Juice Shop
Penetration Test Report

Version	**1.0**
Author	**Omar Hany Mohamed** **Omar Emad Elbaroudy** **Ahmed Medhat**
Issue Date	**9/6/2022**

Table of Contents

1.0 Juice Shop Test Report

1.1 Introduction

Subject of this document is a summary of penetration tests performed against web applications owned by Juice Shop company. Test was conducted according to rules of engagement defined and approved at the beginning by both parties – customer and contractor. Black-box pentesting assignment was requested.

Black-box penetration test classification means that penetration tester has no internal knowledge about target system architecture. He/she can use information commonly available on the Internet. More data can be collected during the reconnaissance phase based on observation of target system behavior. Black-box penetration test results give overview of vulnerabilities exploitable from outside the company network. It shows how unauthenticated actors can take advantage of weaknesses existing in tested web applications.

Time frame defined:

Penetration test start: 20/5/2022,

Penetration test end: 9/6/2022.

1.2 Scope

To perform a Black Box Web Application Penetration Test against the web applications of the organization named Juice Shop.

This is what the client organization defined as scope of the tests:

Dedicated Web Server: 127.0.0.1

Domain: localhost:3000

Subdomains: all subdomains

2.0 – EXECUTIVE SUMMARY

Conducted penetration test uncovered several security weaknesses present in web applications owned by Juice Shop company

When performing the penetration test, there were several alarming vulnerabilities that were identified on Juice Shop networks.

- 13 High, 4 Med, 12 Low issues have been identified.

2.1 – Recommendations

I recommend patching the vulnerabilities identified during the testing to ensure that an attacker cannot exploit these systems in the future. One thing to remember is that these systems require frequent patching and once patched, should remain on a regular patch program to protect additional vulnerabilities that are discovered at a later date.

3.0 – RISK ASSESSMENT

3.1 Likelihood

The likelihood is a measurement of the capacity to carry out an attack. The factor will be the difficulty or skill required to carry out the attack.

Risk	Description
Critical	An attacker is near-certain to carry out the threat event
High	An untrained user could exploit the vulnerability. The vulnerability is obvious or easily accessed
Medium	The vulnerability required some hacking knowledge to carry out the attack
Low	The vulnerability required significant time, skill , access and other resource to carry out the attack

3.2 Impact

The impact is a measurement of the adverse effect carrying out an attack would have on the organization.

Risk	Description
Critical	An attack would cause catastrophic or severe effect on operation, asset or other organization
High	An attack would severely degrade mission capability. The attack may result in damage to asset(data exposure)
Medium	An attack would degrade the mission capability. An attack would allow for primary function to application to resume, but at reduced effectiveness
Low	An attack would degrade mission capability in a limited capacity. The attack may result in marginal damage to assets

4.0 -- FINDINGS SUMMARY

Findings	Likelihood	Impact	Rating	Status
DOM XSS at Search Field	Medium	Low	Low	**Open**
SQL injection in login form	Medium	High	High	**Open**
IDOR at basket view	Medium	Medium	Medium	**Open**
IDOR at adding items to basket	Medium	Medium	Medium	**Open**
Business Logic vulnerability at the chatbot	High	Low	Low	**Open**
IDOR at Customer Feedback	Medium	High	High	**Open**
Reflected XSS at id parameter in track-result path	Medium	Medium	Medium	**Open**
Stored XSS at the comment field	Medium	Critical	High	**Open**
Business Logic at customer feedback	Medium	Low	Low	**Open**
IDOR at writing reviews on products	Medium	Medium	Medium	**Open**
Business Logic at adding an item to basket	Medium	High	High	**Open**
SQL injection to retrieve all users's data	Medium	High	High	**Open**
Stored XSS at email field in registration	High	High	High	**Open**
Lack of input validation in the password and repeated password fields while registering a user	High	Low	Low	**Open**
Upload a file of different type in complain	High	High	High	**Open**

