Vulnerable Web Application - OWASP Top 10 Vulnerabilities

Group 2

Puchakayala Dheeraj Reddy B180902CS

Goutham P B180330CS

Yacha Venkata Rakesh B180427CS

Name: Puchakayala Dheeraj Reddy
Roll Number: B180902CS
Email: dheeraj_b180902me@nitc.ac.in
Course: Computer Security (CS4035D)

Contribution to the Project

1. Broken Authentication Attack and Protection against it

2. Broken Access Control Attack and Protection against it

A. Abstract

The project implements various web vulnerabilities provided by OWASP as *project top ten* that can lead to very severe consequences compromising *Confidentiality*, *Integrity* and *Availability* and the ways to prevent each of these attacks. This is not only to know about attacks but also to enable developers to write safe code by knowing most of the possible vulnerabilities in their code since a single line of vulnerable code can compromise the entire system.

B. Introduction:

This report provides an overview of the vulnerable web application being implemented to demonstrate the OWASP top 10 vulnerabilities.
OWASP(Open Web Application Security Project) is a nonprofit foundation that works to improve the security of software. Through community-led open-source software projects, hundreds of local chapters worldwide, tens of thousands of members, and leading educational and training conferences, the OWASP Foundation is the source for developers and technologists to secure the web.

- Tools and Resources
- Community and Networking
- Education & Training

OWASP top 10 vulnerabilities

Every three to four years, OWASP revises and publishes its list of the top 10 web application vulnerabilities. The list includes not only the OWASP top ten threats but also the potential impact of each vulnerability and how to avoid them. The comprehensive list is compiled from a variety of expert sources such as security consultants, security vendors, and security teams from companies and organizations of all sizes. It is recognized as an essential guide to web application security best practices.

The four main criteria used to compile the OWASP top ten vulnerabilities are:

- Exploitability
- Prevalence
- Detectability
- Business impact.

The latest list of the vulnerabilities from OWASP include

- Injection
- Broken Authentication
- Sensitive Data Exposure
- XML External Entities (XXE)
- Broken Access Control.
- Security Misconfiguration.
- Cross-Site Scripting (XSS)
- Insecure Deserialization
- Using Components with Known Vulnerabilities
- Insufficient Logging & Monitoring

The main motto of the project is to develop a web application to demonstrate the above mentioned vulnerabilities and to try and learn about the ways to tackle them. The web application being developed is vulnerable to many kinds of attacks each pertaining to one or more categories mentioned above.

C. Literature Survey:

1. Broken Authentication:

Broken authentication is typically caused by poorly implemented authentication and session management functions. Broken authentication attacks aim to take over one or more accounts giving the attacker the same privileges as the attacked user. Authentication is “broken” when attackers are able to compromise passwords, keys or session tokens, user account information, and other details to assume user identities.

Due to poor design and implementation of identity and access controls, the prevalence of broken authentication is widespread. Common risk factors include:

- Predictable login credentials
- User authentication credentials that are not protected when stored
- Session IDs exposed in the URL (e.g., URL rewriting)
- Session IDs vulnerable to session fixation attacks
- Session value that does not time out or get invalidated after logout

- Session IDs that are not rotated after successful login
- Passwords, session IDs, and other credentials sent over unencrypted connections

2. Sensitive Data Exposure

Sensitive data is any information that is meant to be protected from unauthorized access. Sensitive data can include anything from personally identifiable information (PII), such as Social Security numbers, to banking information, to login credentials. When this data is accessed by an attacker as a result of a data breach, users are at risk for sensitive data exposure. Data breaches that result in the exposure of sensitive credentials can come with costs in the millions of dollars, destroying a company's reputation along with it. During the 21st century, the use of mobile devices has increased internet usage dramatically. As a result, banks, hospitals, retail, and many other industries have made it their mission to create a user-friendly and efficient online presence, one which applications have improved dramatically.

Attackers target applications with vulnerabilities that leave sensitive data unprotected. Today, data breaches are common and pose a bigger threat than ever as legacy application security falls far behind advanced attack techniques used to exploit application vulnerabilities.

Types of Data Exposed:

Data in Transit:

Data is often on the move, sending commands and requests across networks to other servers, applications, or users. Data in transit is highly vulnerable, especially when moving across unprotected channels or to the application programming interface (API) that allows applications to communicate with one another.

Data at Rest

Data at rest is housed in a system, be it a computer or network. It is thought to be less vulnerable without the threat of attacks in passing, but more valuable. Attackers use different vectors to get hold of housed data, often using malware like Trojan horses or computer worms. Both of these gain access into systems housing data through direct downloading from a malicious USB drive or by clicking malicious links that are sent via email or instant message. If data is housed in a server, attackers could get ahold of information stored in files outside of the normal authenticated areas of access.

Attacks that expose Sensitive data are:

- SQL Injection Attacks
- Network Compromise
- Broken Access Control Attacks
- Ransomware Attacks
- Phishing Attacks
- Insider Threat Attacks

3. Broken Access Control

Broken access control allows attackers to bypass authorization safeguards and perform tasks as if they were privileged users.

When working correctly, access control is the way a web application enforces policies that manage access to content and functions, granting authorization to some users and denying it to others. Application access policies can be "broken" when developers misconfigure functional-level access, resulting in flaws or gaps that deny access to legitimate users and let attackers assume the role of users or administrators outside of an application's intended permissions. Broken access control failures can lead to unauthorized information disclosure, modification or destruction of data, and the performance of a business function that deviates from its intended use. Common access control vulnerabilities include:

- Bypassing access control checks by modifying the URL, internal application state, or the HTML page, or simply using a custom API attack tool
- Allowing the primary key to be changed to another's users record, permitting viewing or editing someone else's account.
- Elevation of privilege. Acting as a user without being logged in, or acting as an admin when logged in as a user.
- Metadata manipulation, such as replaying or tampering with a JSON Web Token (JWT) access control token or a cookie or hidden field manipulated to elevate privileges, or abusing JWT invalidation
- CORS misconfiguration allows unauthorized API access.

