GRROUP 8

APRIL 26, 2022

FINAL REVIEW
PROJECT EXHIBITION II

Command line tool for working with JSON Web Tokens (JWT)

Mr. Vikas Kumar Jain

TEAM GUIDE

Meet Our Team

KHUSHI GARG

20BCY10043

VINAYAK AGRAWAL

20BCY10062

SHAILI GUPTA

20BCY10134

ANURAG SOURAV

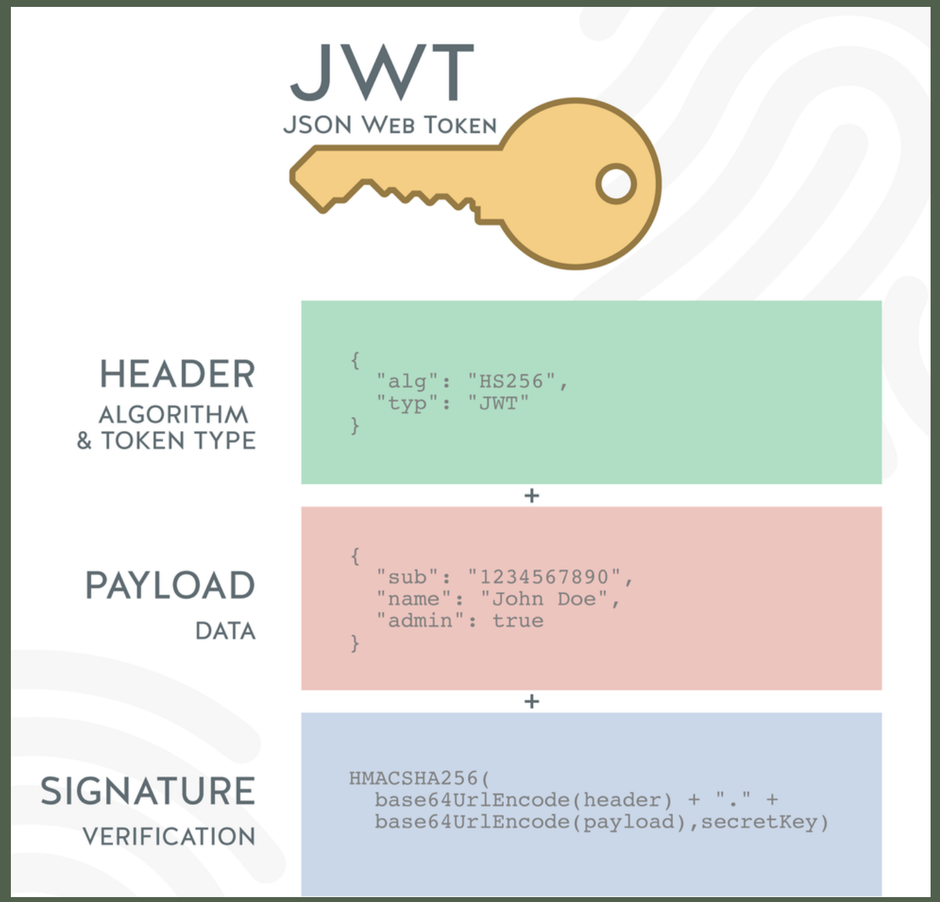
20BCY10146

INTRODUCTION

(JWT TOKEN)

- JWTs are an encoded representation of a JSON object.
- It is widely used these days for authorization purposes.
- It is used to send information between parties in a compact and secure manner.
- It can have different usages: authentication mechanism, url-safe encoding, securely sharing private data, interoperability, data expiration, etc.
- JWTs can be used by a server to tell the client app what actions the user is allowed to execute.

JWT Structure



Token itself consists of three parts: header, payload and signature.

- header contains encoded type of the token and algorithm
- payload contains encoded data and additional metadata
- signature is encoded header + encoded data + secret phrase, encoded with algorithm

Problem Statement

- Session IDs are vulnerable to session fixation attacks.
- Session value does not timeout or does not get invalidated after logout
- Passwords, session IDs, and other credentials are sent over unencrypted connections.
- Proper session authentication is required which can be done by JWT tokens.

Proposed Meihod

This tool will help in session management and secure authentication.

