

Assignment-1

Name: Shravani Abhisheki

Reg. no. : 21BIT0028

Mobile number:8698324619

e-mail : Shravani.abhisheki2021@vitstudent.ac.in

Date:30/08/23

Task-1

1.CWE: CWE 285- Improper Authorization

OWASP CATEGORY : A01 2021 Broken Access Control

DESCRIPTION:

The item fails to properly carry out or inadequately performs an assessment of authorization when an individual attempts to achieve entry to a particular asset or participate in a specific deed.

Business Impact:

The process of authorization encompasses the validation of whether an individual, holding a certain identity, possesses the right to approach a distinct resource. This decision relies on the privileges held by the individual and any regulations pertaining to access control that are allocated to the asset. In situations where measures for controlling access exhibit inconsistency or absence, individuals succeed in obtaining unsanctioned entry to information or actions they should not be entitled to. This could give rise to an array of predicaments, including the exposure of sensitive data, disruptions in service provision, and the execution of malevolent code.

2.CWE: CWE-916: Use of Password Hash With Insufficient Computational Effort

OWASP CATEGORY : A02 2021 Cryptographic Failures

DESCRIPTION:

The product employs a hashing technique for passwords, yet it utilizes a method lacking the requisite level of computational intricacy, failing to deter password-cracking endeavors of impractical or financially burdensome nature.

Business Impact:

In this context, the procedure of authentication involves the reception of an incoming password, followed by the computation of its hash, which is subsequently juxtaposed with the hash held within the system. Once a perpetrator gains access to stored password hashes, they acquire the ability to systematically initiate brute force assaults on these hashes within an offline environment. As a precautionary measure, defenders can exclusively strive to impede offline attacks by opting for hash algorithms that necessitate maximal utilization of computational resources.

3.CWE: CWE 564: SQL Injection: Hibernate

OWASP CATEGORY : A03 2021 Injection

DESCRIPTION:

The utilization of Hibernate for the execution of dynamic SQL statements, constructed utilizing input regulated by the user, has the potential to grant an unauthorized party the ability to modify the intended context of the statement or perform unrestrained SQL directives.

Business Impact:

Malicious actors leverage SQL injection attacks to attain unauthorized entry to valuable business data or personally identifiable information (PII). This results in an elevated susceptibility of sensitive information. Through SQL injection, malevolent entities can access and manipulate data, jeopardizing the confidentiality of vital corporate information retained within the SQL server.
Compromise of User Privacy: Depending on the nature of the information stored within the SQL server, an exploitation can lead to the exposure of personal user data, including delicate particulars such as credit card numbers.

4. CWE: CWE 653: Improper Isolation or Compartmentalization

OWASP CATEGORY : A04 2021 Insecure Design

DESCRIPTION:

The given product overlooks the established protocols of secure design, which in turn lays the groundwork for potential vulnerabilities or an elevated probability of developers inadvertently introducing similar vulnerabilities during the implementation phase. As the code structure is rooted in design principles, rectifying design-related issues could demand substantial resource allocation.

Business Impact:

Security vulnerabilities within system configurations stem from insufficiencies in the security parameters, setup processes, and fortification of various systems across the pipeline (such as Source Code Management, Continuous Integration, and Artifact Repositories). Frequently, these vulnerabilities serve as convenient targets for attackers aiming to establish a more commanding presence within the environment.

5. CWE: CWE 614-Sensitive Cookie in HTTPS Session Without 'Secure' Attribute

OWASP CATEGORY : A05 2021 Security Misconfiguration

DESCRIPTION:

The omission of the Secure attribute within HTTPS sessions concerning sensitive cookies has the potential to result in the user agent transmitting these cookies without encryption over an HTTP session.

Business Impact:

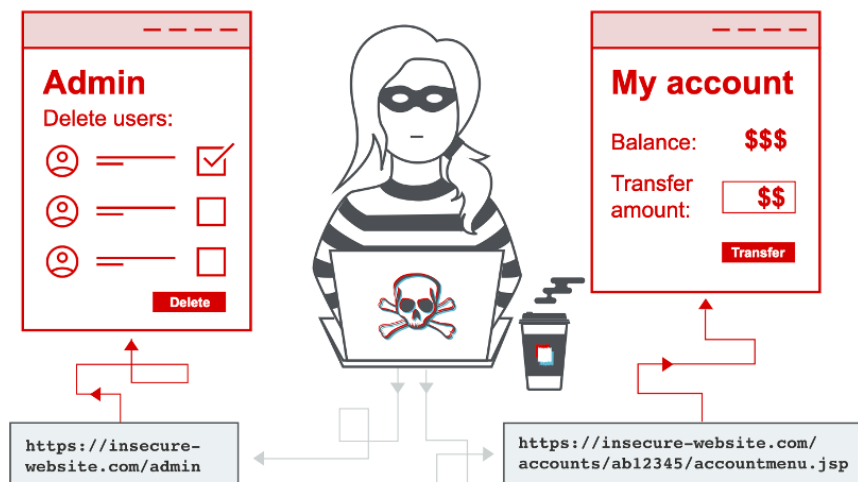
Lapses in security configurations furnish attackers with the means to illicitly infiltrate networks, systems, and data, leading to substantial financial losses and detrimental effects on your organization's reputation.

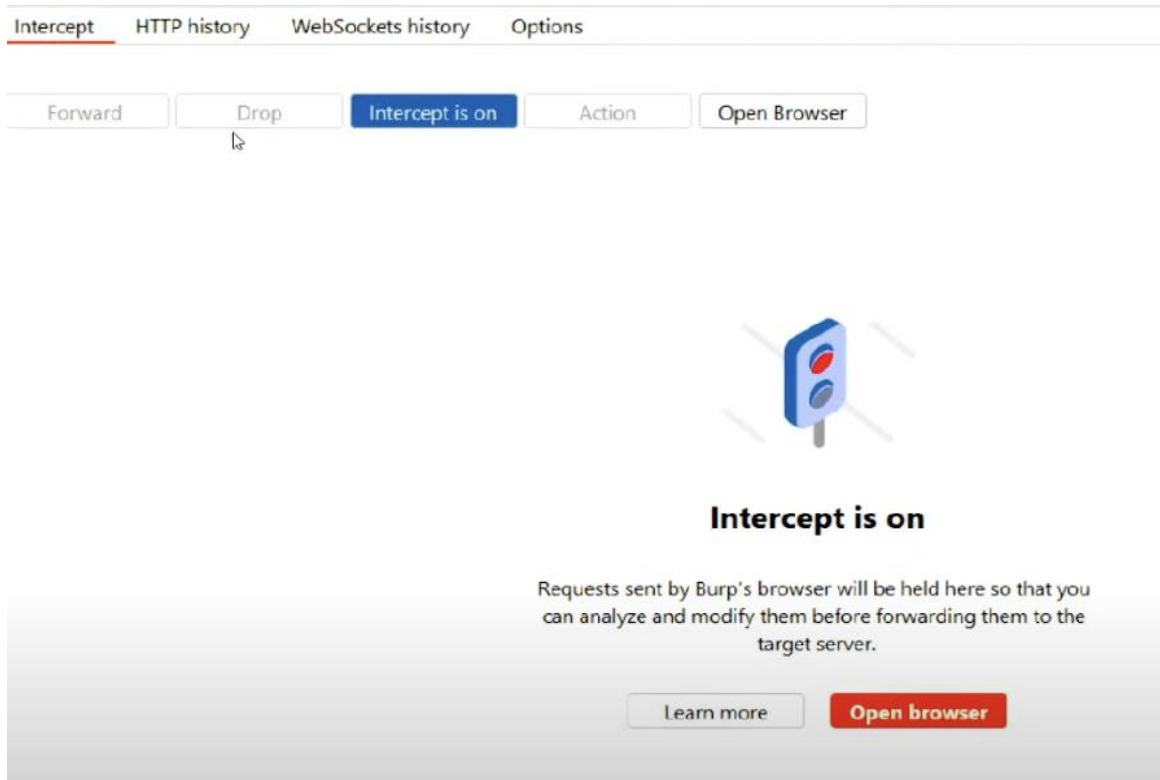
Task-2

Broken access controls: are a commonly encountered and often critical security vulnerability. Design and management of access controls is a complex and dynamic problem that applies business, organizational, and legal constraints to a technical implementation. Access control design decisions have to be made by humans, not technology, and the potential for errors is high.