- Force browsing to authenticated pages as an unauthenticated user or to privileged pages as a standard user. Accessing API with missing access controls for POST, PUT and DELETE.

D. System Environment for Implementation

Used Node JS for the backend to connect with the MySQL database. Front end includes HTML, CSS, JavaScript. View engine ejs is used to automatically generate the html pages from the data obtained from the database. Works with any operating system Linux, Windows, Mac.

E. Design of various modules of the system

1. Broken Authentication

a. Session ID or User ID exposed in URL

The above mentioned vulnerability is demonstrated using an authentication system, where the User Id, the key parameter for the authentication of a user for this website is exposed in the URL of the website. The userid is neither hashed nor is hidden in this case. When an attacker comes across the URL of the website, the user can easily change the userid to a random value, so that he can impersonate any other user of the website. This is a very serious situation where the attacker can impersonate any user registered in the system.

b. Passwords, session IDs, and other credentials sent over unencrypted connections

This way of broken authentication is demonstrated using another authentication system, where the user id is stored in a cookie without being hashed. This website being hosted on an insecure channel, provides easy access to the cookie. The attacker can easily capture the HTTP requests sent to the website requesting any of the webpages. The attacker can then refabricate the same request with a changed user id value to impersonate any other user of the website. This attack is demonstrated using Burp Suite.

Burp Suite is a HTTP sniffer and repeater. The HTTP requests sent from this website are captured. Then the user id value in the cookie is changed and the request is then repeated to get the access to the account of another user.

2. Broken Access Control

a. Accessing the Admin APIs using the user credentials

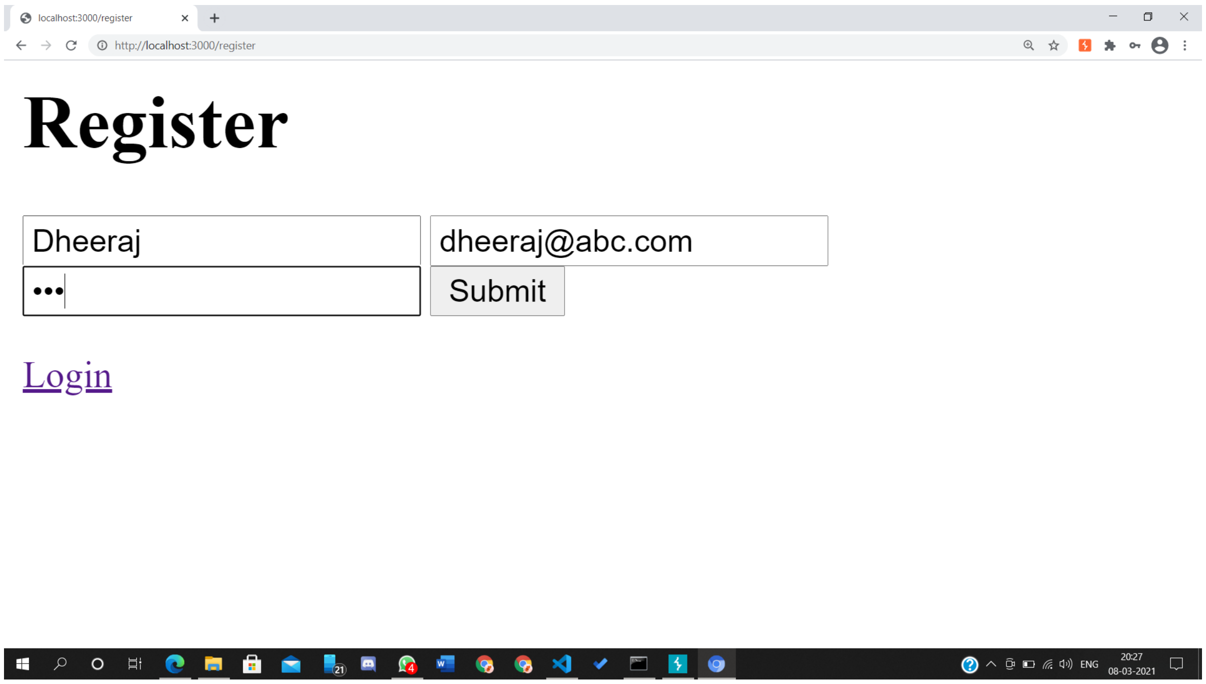
The above mentioned attack of using the user credentials to access the admin APIs is one of the types of broken access control attacks. Here the attacker is also a user of the website. But he uses his user level credentials to access the admin APIs. This happens because the web application does not differentiate between the user level authentication and the admin level authentication. The attacker is thereby able to elevate his privilege after logging into the website. The attacker can change the URL to the URL of a page with admin privilege to access it as the system does not check for a privilege in authentication of the user.

F. Implementation

1. Broken Authentication

a. Session ID or User ID exposed in URL

The user of the website can easily access the user id from the user id used for authentication process through the URL of the page. By simply changing the url parameter of the user id, the user can easily gain access over other user's accounts.