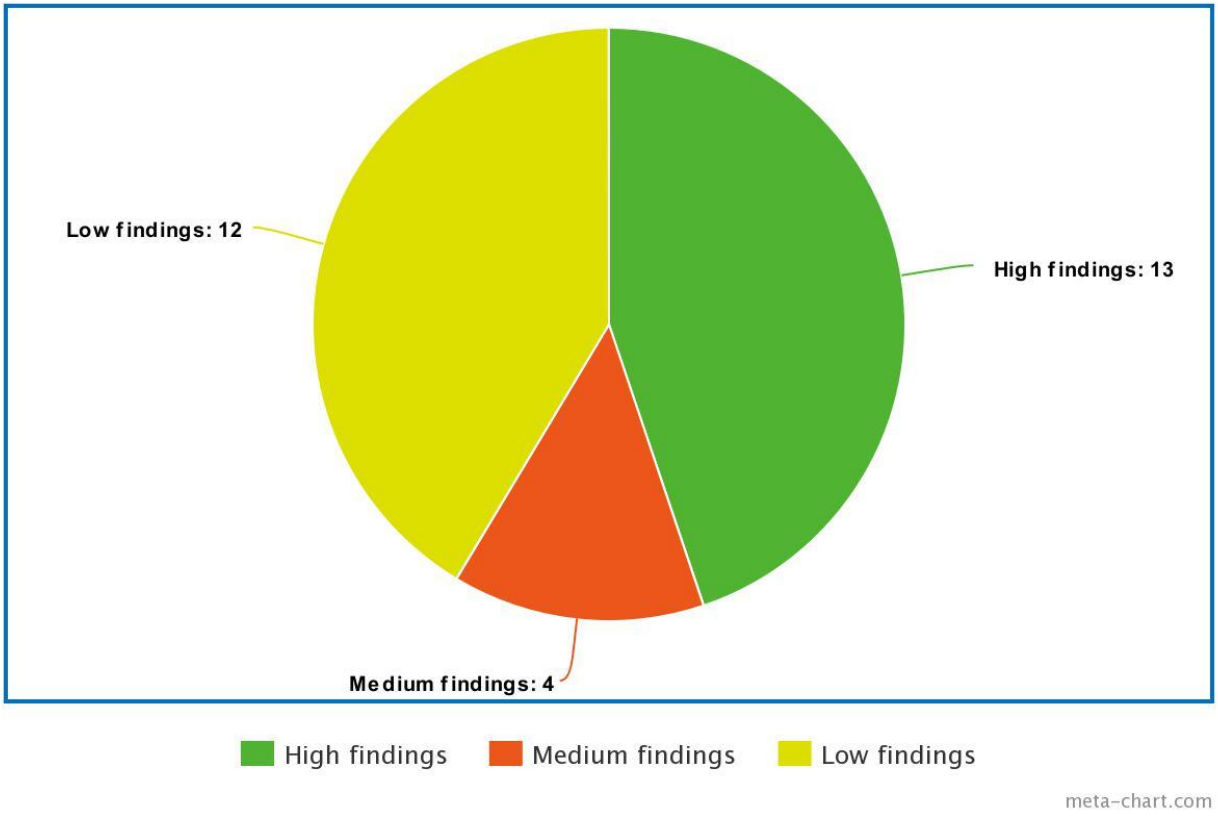
Upload a file of more than 100kb size	Medium	Medium	Medium	**Open**
Stored XSS in the username field	Low	High	High	**Open**
Bruteforce the security question	Low	Medium	Medium	**Open**
CSRF attack to change user's password	Medium	High	High	**Open**
Package.json backup exposure	Low	Low	Low	**Open**
Coupons file exposure	Low	Low	Low	**Open**
suspicious_errors File exposure	Low	Low	Low	**Open**
CSRF attack to change user's username	Medium	High	High	**Open**
Access logs files	Low	Low	Low	**Open**
Stored XSS in adding products to website	Low	High	High	**Open**
Stored XSS at last login	Low	High	High	**Open**
Registering a user with admin rule	Medium	High	High	**Open**
Bypassing captcha in feedback	Medium	Low	Low	**Open**
Open redirect	Medium	Low	Low	**Open**
Blind SSRF at image URL parameter in Link img	Low	High	High	**Open**