From a user perspective, access controls can be divided into the following categories:

- [Vertical access controls](#)
- [Horizontal access controls](#)
- [Context-dependent access controls](#)





Login

Username

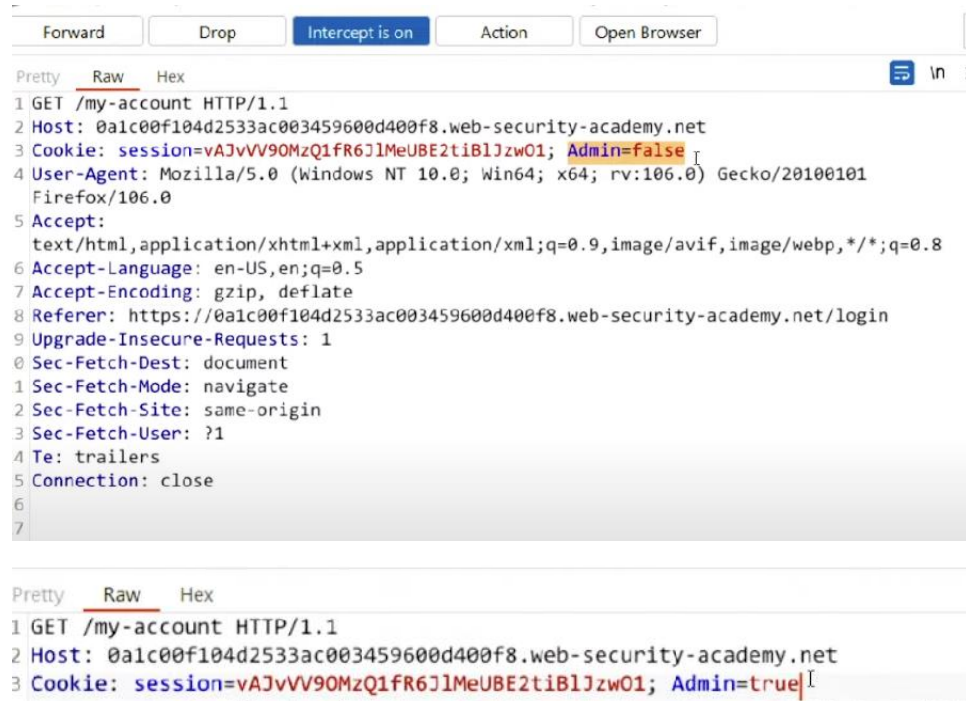
wiener I

Please fill out this field.

Password

•••••

Log in



Forward Drop Intercept is on Action Open Browser

Pretty Raw Hex

```
1 GET /my-account HTTP/1.1
2 Host: 0a1c00f104d2533ac003459600d400f8.web-security-academy.net
3 Cookie: session=vAJvVV90MzQ1fR6JlMeUBE2tiBlJzw01; Admin=false
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:106.0) Gecko/20100101
  Firefox/106.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://0a1c00f104d2533ac003459600d400f8.web-security-academy.net/login
9 Upgrade-Insecure-Requests: 1
0 Sec-Fetch-Dest: document
1 Sec-Fetch-Mode: navigate
2 Sec-Fetch-Site: same-origin
3 Sec-Fetch-User: ?1
4 Te: trailers
5 Connection: close
6
7
```

Pretty Raw Hex

```
1 GET /my-account HTTP/1.1
2 Host: 0a1c00f104d2533ac003459600d400f8.web-security-academy.net
3 Cookie: session=vAJvVV90MzQ1fR6JlMeUBE2tiBlJzw01; Admin=true
```

Users

carlos - [Delete](#)

wiener - [Delete](#)

Congratulations, you solved the lab!

User deleted successfully!

Users

wiener - Delete

Cryptographic Failure Vulnerability:

Less than 4 years ago, a very small (<10 employees) marketing and data aggregation firm called Exactis accidentally exposed its database that contained around 340 million individual records. Be it experience, negligence, or ignorance, the people in charge had put the database on a publicly accessible server. What that means is that anyone (anyone who knew where to look, that is) could access this data.

The exposed records included names, phone numbers, emails, and other sensitive data of **millions of US citizens**. And because this information was intended for highly targeted marketing purposes, it was much more detailed and personal than what people usually expose in an everyday data breach.