- Server generates an "accessToken", encrypting the "userId" and "expiresIn", with the ACCESS_TOKEN_SECRET, and sends the "accessToken" to the browser (client side).
- The browser (client side) receives the "accessToken" and saves it on the client side.
- The "accessToken" is included in every subsequent request to the server.

Objective

- To create a terminal based tool using Java Script and it's libraries for generating JWT Token using the method of encryption and decryption.
- Many tools exists which uses cookie based authentication we will be using it using JWT.

Real Time Usage

Application Security

Session Id Management

Secure Authentication

Capture the Flags

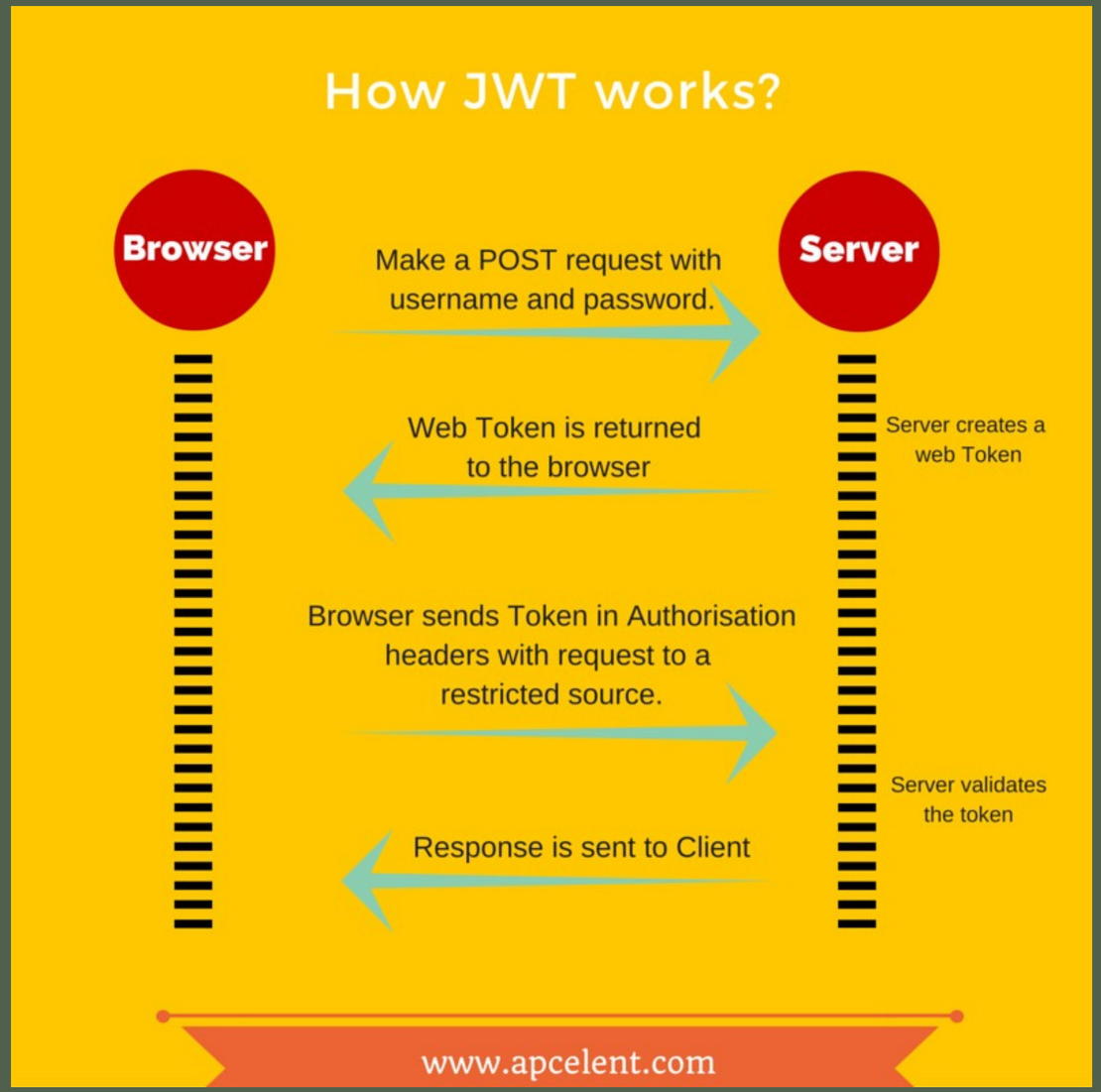
Hardware Requirements

- 1-2mb disk space
- 64bit processor
- laptop/desktop

Software Requirements

- Pycharm
- VM Ware (Linux)
- Node Js
- Vs Code
- Git

OVERALL SYSTEM ARCHITECTURE DESIGN



Encryption Module

The desired information is encoded using a specified algorithm and secret phrase. Special information (defined in the Registered Claim Names section of standard) may be added, like expiration time. Some of this information, such as exp, is handled automatically by specific implementations.
The output is a token of the following type aaaaa.bbbbb.ccccccc.

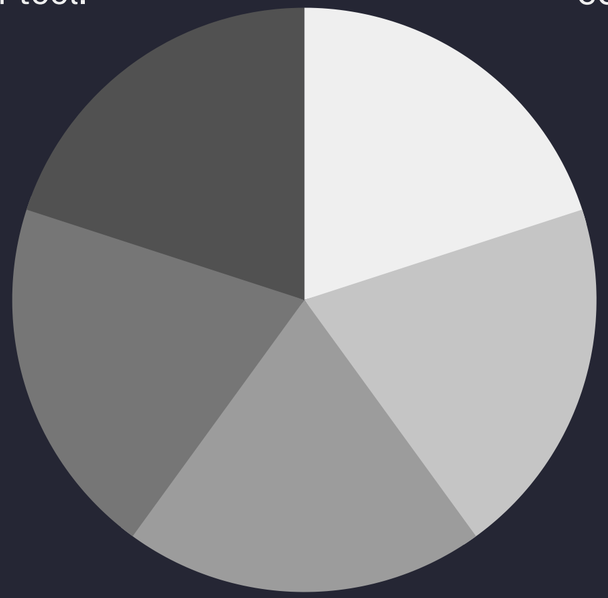
Decryption Module

The generated token is passed to a client. There are a few choices of where to store tokens in the client. For web application it can be a local storage or cookies. There is a discussion on the internet on this topic, such as.
On requests, the token is attached to headers or passed in cookies, depending on chosen storage. The server app is responsible for decoding and validation of data encoded, as well as for resolving permissions.

Project Timeline Chart

WEEK 1
Analyzing and understanding the basic idea of tool.

WEEK 2
Working on the algorithm and code of the tool..



WEEK 5
Check errors and bugs and rectify it.

WEEK 3
Code the different module of tool.

WEEK 4
Combine each module and sub parts of tool in a single environment.