localhost:3000/register

http://localhost:3000/register

Register

Dheeraj

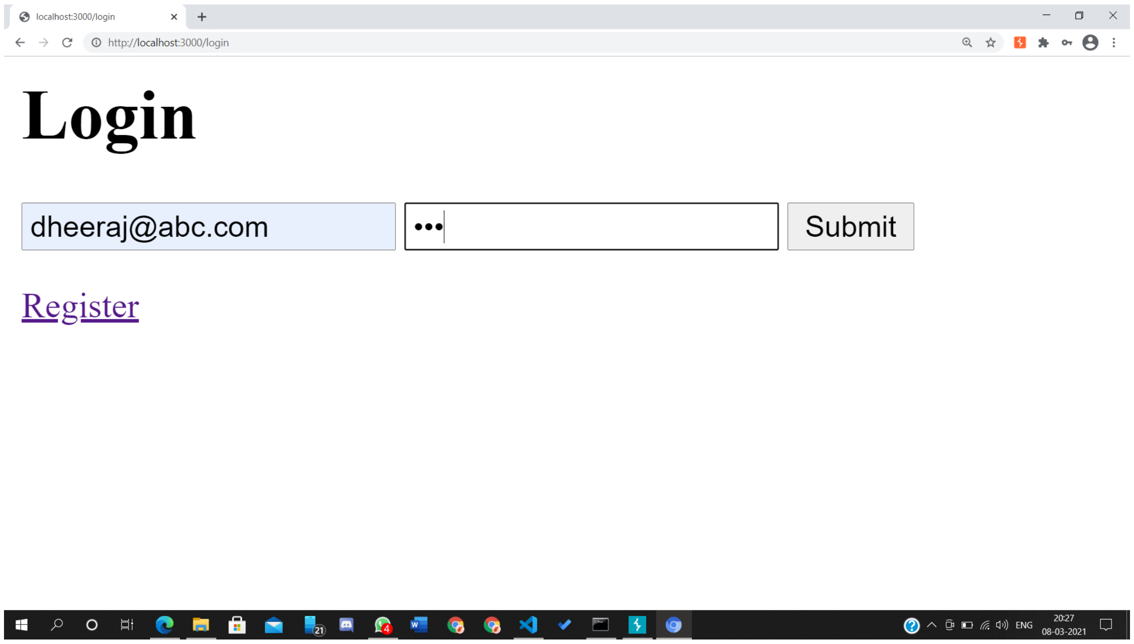
dheeraj@abc.com

•••

Submit

Login

Register a User


localhost:3000/login

http://localhost:3000/login

Login

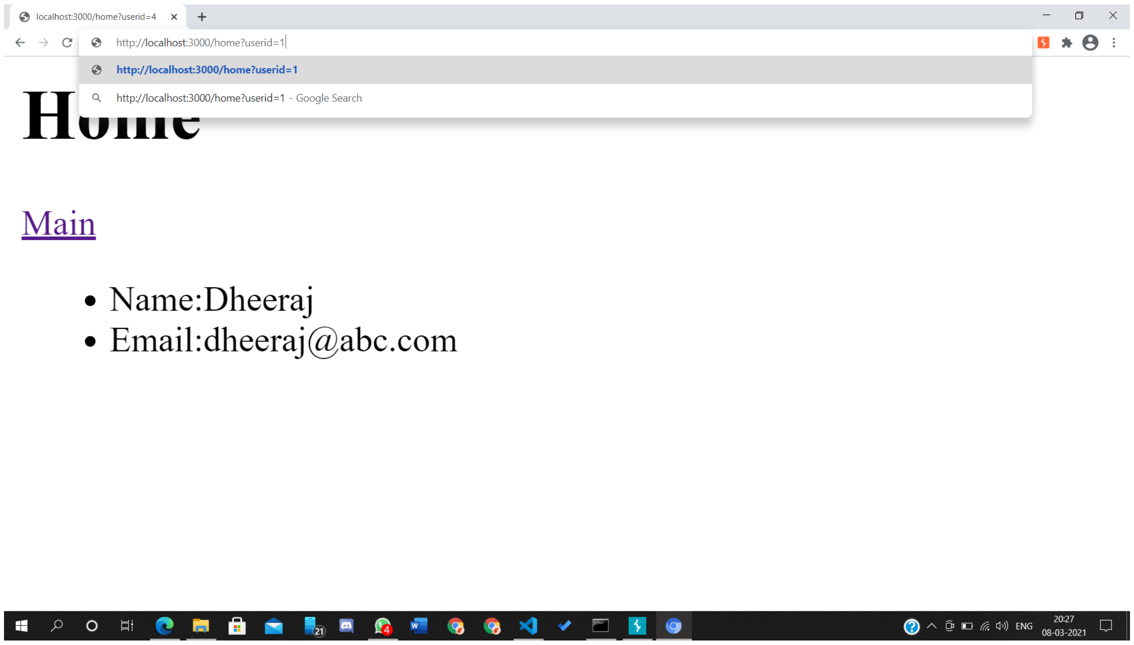
dheeraj@abc.com

•••

Submit

Register

Login as a User

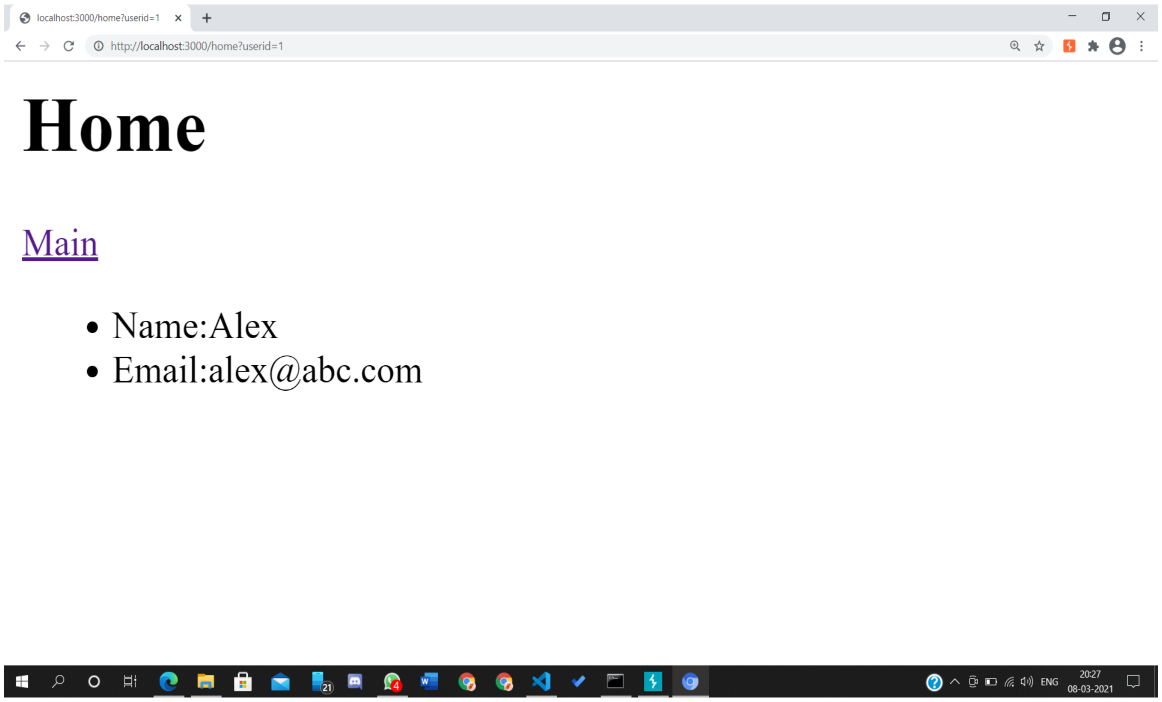

localhost:3000/home?userid=4
http://localhost:3000/home?userid=1
http://localhost:3000/home?userid=1
http://localhost:3000/home?userid=1 - Google Search