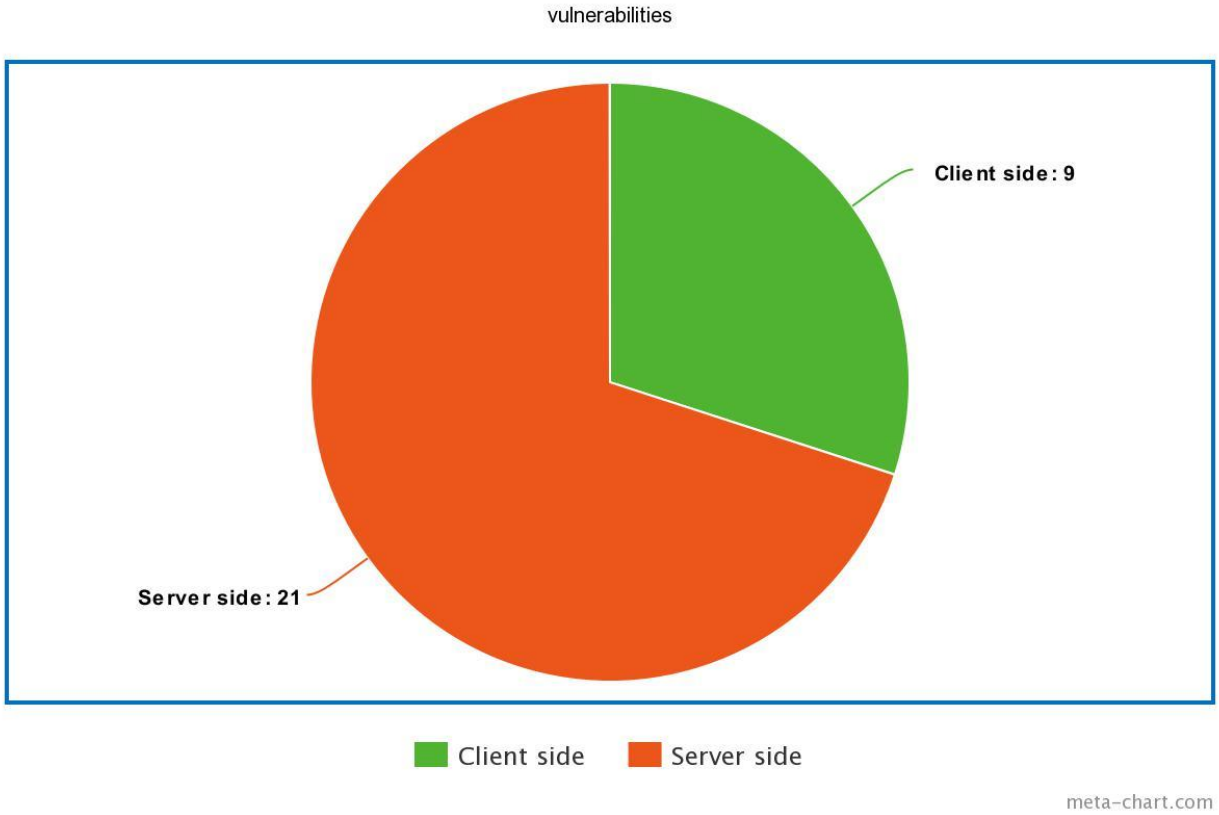
5.0 – VULNERABILITY & REMEDIATION REPORT

Penetration test finding classification, description and recommendations mentioned in the report are taken mostly from OWASP TOP 10 project documentation available on site: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project.

The OWASP TOP 10 is a list of definitions of web application vulnerabilities that pose the most significant security risks to organizations when exploited.

Vulnerability Summary




vulnerabilities
Client side : 9
Server side : 21
Client side
Server side
meta-chart.com

6.0 – INFORMATION GATHERING

The information gathering portion of a penetration test focuses on identifying the scope of the penetration test. I started the pentest by finding all subdomains.