Snapshot of Code

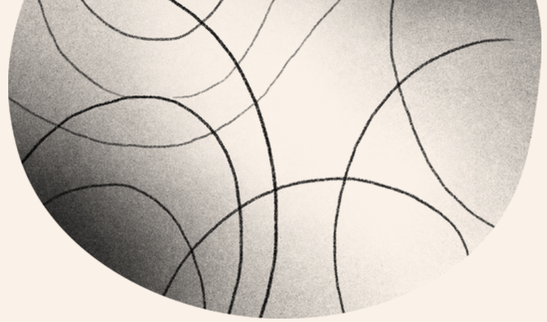
```
#!/usr/bin/env node

"use strict";

const commander = require("commander");
const { version } = require("../package.json");
const { sign, verify, decode, algorithmsRegex } = require("../lib/jwt");
const { error } = console;
const { exit, argv } = process;

commander.version(version).on("command:*", () => {
  error(`Invalid command: ${commander.args.join(" ")}\nSee --help for a list of availab
  exit(1);
});

commander
  .command("sign [payload] [secret]")
  // .alias("create")
```

```
commander
  .command("sign [payload] [secret]")
  // .alias("create")
  .description("Sign a new JWT")
  .option("-a, --algorithm [algorithm]", "Identifies the cryptographic algorithm used t
  .option("-d, --audience [audience]", "Identifies the recipient(s) that the JWT is int
  .option("-e, --expiresIn [time]", "Identifies the expiration time on or after which t
  .option("-h, --header [header]", "Header for JWT.")
  .option("-i, --issuer [issuer]", "Identifies principal that issued the JWT.")
  .option("-k, --keyid [keyid]", "A hint indicating which key was used to secure the JW
  .option("-b, --notBefore [time]", "Identifies the time before which the JWT must not
  .option("-t, --noTimestamp", "The generated JWT will not include an iat.")
  .option("-j, --jwtid [jwtid]", "Case sensitive unique identifier")
  .option("-s, --subject [subject]", "Identifies the subject of the JWT.")
  .option("-p, --passphrase [passphrase]", "The passphrase for your secret key.")
  .option("-n, --noCopy", "Do not copy the token to the clipboard.")