Home

Main

- Name:Dheeraj
- Email:dheeraj@abc.com

Changing the URL parameter


localhost:3000/home?userid=1
http://localhost:3000/home?userid=1

Home

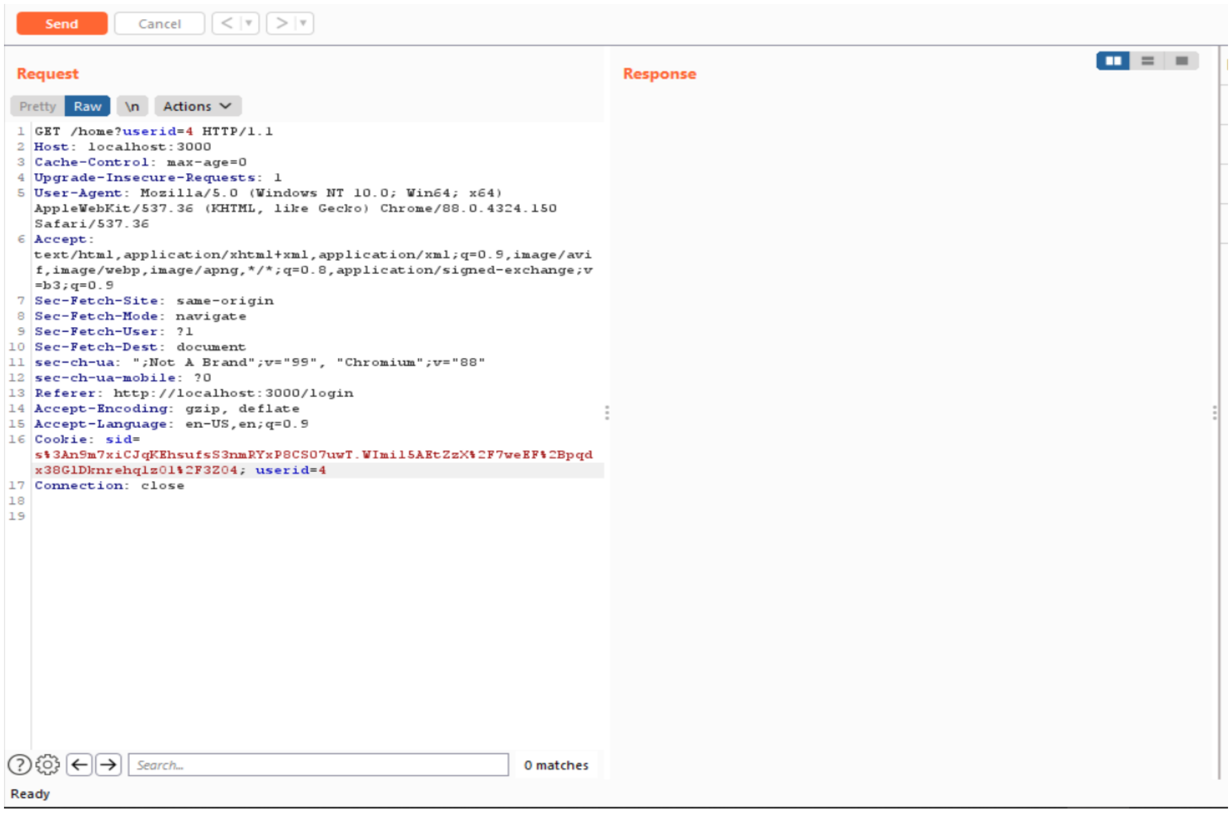
Main

- Name:Alex
- Email:alex@abc.com

Details of other User

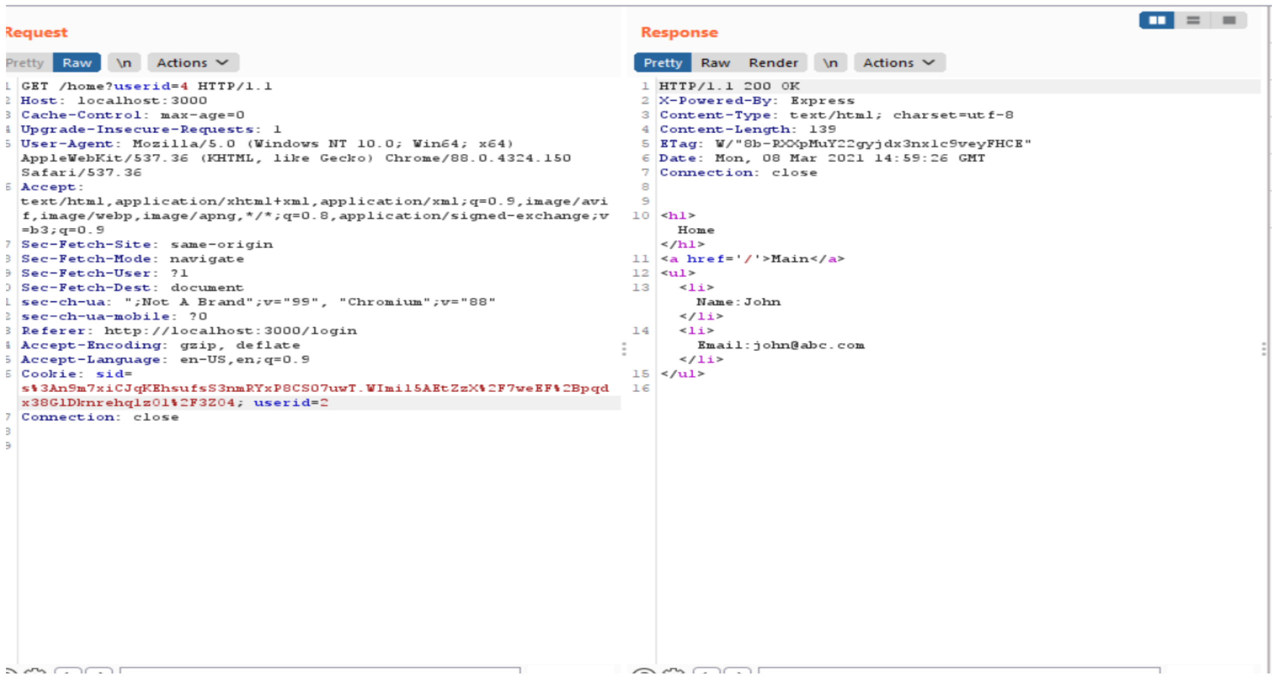
b. Passwords, session IDs, and other credentials sent over unencrypted connections

In this case, the user id used in the authentication process and for maintaining the session is passed over using an unencoded cookie which can easily be acced by the users by capturing the http requests using any http sniffer. After capturing the http request, the attacker can refabricate the packet using other user ids. This could lead to authentication of other users and access to their details



```
GET /home?userid=4 HTTP/1.1
Host: localhost:3000
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.150
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avi
f,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v
=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
sec-ch-ua: ";Not A Brand";v="99", "Chromium";v="88"
sec-ch-ua-mobile: ?0
Referer: http://localhost:3000/login
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: sid=
s%3An9m7xiCJqKEhsufsS3nmRYxP8CSO7uwT.WImil5AEtZzX%2F7weEF%2Bpqd
x38GlDknrehqlz01%2F3Z04; userid=4
Connection: close
```

Captured request during login



Request

```
GET /home?userid=4 HTTP/1.1
Host: localhost:3000