I have used multiple tools to make sure that I haven't missed any domain.

Tools Used:

1) Burp Suite
2) Inspector tool in browser
3) Sublist3r

Above tools discovered multiple subdomains, the working one's are below:

http://localhost:3000/#/login
http://localhost:3000/#/profile
http://localhost:3000/#/contact
http://localhost:3000/rest/products/search
http://localhost:3000/#/register
http://localhost:3000/#/complain
http://localhost:3000/#/privacy-security/change-password
http://localhost:3000/api/Products
http://localhost:3000/#/ftp

7.0 High Findings

7.1. Title: SQL injection in login form

Rating: High

URL: http://localhost:3000/#/login

Description:

Due to this vulnerability an attacker can log in as the admin without knowing either his user name or password using SQL injection

Proof of Concept:

Step 1: visit http://localhost:3000/#/login

Step 2: Enter ' OR 1=1 -- in the username field

Step 3: Enter anything in the password field

Step 4: Click login

Step 5 : You are logged in as admin@juice-sh.op which is the first record in the users table

Remediation Steps:

Use input validation on both fields

7.2. Title: Stored XSS at the comment field in the contact path

Rating: High

URL: http://localhost:3000/#/contact

Description :

Proof of Concept:

Step 1 : Navigate to http://localhost:3000/#/contact

Step 2 : Paste the payload `<<script>Foo</script>iframe src="javascript:alert(`xss`)">` in the Comment section

Step 3 : Choose a rating and enter the result of the CAPTCHA and then and click Submit

Step 4 : Visit http://localhost:3000/#/about for a first "xss" alert

Step 5 : Visit http://localhost:3000/#/administration for another "xss" alert

Remediation Steps:

Have input validation on the comment field.

7.3. **Title:** Business Logic at adding an item to basket

Rating: High

URL: http://localhost:3000/#/

Description :

By this business logic vulnerability an attacker can add negative number of elements to his basket then proceed to checkout and gain money instead of paying

Proof of Concept:

Step 1 : Navigate to http://localhost:3000/#/

Step 2 : Add an item to basket

Step 3 : Intercept the request and change quantity to negative number

Step 4 : Proceed to checkout

Step 5 : You will get the negative amount in your account

Remediation Steps:

Have input validation and refuse any negative quantities

7.4. **Title:** SQL injection to retrieve all users's data

Rating: High

URL: http://localhost:3000/rest/products/search

Description :

By this SQL injection vulnerability, an attacker can retrieve the database records for all users.

Proof of Concept:

Step 1 : Navigate to http://localhost:3000/rest/products/search?q=test

Step 2 : change the value of q to be test`')) UNION SELECT id, email, password, '4', '5', '6', '7', '8', '9' FROM Users--`

Step 3 : All users data is shown

Remediation Steps:

Have input validation on the q parameter

7.5. **Title:** Stored XSS at email field in registration

Rating: High

URL: http://localhost:3000/#/register

Description :

By this XSS vulnerability, an attacker can inject javascript code in the admin panel and steal his/her cookies

Proof of Concept:

Step 1 : Go to login

Step 2 : Press not yet a customer

Step 3 : enter an email , password, repeated password, security question and answer

Step 4 : Click register

Step 5 : Intercept the packet and change the email parameter to something of this form "user@domain.com<script>alert(\"XSS2\")</script>"

Step 6: The code is injected successfully and when logging in as an admin and visiting the administration page an alert is reflected.

Remediation Steps:

Have input validation on the email parameter

7.6. Title: Upload a file of different type in complain

Rating: High

URL: http://localhost:3000/#/complain

Description :

By this XSS vulnerability, an attacker can upload a file of different type other than .zip and .pdf.

Proof of Concept:

Step 1 : Go to complain.

Step 2 : Write a message.

Step 3 : choose any .pdf or .zip file.

Step 4 : Intercept the packet

Step 5 : Choose another file with different extension

Step 6: Any file is uploaded

Remediation Steps:

Have server side input validation instead of the client side one.