  .action(sign);
```

```
  .command("verify [token] [secret]")
  .description("Verify a JWT")
  .option("-a, --algorithms [algorithms]", "Identifies the cryptographic algorithm use
  .option("-d, --audience [audience]", "Identifies the recipient(s) that the JWT is in
  .option("-t, --clockTimestamp [seconds]", "The time in seconds that should be used a
  .option("-c, --clockTolerance [seconds]", "The number of seconds to tolerate when ch
  .option("-e, --ignoreExpiration", "If true do not validate the expiration of the tok
  .option("-b, --ignoreNotBefore", "If true do not validate the not before of the toke
  .option("-i, --issuer [issuer]", "Identifies principal that issued the JWT.")
  .option("-m, --maxAge [time]", "The maximum allowed age for tokens to still be valid
  .option("-s, --subject [subject]", "Identifies the subject of the JWT.")
  .option("-p, --passphrase [passphrase]", "The passphrase for your secret key.")
  .action(verify);
```

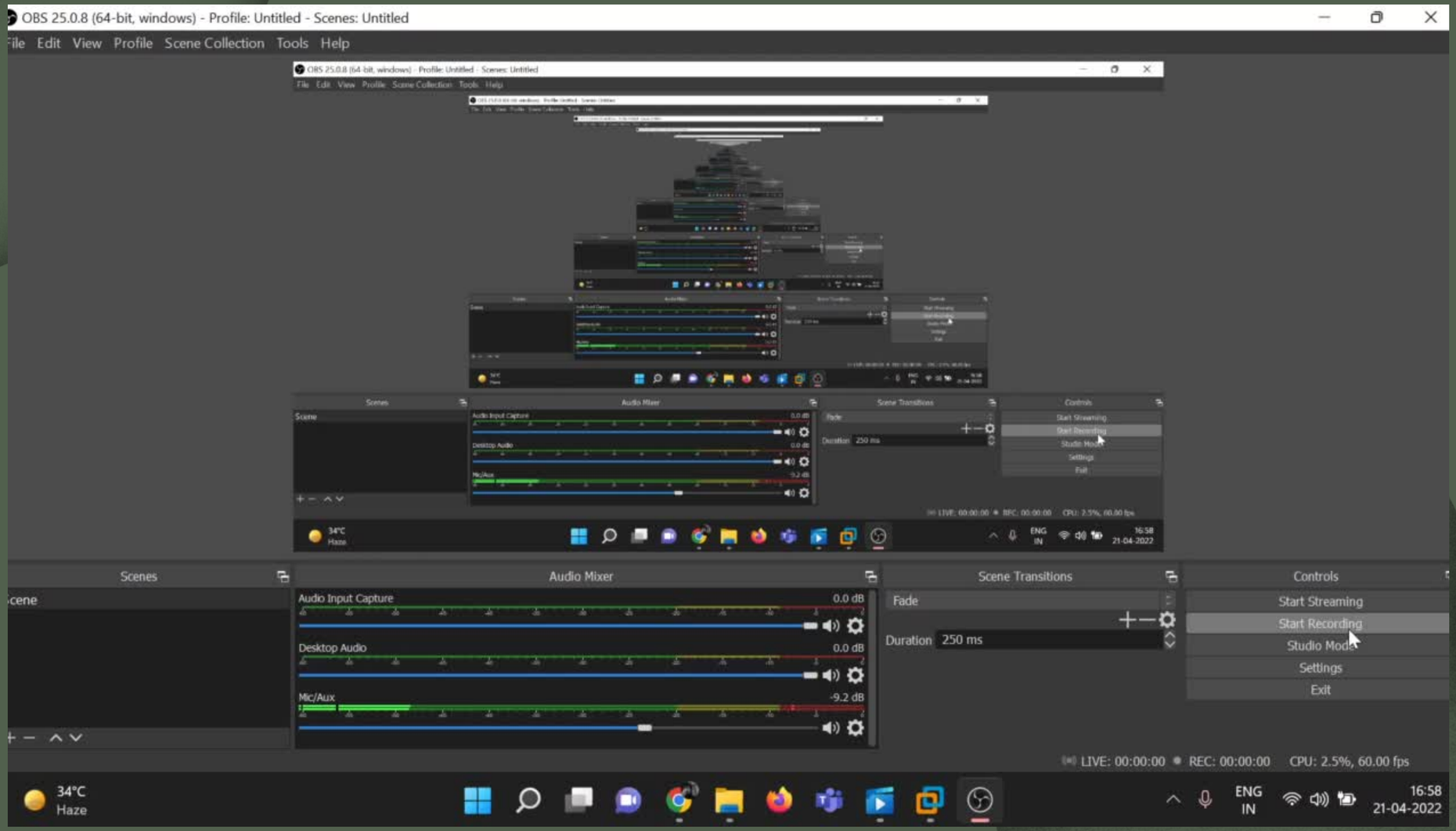
```
commander
  .command("decode [token]")
  .description("Decode a JWT")
  .option("-c, --complete", "Return header, payload, and signature.")
  .option("-n, --noCopy", "Do not copy the payload to the clipboard.")
  .action(decode);

// TODO
// .command('refresh')

// TODO
// commander
//   .command('scope')
//   .description('Scopes of a JWT')
//   .option('-l, --list', 'List scopes of JWT.')
//   .option('-v, --verify [scopes]', 'Scope(s) to verify against JWT.')

commander.parse(argv);

if (!argv.slice(2).length) {
  commander.outputHelp(help => help);
}
```

