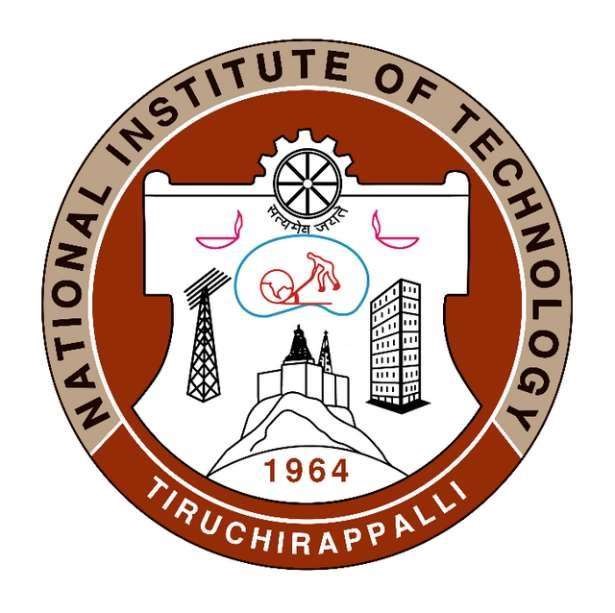
NAtional Institute of Technology

Tiruchirappalli



Department of Computer Applications

MCA Iv SEM

topic – database security issues and solutions

|  |
| --- |
| Database security |
| through securing api data |
|  |
|  |
| **submitted to - monika lakshmi mam** |
| submitted by - Purushottam patidar |
| 205121075 |

|  |  |
| --- | --- |
| table of contents | Problem definition & solution |
| related & proposed work |
| summary and architecture diagram |
| Module explanation & source |
| source code + software used + screenshot |
|  |
|  |
|  |
| api security |

# introduction

JWTs can be signed using a secret (with the **HMAC** algorithm) or a public/private key pair using **RSA** or **ECDSA**.

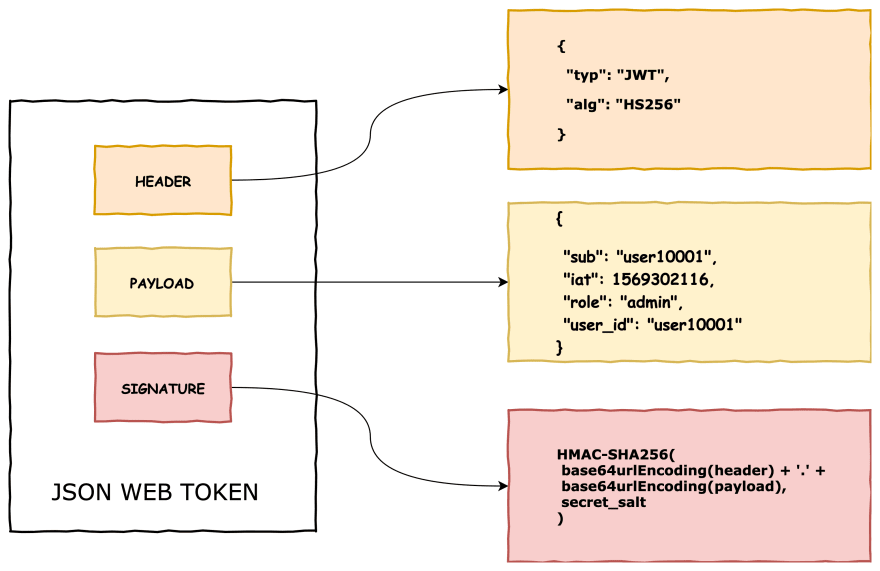
Although JWTs can be encrypted to also provide secrecy between parties, we will focus on signed tokens. Signed tokens can verify the integrity of the claims contained within it, while encrypted tokens hide those claims from other parties. When tokens are signed using public/private key pairs, the signature also certifies that only the party holding the private key is the one that signed it.

|  |
| --- |
| prolem in modern api |
| Various security risks in modern API   * **Excessive Data Exposure** * **Broken authorization** * **Lack of authentication security** * **Stealing credentials**   **Note:** Solutions provided using various security techniques, but I am explaining one of the efficient ways (JWT) to protect data from unauthorized user. |

|  |  |
| --- | --- |
| problem solutions | |
| Handshake with solid fill | #1 JWT token |
| Authentication technique |
| Customer review with solid fill | #2 Encrypt whole data |
| From unauthorized and threat activity |
| Business Growth with solid fill | #3 Oauth |
| Authorization of an application |