7.7. Title: Stored XSS in the username field

Rating: High

URL: http://localhost:3000/#/profile

Description :

By this XSS vulnerability, an attacker can inject scripts in the profile page.

Proof of Concept:

Step 1 : Go to profile.

Step 2 : Use the payload `<<a|ascript>alert(`xss`)</script> in change username field`.

Step 3 : The script is injected successfully but not executed due to content security policy..

Step 4 : IEnter the following payload in the Image Url field https://a.png; script-src 'unsafe-inline' 'self' 'unsafe-eval' https://code.getmdl.io http://ajax.googleapis.com

Step 5 : Press Link Image

Step 6: The content security is updated and the script is run

Remediation Steps:

Have input validation on both the username field and the image url

7.8. Title: CSRF attack to change user's password

Rating: High

URL: **http://localhost:3000/#/privacy-security/change-password**

Description :

Change the password of any logged in user by utilizing change password vulnerability

Proof of Concept:

Step 1 : Visit **http://localhost:3000/#/login** and login using an existing email and password (e.g email:test@test.com password:password)

Step 2 : Navigate to **http://localhost:3000/#/privacy-security/change-password** and change password into password123 and click change.

Step 3 : Go to burp suite, check the HTTP history in proxy for the last GET request in this form GET /rest/user/change-password?current=password&new=password123&repeat=password123 HTTP/1.1

Step 4 : Right click, send request to repeater and then navigate to repeater

Step 5 : First try to change current field to any arbitrary text (e.g test) , click send and then notice in the response is of type 401 unauthorized

Step 6: Remove "current=password" from the request header GET /rest/user/change-password?new=password123&repeat=password123 HTTP/1.1 and

Step 7: Click send and then you will notice "200 OK" status in response, which means that everything is processed correctly

Step 8: Now we have to find some injection point to inject a script that perform the password change

Step 9: There is an XSS vulnerability in the search field.

Step 10 : We enter the following script in the search field :

<script>

xmlhttp =XMLHTTPRequest;

xmlhttp.open('Get',
'https://localhost:3000/rest/user/change-password?new=password123&repeat=password123');

xmlhttp.send();

</script>

Step 10: We get the link
http://localhost:3000/#/search?q=%3Cscript%3E%20xmlhttp%20%3DXMLHTTPRequest;%20xmlhttp.open('Get',%20'https:%2F%2Flocalhost:3000%2Frest%2Fuser%2Fchange-password%3Fnew%3Dpassword123%26repeat%3Dpassword123');%20xmlhttp.send();%20%3C%2Fscript%3E

and send it to a victim

Step 11: When the victim open the link password is changed to a new password

Remediation Steps:

Use csrf token

7.9. Title: CSRF attack to change user's username

Rating: High

URL: http://localhost:3000/#/privacy-security/change-password

Description :

Change the password of any logged in user by utilizing change password vulnerability

Proof of Concept:

Step 1 : Visit **http://localhost:3000/#/login** and login using an existing email and password (e.g email:test@test.com password:password)

Step 2 : Navigate to **http://localhost:3000/#/profile** and change username into CSRFand click change.

Step 3 : Go to burp suite, check the HTTP history in proxy for the last GET request in this form GET /profile?username="CSRF" HTTP/1.1

Step 4: Click send and then you will notice "200 OK" status in response, which means that everything is processed correctly and we know we can use this

Step 5: Now we have to find some injection point to inject a script that perform the password change

Step 6: There is an XSS vulnerability in the search field.

Step 7 : We enter the following script in the search field :

<script>

xmlhttp =XMLHTTPRequest;

xmlhttp.open('Get', 'https://localhost:3000/rest/user/change-password?username=’CSRF’);

xmlhttp.send();

</script>

Step 8: We get the link
http://localhost:3000/#/search?q=%3Cscript%3E%20xmlhttp%20%3DXMLHTTPRequest;%20xmlhttp.open('Get',%20'https:%2F%2Flocalhost:3000%2Frest%2Fuser%2Fchange-password%3Fusername%3D%E2%80%99CSRF%E2%80%99);%20xmlhttp.send();%20%3C%2Fscript%3E

and send it to a victim

Step 9: When the victim open the link password is changed to a new password

Remediation Steps:

Use csrf token

7.10. **Title:** Stored XSS in adding products to website

Rating: High

URL: http://localhost:3000/#/

Description :

By this XSS vulnerability, an attacker can inject scripts in the profile page.