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.150
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avi
f,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v
=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
sec-ch-ua: ";Not A Brand";v="99", "Chromium";v="88"
sec-ch-ua-mobile: ?0
Referer: http://localhost:3000/login
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: sid=
s%3An9m7xiCJqKEhsufsS3nmRYxP8CS07uwT.WImil5AEtZzX%2F7weEF%2Bpqd
x38G1Dknrehqlz01%2F3Z04; userid=2
Connection: close
```

Response

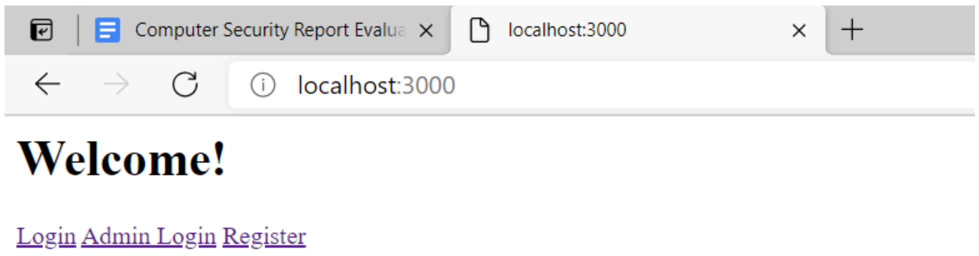
```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/html; charset=utf-8
Content-Length: 139
ETag: W/"8b-RXXpMuY22gyjdx3nx1c9veyFHCE"
Date: Mon, 08 Mar 2021 14:59:26 GMT
Connection: close


<h1>
  Home
</h1>
<a href='/'>Main</a>
<ul>
  <li>
    Name:John
  </li>
  <li>
    Email:john@abc.com
  </li>
</ul>
```

Changed cookie values, logs in as another user

2. Broken Access Control

a. Gaining access to admin APIs or pages

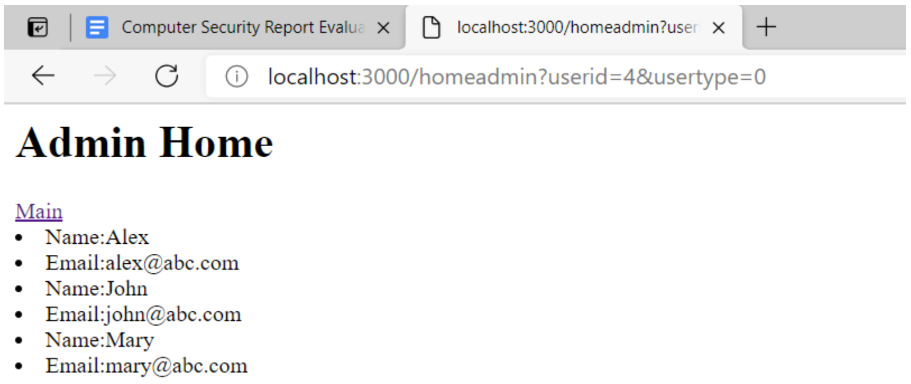
In this attack, the attacker can easily spoof himself to be the admin of the website (a user with higher privileges to the website). This is done by simply changing the route through the url of the page. The user can easily add admin to the name of the route and access the admin pages. By doing this he can escalate his privilege level and perform malicious activities which are unintended



Welcome!