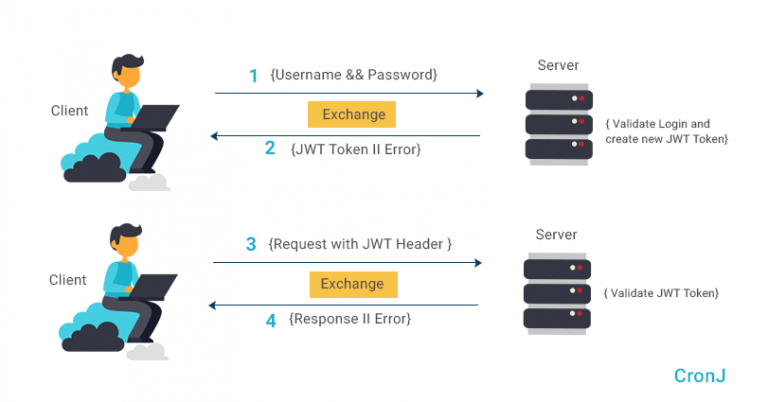
# JSON Web Tokens

 JSON tokens look like

JSON web tokens consist of three parts:

|  |  |
| --- | --- |
| why we used? | |
| **Securely transmitting information between parties as a JSON object and provide following services -** | |
| Authorization - | |
| **allowing the user to access routes, services, and resources that are permitted with that token.** |  |
| Information Exchange - | |
| **using public/private key pairs—you can be sure the senders are who they say they are. you can also verify that the content hasn't been tampered with.** |  |
| Other | |
| * **No session to manage** * **Portable : A single token can be used with multiple backends.** * **No cookies required.** * **Reduce network round trip time.** |  |

Architecture Diagram –

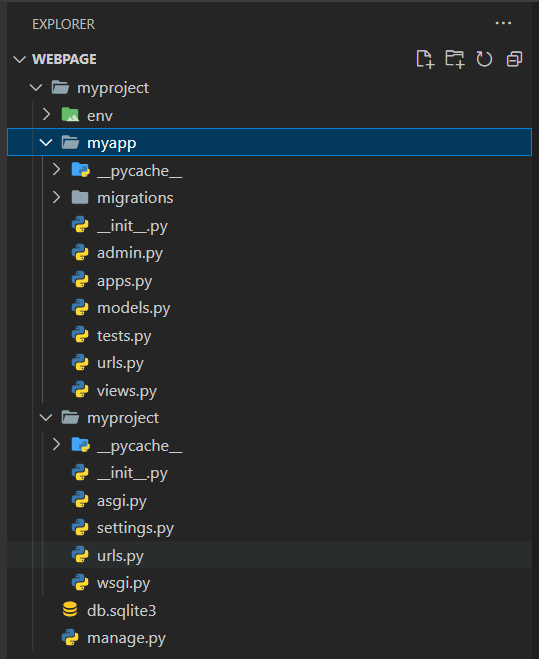


Summary –

* 1. **an authorization server creates a JWT.**
  2. **signs it and give it to client then,**
  3. **client will then send this JWT to a REST API**
  4. **REST API will verify and provide access.**

**Source Code**

#Hierarchy of files



**#myproject**

**|**

**#myapp**

**|**

**|----🡪urls.py**

**|----🡪views.py**

**urls.py**

from django.urls import path

from rest\_framework\_simplejwt.views import TokenObtainPairView, TokenRefreshView

from .views import ObjectListView

urlpatterns = [

    path('objects/', ObjectListView.as\_view(), name='object-list'),

    path('api/token/', TokenObtainPairView.as\_view(), name='token\_obtain\_pair'),

    path('api/token/refresh/', TokenRefreshView.as\_view(), name='token\_refresh'),

]

**views.py**

from django.shortcuts import render

from rest\_framework.views import APIView

from rest\_framework.response import Response

from rest\_framework.authentication import TokenAuthentication

from rest\_framework.permissions import IsAuthenticated

from rest\_framework\_simplejwt.authentication import JWTAuthentication

class ObjectListView(APIView):

    # authentication\_classes = [JWTAuthentication]

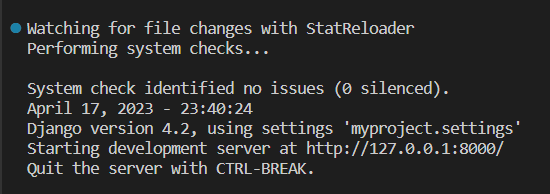
    # permission\_classes = [IsAuthenticated]

    def get(self, request):