Proof of Concept:

Step 1 : Sign in as an admin.

Step 2 : Send a post request to http://localhost:3000/api/Products using the body {"name": "XSS", "description": "<iframe src=\"javascript:alert(`xss`)\">", "price": 47.11} , admin token in header and `Content-Type : application/json`

Step 3 : The product is added successfully with a 200 OK reply

Step 4 : Search for a product called xss

Step 5 : An alert is popped

Remediation Steps:

Have input validation on both the product description

7.11. Title: Stored XSS in last login http://localhost:3000/#/privacy-security/last-login-ip

Rating: High

URL: http://localhost:3000/#/privacy-security/last-login-ip

Description :

By this XSS vulnerability, an attacker can inject scripts in the last login page.

Proof of Concept:

Step 1 : Sign in with any credentials..

Step 2 : Click logout and send the http://localhost:3000/rest/saveLoginIp request to repeater

Step 3 : add True-Client-IP header to the packet with value = <iframe src="javascript:alert(xss)">

Step 4 : Send the packet

Step 5 : Visit **http://localhost:3000/#/privacy-security/last-login-ip and an alert is popped**

Remediation Steps:

Have input validation on both the last login value

7.12. Title: Registering user with admin rule

Rating: High

URL: http://localhost:3000/#/register

Description :

By this a user can register himself as an admin

Proof of Concept:

Step 1 : Go to register page

Step 2 : Enter credentials

Step 3 : Click register and and intercept the packer

Step 4 : add “rule” : “admin” inside the json

Step 5 : a user of admin rule is registered successfully

Remediation Steps:

Never allow to register an admin rule except from the localhost of the server hosting the site.

7.13. Title:Blind SSRF at image URL parameter in Link img

Rating: High

URL: http://localhost:3000/#/profile

Description :

By this a user download any content using the server

Proof of Concept:

Step 1 : Go to profile

Step 2 : Enter a link in Image url even though it is not a link of an image

Step 3 : The content in downloaded in the server successfully and can be retrieved using a dns call

Remediation Steps:

Do input validation in the Image url parameter .

8.0 Medium Findings

8.1. Title: Insecure Direct Object Reference (IDOR) at basket

Rating: Medium

URL: http://localhost:3000/#/basket

Description:

By this vulnerability the attacker can access other users baskets by changing the bid parameter in the session storage

Proof of Concept:

Step 1: Visit http://localhost:3000/#/basket

Step 2: Right click and click on inspect elements

Step 3: Click on Application

Step 4: Click on Session Storage

Step 5: Click on http://localhost:3000

Step 6: Change bid value to other value

Step 7: Refresh the page

Step 8 : Now you can access other user basket

Remediation Steps:

Don't view basket based on Session storage and instead based on token

8.2. Title: IDOR at adding items to basket

Rating: Medium

URL: http://localhost:3000/#/

Description:

By this vulnerability an attacker can add an item to a basket of another user

Proof of Concept:

Step 1: Visit http://localhost:3000/#/

Step 2: Add any item to basket

Step 3: Intercept the packet using Burp Suite

Step 4: You will find this post request POST /api/BasketItems/ HTTP/1.1 with the following body

{"ProductId":6,"BasketId":"2","quantity":1}

Step 5: We change the body to add to another basket of something like this

{"ProductId":6,"BasketId":"1","quantity":1,"BasketId": "2"}

Step 6: You send the packer

Step 7: You successfully add an item to another user

Remediation Steps:

Verification that a specific user only add item to his basket by token

8.3. Title: Upload a file of more than 100kb size

Rating: Medium

URL: http://localhost:3000/#/profile

Description :

By this XSS vulnerability, an attacker can upload a file of different types other than .zip and .pdf.