IMPLEMENTATION

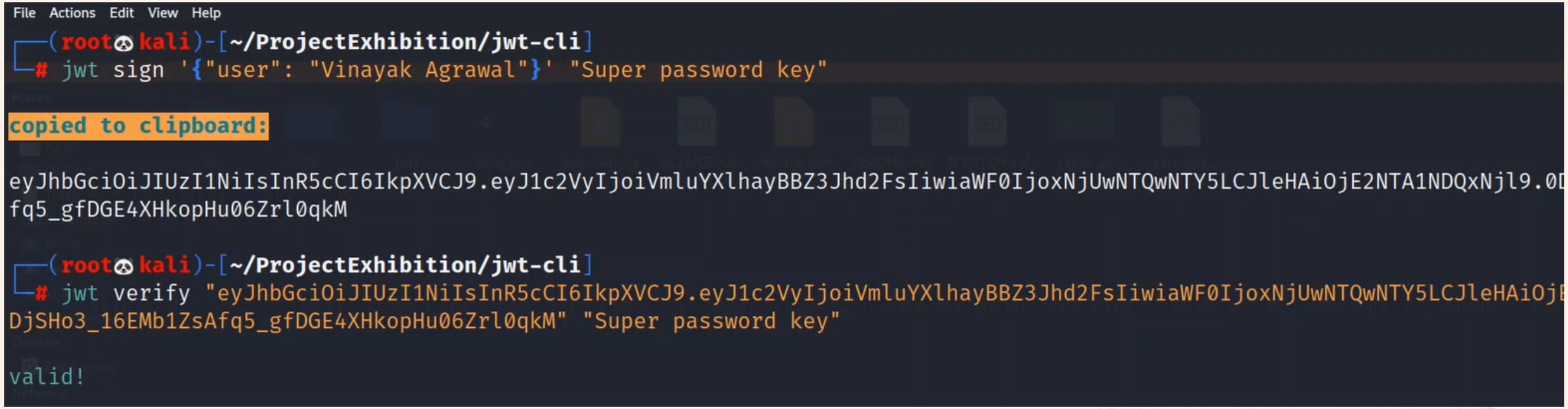


- Run the tool.
- In the login field the user submit the credentials.
- The server generates the verification token and return it to browser.
- Browser will send token with request to a restricted source.
- Server will identify the token whether the public & private key signatures are correct and the will respond to client.