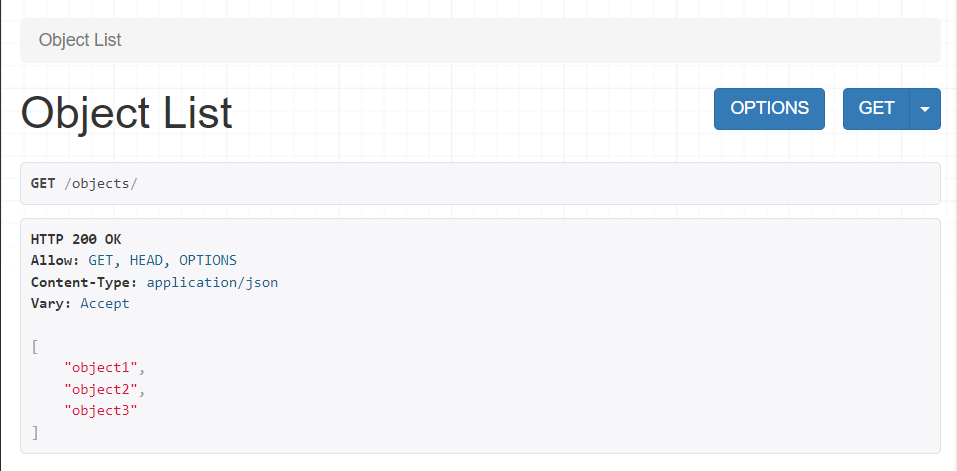
        objects = ["object1", "object2", "object3"]

        return Response(objects)