Request

Pretty Raw \n Actions ▾

```

1 GET /product?productId=2 HTTP/1.1
2 Host:
  0a4c00d604f371f6c0a15b2c00cb002d.web-security-academy.net
3 Cookie: session=
  bER4FrW4AdBi7i0erQlQdaHqYv2S5QaJ
4 User-Agent: Mozilla/5.0 (Windows NT
  10.0; Win64; x64; rv:106.0)
  Gecko/20100101 Firefox/106.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
  q=0.8

```

Response

Pretty Raw Render \n Actions ▾

```

1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 Connection: close
4 Content-Length: 4034
5
6 <!DOCTYPE html>
7 <html>
8   <head>
9     <link href=/resources/labheader/css/academyLabHeader.
10     <link href=/resources/css/LabsEcommerce.css rel=style
11     <title>
      Information disclosure in error messages
    </title>
12 </head>
13 <body>
14   <script src="/resources/labheader/js/labHeader.js">

```


Response

Pretty Raw Render \n Actions

```
1 HTTP/1.1 500 Internal Server Error
2 Connection: close
3 Content-Length: 1613
4
5 Internal Server Error: java.lang.NumberFormatException: I
6 at java.base/java.lang.NumberFormatException.forInputStri
7 at java.base/java.lang.Integer.parseInt(Integer.java:654)
8 at java.base/java.lang.Integer.parseInt(Integer.java:786)
9 at lab.c.y.j.c.S(Unknown Source)
10 at lab.c.n.c.a.V(Unknown Source)
11 at lab.e.n.s.u.w.o(Unknown Source)
12
13
14
15
16
17
18 at java.base/java.lang.Thread.run
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38 at java.base/java.lang.Thread.run
39
40 Apache Struts 2 2.3.31
```

Congratulations, you solved the lab!

Injection :

[Sign In](#) | [Contact Us](#) | [Feedback](#) | Search

Search Results

No results were found for the query:

demo.testfire.net

XSS

OK

Insecure Design:

What is Insecure Design?

Insecure design encompasses various risks that arise from **ignoring design** and **architectural best practices**, starting from the planning phase before actual implementation. A quick point to note here is that an insecure design differs from an insecure implementation, and a near-perfect implementation cannot prevent defects arising from an insecure design. While the Insecure design flaw is a new entrant to the **OWASP top 10**, it ranks number four on the 2021 list since mitigating risks at the design phase is considered fundamental toward 'Shift Left' security practices.

Vulnerabilities?

Insecure design vulnerabilities arise when developers, QA, and/or security teams **fail to anticipate** and **evaluate threats** during the code design phase. These vulnerabilities are also a consequence of the non-adherence of **security best practices** while designing an application. As the threat landscape evolves, mitigating design vulnerabilities requires consistent threat modeling to prevent known attack methods. Without a secure design, it is difficult to detect and remediate architectural flaws such as:

```
try { openDbConnection(); } //print exception message that includes exception message and
configuration file location catch (Exception $e) { echo 'Caught exception: ', $e->getMessage(), "\n";
echo 'Check credentials in config file at: ', $mysql_config_location, "\n"; }
```

```
SELECT * FROM vehicles WHERE type = 'SUV' AND registered=1
```

This query asks the backend to return all details from the **vehicle** table, where the category is **SUV**, and the value of the **registered** column is **1**. The restriction **registered=1** is used to **hide** existing vehicles that have not been registered.

In the absence of user-supplied URL validation, the malicious user can construct a URL like:

<https://darwin-cars.com/vehicles?category=SUV>

```
SELECT * FROM vehicles WHERE type = 'SUV' --' AND registered = 1
```

security misconfiguration vulnerability:

Here are a few real life attacks that caused damage to major organizations, as a result of security misconfigurations:

- Citrix legacy protocols attack – Citrix used an IMAP-based cloud email server and became the target of IMAP-based password-spraying. IMAP is an insecure, legacy protocol, and attackers exploited it to get access to cloud-based accounts and SaaS applications. Using multi factor authentication (MFA) could have stopped the attack.
- Mirai (未来) botnet – Mirai was a mega-scale botnet that infected network devices like CCTV cameras, DVD devices and home routers. The botnet exploited a misconfiguration in these devices – the use of insecure default passwords. The botnet was used to carry out DDoS attacks of unprecedented magnitude, which brought down websites like Twitter, Reddit, and Netflix.
-



CORS vulnerability with basic origin reflection

[Go to exploit server](#)[Submit solution](#)[Back to lab description >>](#)

Login

Username

Please fill out this field.

