersecurity and problem-solving. From the initial stages learning basic concepts to the more intricate puzzles requiring creative thinking, each challenge has pushed me to think outside the box and approach problems from different angles.