Login Admin Login Register

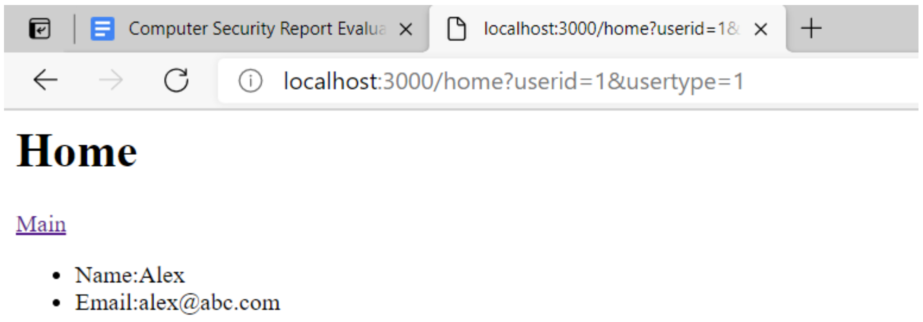
Index page for the users


Computer Security Report Evalua
localhost:3000/homeadmin?user
localhost:3000/homeadmin?userid=4&usertype=0

Admin Home

Main
- Name:Alex
- Email:alex@abc.com
- Name:John
- Email:john@abc.com
- Name:Mary
- Email:mary@abc.com

Administrator home page

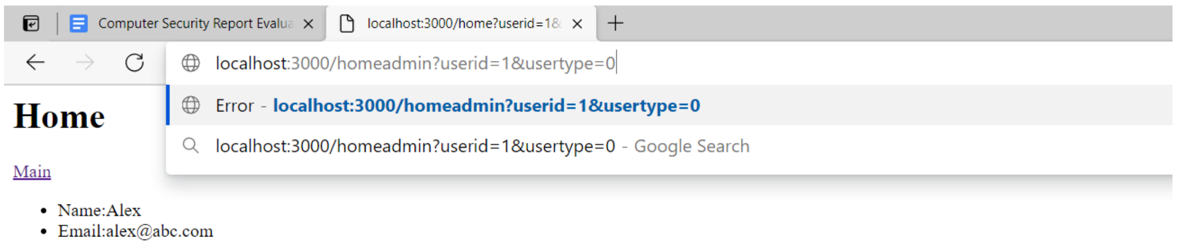

Computer Security Report Evalua
localhost:3000/home?userid=1&
localhost:3000/home?userid=1&usertype=1

Home

Main

- Name:Alex
- Email:alex@abc.com

User Home page

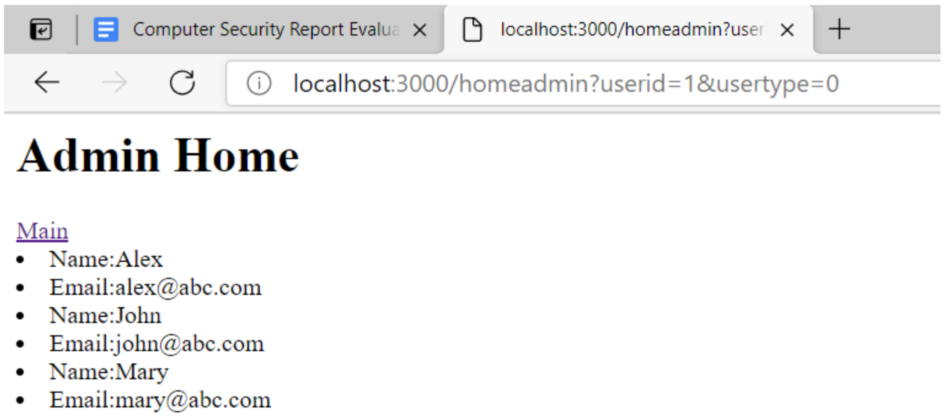

Computer Security Report Evalua
localhost:3000/home?userid=1&
localhost:3000/homeadmin?userid=1&usertype=0
Error - localhost:3000/homeadmin?userid=1&usertype=0
localhost:3000/homeadmin?userid=1&usertype=0 - Google Search

Home

Main

- Name:Alex
- Email:alex@abc.com

Attacker manipulating the route


Computer Security Report Evalua
localhost:3000/homeadmin?user
localhost:3000/homeadmin?userid=1&usertype=0

Admin Home

Main

- Name:Alex
- Email:alex@abc.com
- Name:John
- Email:john@abc.com
- Name:Mary
- Email:mary@abc.com

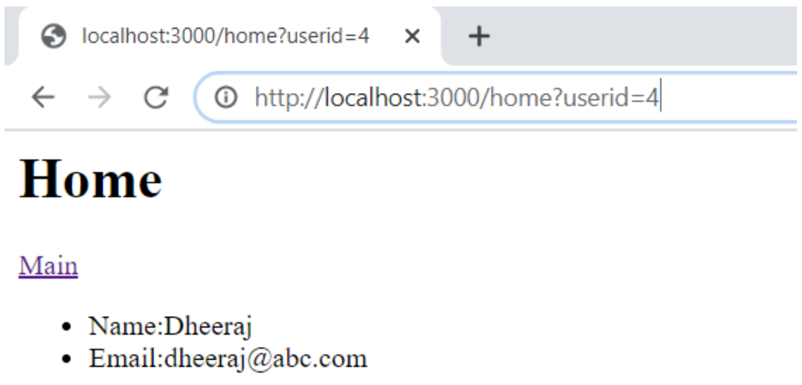
Attacker gains access to the admin home page

G. Preventing the attacks

a. Broken Authentication

The broken authentication attack is possible in this case because the user is able to manipulate the url parameters or the unencoded cookie values. If these parameters were secured and encoded with a secret known only to the controller, then the attacker cannot manipulate these parameters.