Proof of Concept:

Step 1 : Go to profile.

Step 2 : upload an image of size less than 100 kb.

Step 4 : Intercept the packet

Step 5 : Choose another file with size more than 100kb

Step 6: Any file is uploaded

Remediation Steps:

Have server side input validation instead of the client side one.

8.4. Title:Bruteforce the security question

Rating: Medium

URL: http://localhost:3000/#/forgot-password

Description :

By this vulnerability, an attacker can get access to the user's account by brute forcing the security question answer.

Proof of Concept:

Step 1 : Go to Login.

Step 2 : Forget Password.

Step 4 : Enter an email of the victim

Step 5 : Enter a new password and repeat in the repeated password and enter anything is the security question answer

Step 6: Intercept the packet and try bruteforce the answer or you can limit the number of trials by getting a list of a reasonable answers of the question

Step 7: You will get the correct answer

Remediation Steps:

Apply bruteforce prevention techniques either by blocking an ip when making large number of request or requiring a captcha to continue

9.0 Low Findings

9.1. Title: DOM XSS at search field

Rating: Low

URL: http://localhost:3000/#/

Description :

Proof of Concept:

Step 1 : Visit http://localhost:3000/#/

Step 2 : Enter the payload `<iframe src="javascript:alert(`xss`)"> into the search field`

Step 3 : Click Enter

Step 4: Alert will pop up with text “xss”

Remediation Steps:

Do input validation on the search field.

9.2. Title: Business Logic vulnerability at the chat bot

Rating: Low

URL: http://localhost:3000/#/chatbot

Description :

By using this business logic vulnerability an attacker can receive a large number of discount coupons

Proof of Concept:

Step 1 : Click on the side bar

Step 2 : Click Support Chat

Step 3 : Tell the bot your name

Step 4 : Keep asking the chatbot 'Can I have a coupon code?'

Step 5 : You will receive a 10% coupon

Remediation Steps:

Never allow the chatbot to give coupons

9.3. Title: IDOR at Customer Feedback

Rating: Low

URL: http://localhost:3000/#/contact

Description :

By using this IDOR vulnerability an attacker can post a feedback user another user's username

Proof of Concept:

Step 1 : Click on the side bar

Step 2 : Click Customer Feedback

Step 3 : Inspect the page

Step 4 : Remove the hidden in the input tag `<input _ngcontent-c23 hidden id="userId" type="text" class="ng-untouched ng-pristine ng-valid">`

Step 5 : Change the id to another user's id

Step 6 : Write the feedback

Step 7 : Answer the mathematical question

Step 8 : Click submit

Step 9 : You know entered a feedback as if you are another user

Remediation Steps:

Base the feedback authentication on the token and not on the hidden input form

9.4. Title: Reflected XSS attack at the id parameter in the track-result

Rating: Low

URL: http://localhost:3000/#/

Description :

Proof of Concept:

Step 1 : Sign up and login

Step 2 : Add products to basket

Step 3 : Place an order after specifying a delivery address and credit card credentials

Step 4 : Navigate to http://localhost:3000/#/order-history

Step 5 : Click on the little truck icon button to track the status of your order.

Step 6 : Notice the id parameter in the URL in the form
http://localhost:3000/#/track-result?id=a0c9-c9272915cd5e11f5

Step 7 : Paste the payload `<iframe src="javascript:alert(`xss`)">` into the id parameter field in the URL

Step 8 : Reload the page then you will notice a pop up alert.

Remediation Steps:

9.5. Title: Business Logic at customer feedback

Rating: Low

URL: http://localhost:3000/#/contact

Description :

By this business logic vulnerability, users can give a zero star rating to the website even though the minimum is 1 star.