Password

Log in

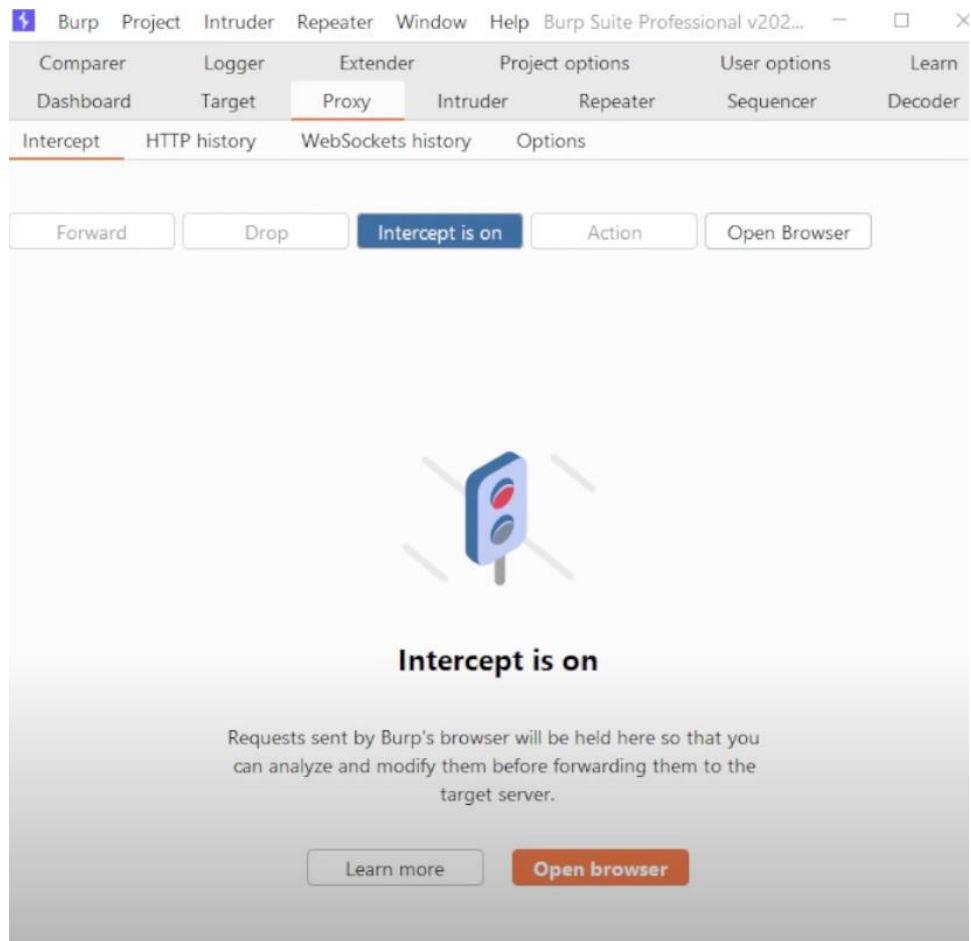
My Account

Your username is: wiener

Your API Key is: WlewWc1dlul6siyjccsNJPVXhsc5ORx7

Email

Update email



Intercept HTTP history WebSockets history Options

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	S
51	https://0a0200e003cd4dccc0c...	GET	/my-account			20
52	https://0a0200e003cd4dccc0c...	GET	/accountDetails			20
53	https://0a0200e003cd4dccc0c...	GET	/academyLabHeader			10
54	https://0a0200e003cd4dccc0c...	GET	/			20
55	https://0a0200e003cd4dccc0c...	GET	/resources/images/shop.svg			20
74	https://0a0200e003cd4dccc0c...	GET	/academyLabHeader			

Request

Pretty Raw Hex

```

1 GET / HTTP/1.1
2 Host:
0a0200e003cd4dccc0c9686a0075003b.web-security-academy.net
3 Cookie: session=IbwK1lFxxQLxX5kGBWcxBR4EWtjxOVqpy
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:107.0) Gecko/20100101 Firefox/107.0
5 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,im
age/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer:

```

Response

Pretty Raw Hex Render

```

1 HTTP/1.1 200 OK
2 Content-Type: text/html
3 Connection: close
4 Content-Length: 10767
5
6 <!DOCTYPE html>
7 <html>
8 <head>
9 <link href=
/resources/labhead
stylesheet>
10 <link href=/resour
stylesheet>

```

Match and Replace

These settings are used to automatically replace parts of requests and responses passing through the Proxy

	Enabled	Item	Match	Replace
Add	<input type="checkbox"/>	request header	^Accept-Encoding: *	
Edit	<input type="checkbox"/>	Response header	^Set-Cookie:*\$	
Remove	<input type="checkbox"/>	Request header	^Host: foo.example.org\$	Host: bar.example.org
Up	<input checked="" type="checkbox"/>	Request header		Origin: foo.example.org
Down	<input type="checkbox"/>	Response header	^Strict-Transport-Security: *	
	<input type="checkbox"/>	Response header		X-XSS-Protection: 0