Therefore, the user id parameter used for authentication is embedded in the encoded cookie maintained by the server. This cookie can only be decoded with the secret known to the server. Therefore, the user cannot access the accounts of other users by just manipulating the url parameters or the unencoded cookie values.

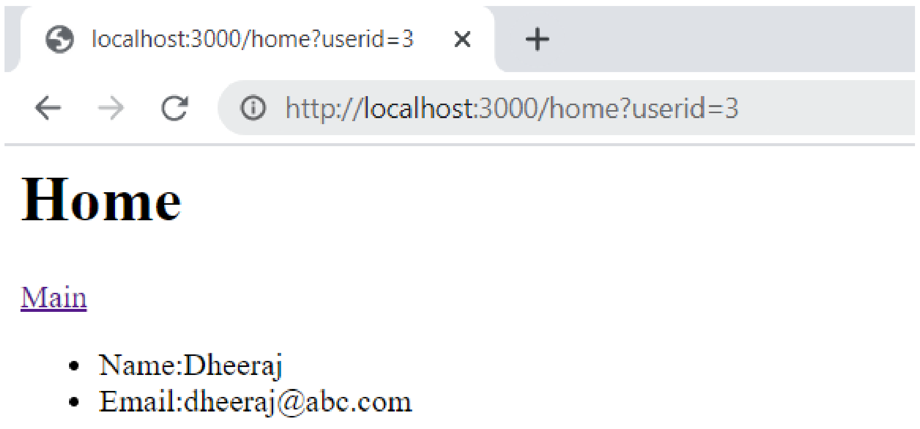

localhost:3000/home?userid=4
http://localhost:3000/home?userid=4

Home

Main

- Name:Dheeraj
- Email:dheeraj@abc.com

User logged in

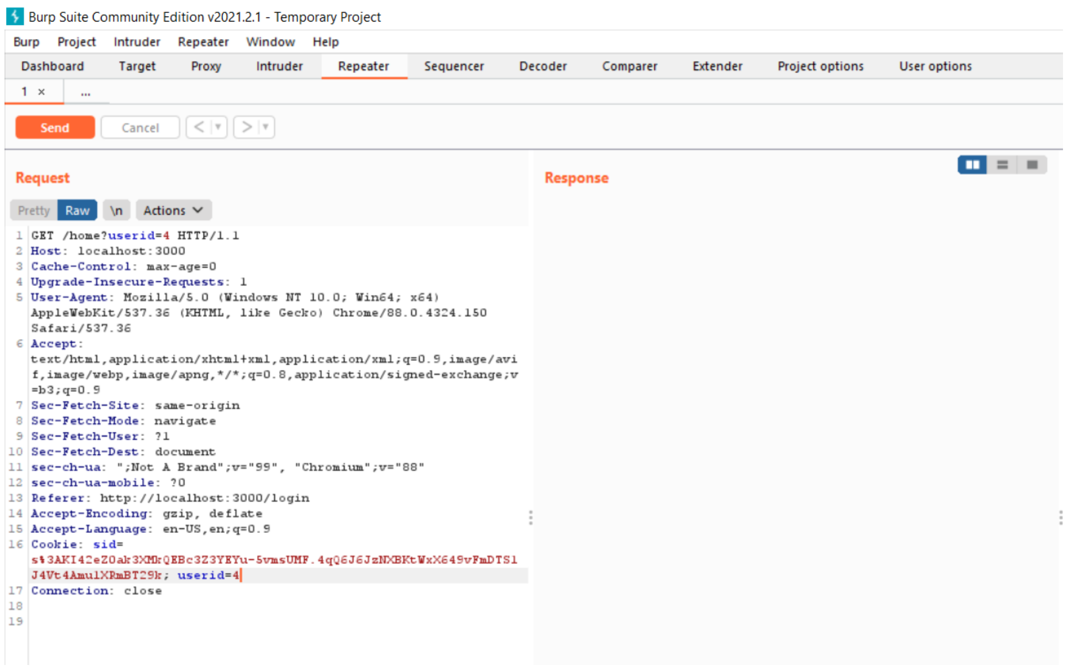

localhost:3000/home?userid=3
http://localhost:3000/home?userid=3

Home

Main

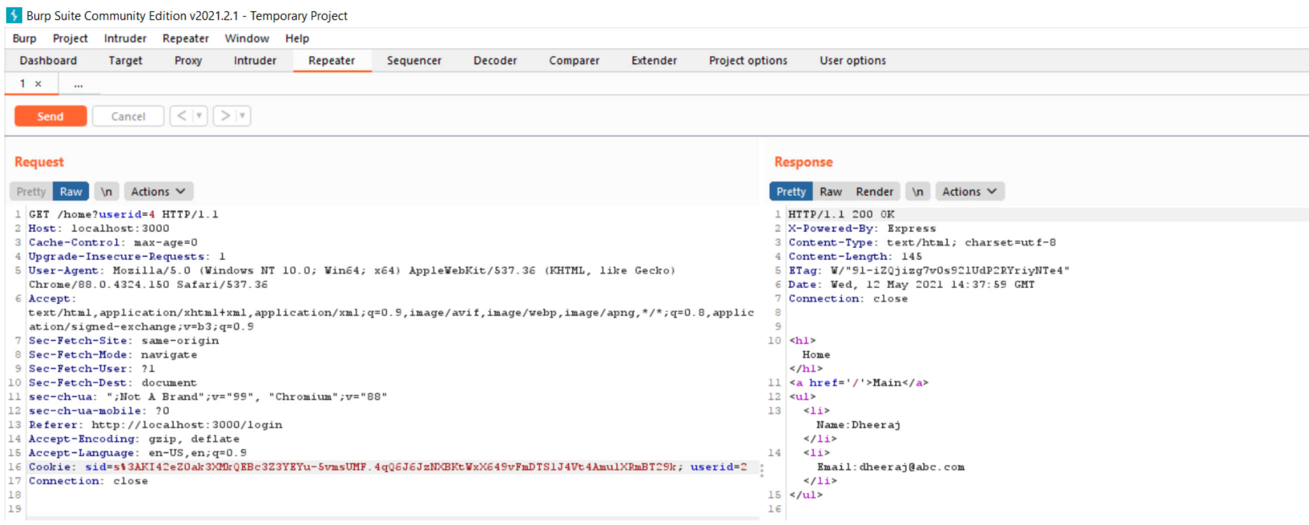
- Name:Dheeraj
- Email:dheeraj@abc.com

URL parameter changed but no change in the authentication or web page



```
GET /home?userid=4 HTTP/1.1
Host: localhost:3000
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.150
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avi
f,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v
=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
sec-ch-ua: ";Not A Brand";v="99", "Chromium";v="88"
sec-ch-ua-mobile: ?0
Referer: http://localhost:3000/login
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: sid=
s%3AKI42eZ0ak3XMkQEBc3Z3YEYu-5vmsUMF.4qQ6J6JzNXBKtWxX649vFmDTS1
J4Vt4AmulXRmBT29k; userid=4
Connection: close
```

Attacker trying to manipulate the captured cookie value



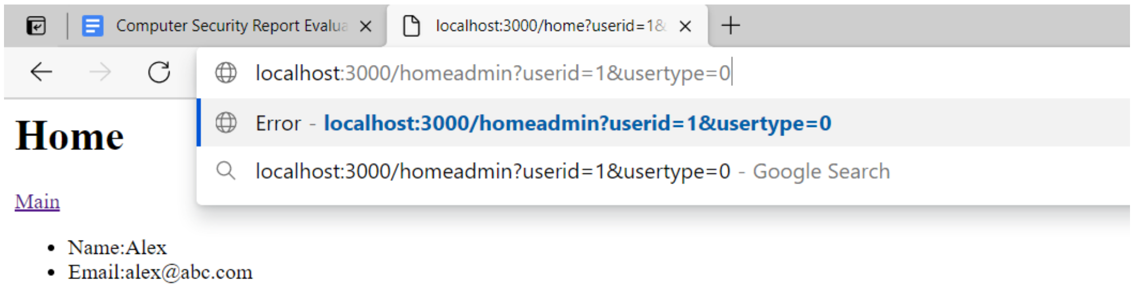
```
GET /home?userid=4 HTTP/1.1
Host: localhost:3000
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/88.0.4324.150 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applic
ation/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
sec-ch-ua: ";Not A Brand";v="99", "Chromium";v="88"
sec-ch-ua-mobile: ?0
Referer: http://localhost:3000/login
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: sid=s%3AKI42eZ0ak3XMkQEBc3Z3YEYu-5vmsUMF.4qQ6J6JzNXBKtWxX649vFmDTS1J4Vt4AmulXRmBT29k; userid=2
Connection: close
```

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/html; charset=utf-8
Content-Length: 145