Proof of Concept:

Step 1 : Visit http://localhost:3000/#/contact

Step 2 : Write a comment in the comment field

Step 3 : give any rating using the slider

Step 4 : Solve the mathematical problem

Step 5 : Click submit

Step 6 : Intercept the packet and change the rating parameter to 0

Remediation Steps:

Have input validation and only include number from 1 to 5

9.6. Title: IDOR at writing reviews on products

Rating: Low

URL: http://localhost:3000/#/

Description :

By this IDOR vulnerability, users can write a can write a review on any product using any other user's username,

Proof of Concept:

Step 1 : Visit http://localhost:3000/#/

Step 2 : Click on any product

Step 3 : Write a review

Step 4 :Click submit

Step 5 : Intercept the packet and change the author to any other user's email

Remediation Steps:

Base the review authentication on the token and not to a passed parameter

9.7. Title: Lack of input validation in the password and repeated password fields while registering a user

Rating: Low

URL: http://localhost:3000/#/register

Description :

There is a lack of input validation in the passwords and repeated password fields

Proof of Concept:

Step 1 : Visit http://localhost:3000/#/register

Step 2 : Fill email and security question fields

Step 3 : Type identical password in the password and repeated password fields

Step 4 : Now change the password field into any other password that is at least 5 characters long. Then you will notice that the “passwords do not match” error did not appear

Step 5 : Click register

Step 6: Login with the password entered in step 4

Remediation Steps:

Do input validation in the password and repeated password and check they are the same.

9.8. Title: package.json file backup exposure

Rating: Low

URL: http://localhost:3000/ftp

Description :

There are some files exposed in this path.

Proof of Concept:

Step 1 : Visit http://localhost:3000/ftp

Step 2 : click package.json.back

Step 3 : An error appears

Step 4 : Change the link to http://localhost:3000/ftp/package.json.bak%2500.md to bypass the black list validation

Step 5 : File is downloaded

Remediation Steps:

Hide all files and make them inaccessible

9.9. Title: Coupons File exposure

Rating: Low

URL: http://localhost:3000/ftp

Description :

There are some files exposed in this path.

Proof of Concept:

Step 1 : Visit http://localhost:3000/ftp

Step 2 : click coupons_2013.md.bak

Step 3 : An error appears

Step 4 : Change the link to http://localhost:3000/ftp/coupons_2013.md.bak%2500 to bypass the black list validation

Step 5 : File is downloaded

Remediation Steps:

Hide all files and make them inaccessible

9.10. Title: suspicious_errors File exposure

Rating: Low

URL: http://localhost:3000/ftp

Description :

There are some files exposed in this path.

Proof of Concept:

Step 1 : Visit http://localhost:3000/ftp

Step 2 : click suspicious_errors.yml

Step 3 : An error appears

Step 4 : Change the link to http://localhost:3000/ftp/suspicious_errors.yml%2500.md to bypass the black list validation

Step 5 : File is downloaded

Remediation Steps:

Hide all files and make them inaccessible

9.11. Title: Access logs files

Rating: Low

URL: http://localhost:3000/support/logs

Description :

There are some files exposed in this path.

Proof of Concept:

Step 1 : Visit http://localhost:3000/support/logs

Step 2 : click access.logs.date

Step 3 : File is downloaded

Remediation Steps:

Hide all files and make them inaccessible

7.11. Title: Bypassing Captcha in feedback

Rating: Low

URL: http://localhost:3000/#/contact

Description :

By this vulnerability an attacker can send many feedbacks with the same captcha

Proof of Concept:

Step 1 : Visit http://localhost:3000/support/contact

Step 2 : write w message, choose a rating and solve the captcha

Step 3 : click submit and capture the request

Step 4 : send to repeater and send the feedback as many times as the request contains the same captcha id every time

Remediation Steps:

Hide the captcha id from the request

9.12. Title: Open Redirect

Rating: Low

URL: http://localhost:3000/redirect

Description :

By this vulnerability an attacker can a link that seems to be related to juiceshop but then redirect to any website

Proof of Concept:

Step 1 : Visit
http://localhost:3000/redirect?to=https://evil.com/?https://github.com/bkimminich/juice-shop

Step 2 : You will find yourself at evil.com

Remediation Steps:

Do validation on redirects