**#Step 1 : Run the Project**

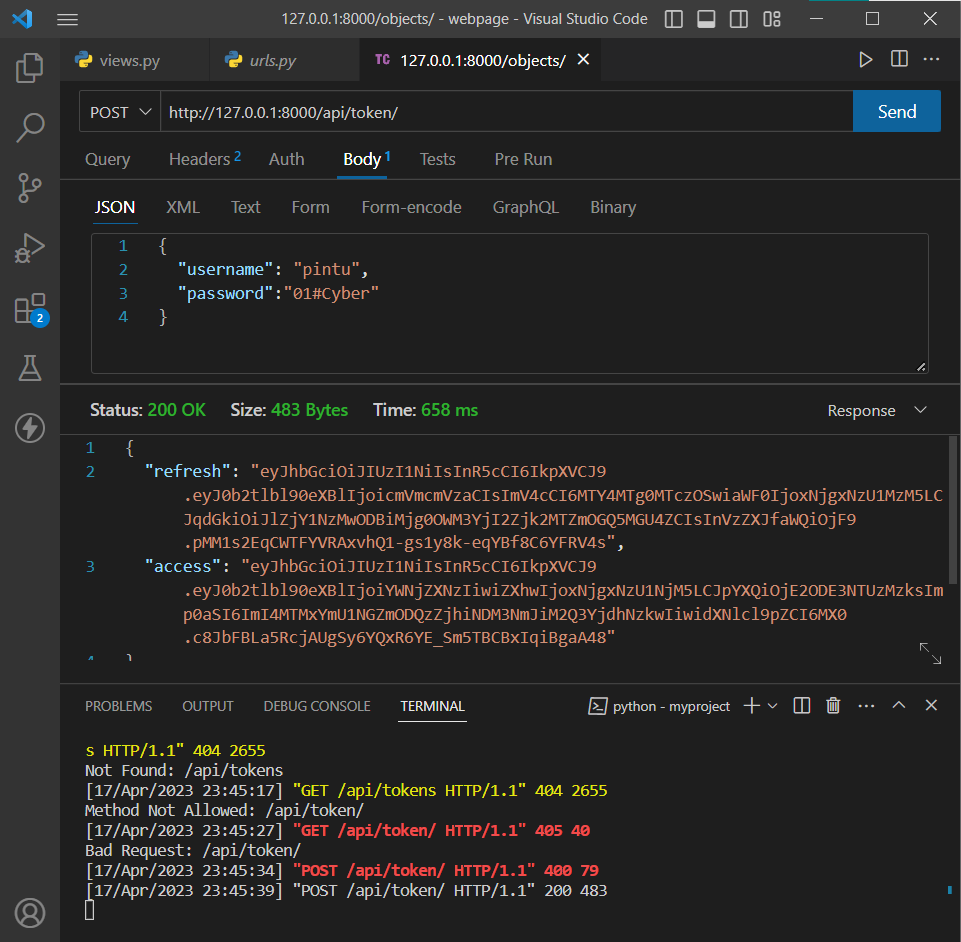


**#Step 2 : Open the localhost in browser ( Without JWT any one can see object data of API )**



**Applying JWT based authentication**

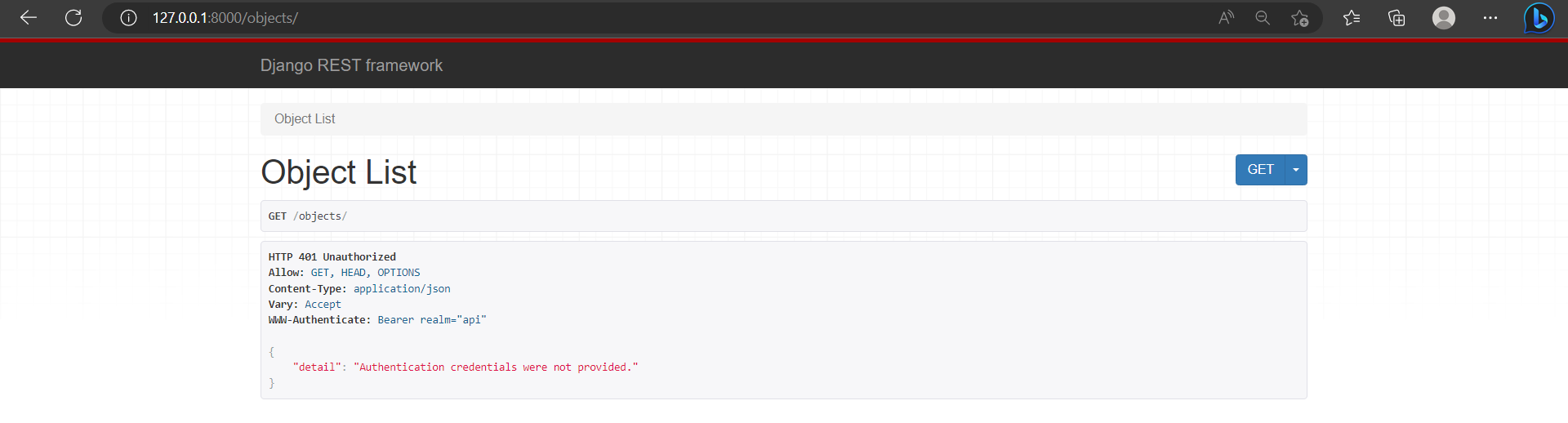
**#Step 3 : POST request to get authentication tokens**



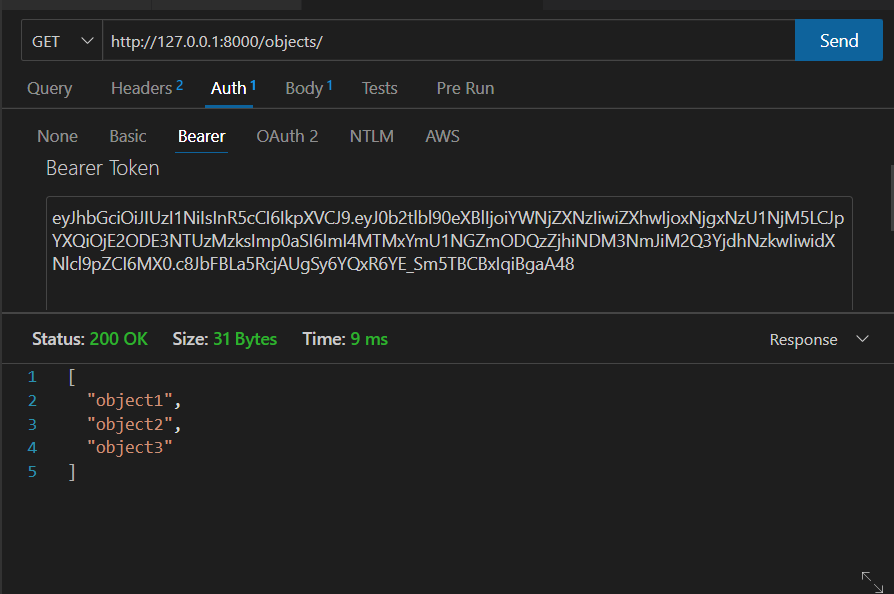
Note : Here i’m using **Thunderbolt extension** of **VScode** to check the request/response activity.

After accepting the tokens and applying authentication

only who have token can access the api data.



**#Step 4: Copy access tokens and then get request to access data**



Software :

* Visual studio code
* Thunderbolt Extension (api request / response)

Technology used :

* Python with Django framework