ETag: W/"91-iZQjizg7v0s921UdP2RYriyNTe4"
Date: Wed, 12 May 2021 14:37:59 GMT
Connection: close


<h1>
  Home
</h1>
<a href='/'>Main</a>
<ul>
  <li>
    Name:Dheeraj
  </li>
  <li>
    Email:dheeraj@abc.com
  </li>
</ul>
```

Response of same user data

b. Broken Access Control

The broken access control attack in this case occurs because the webpages do not have route protection. The server does not check if the person requesting a particular page is legitimate and has the permissions to access a particular page. In that case, they take advantage of this situation and can request for admin pages. This is corrected by checking the user type before providing access to any page of the web application. A session variable having the type of the user is maintained to verify during the access time.

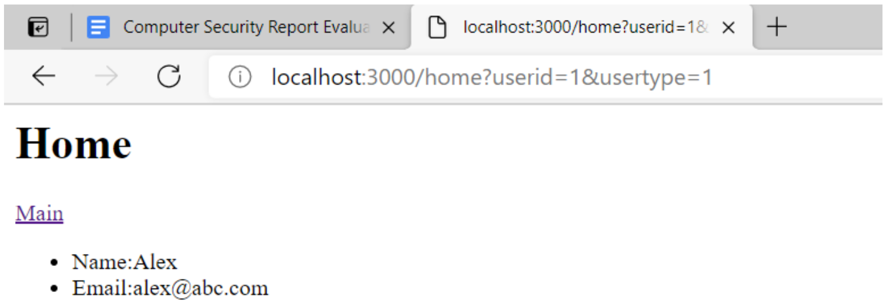

Computer Security Report Evalua
localhost:3000/home?userid=1&
localhost:3000/homeadmin?userid=1&usertype=0
Error - localhost:3000/homeadmin?userid=1&usertype=0
localhost:3000/homeadmin?userid=1&usertype=0 - Google Search

Home

Main

- Name:Alex
- Email:alex@abc.com

Logged in user trying to access admin pages


Computer Security Report Evalua
localhost:3000/home?userid=1&
localhost:3000/home?userid=1&usertype=1

Home

Main

- Name:Alex
- Email:alex@abc.com

The user is redirected back to the user page after checking the usertype from the session variable

H. Summary

Vulnerable Web application allows interested users to observe all the top 10 OWASP web vulnerabilities and along with it we have implemented and documented the attacks corresponding to each vulnerability. We have also discussed ways to prevent each of these attacks. The application will also enable developers to write safe code once they understand the different ways in which attacks can be performed to exploit the vulnerability.

I. References

a. https://owasp.org/www-project-top-ten/

b. https://hdivsecurity.com/owasp-broken-access-control

c. What is and how to prevent Broken Authentication | OWASP Top 10 2017 (A2)