DEMO VIDEO


OBS 25.0.8 (64-bit, windows) - Profile: Untitled - Scenes: Untitled
File Edit View Profile Scene Collection Tools Help
Scenes
Audio Mixer
Scene Transitions
Controls
Audio Input Capture
0.0 dB
Desktop Audio
0.0 dB
Mic/Aux
-9.2 dB
Fade
Duration 250 ms
Start Streaming
Start Recording
Studio Mode
Settings
Exit
LIVE: 00:00:00 REC: 00:00:00 CPU: 2.5%, 60.00 fps
34°C
Haze
ENG
IN
16:58
21-04-2022

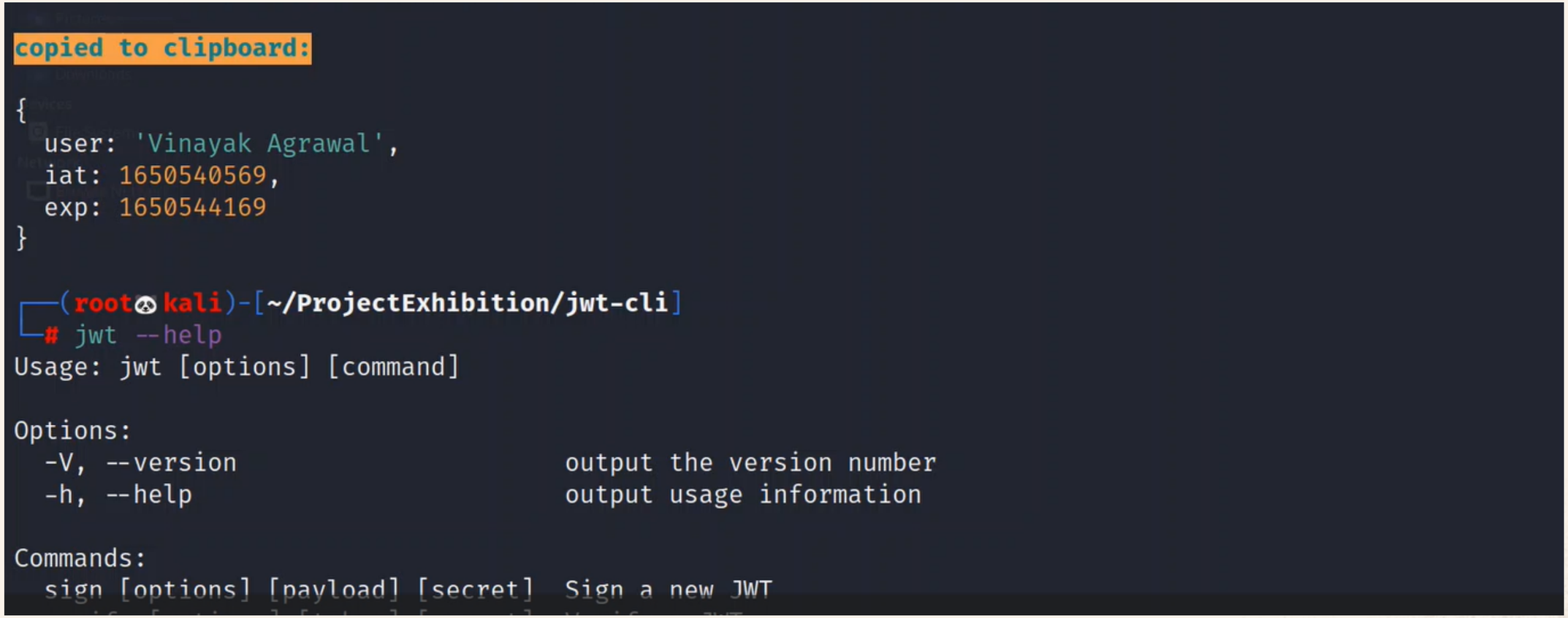
Snap of Project

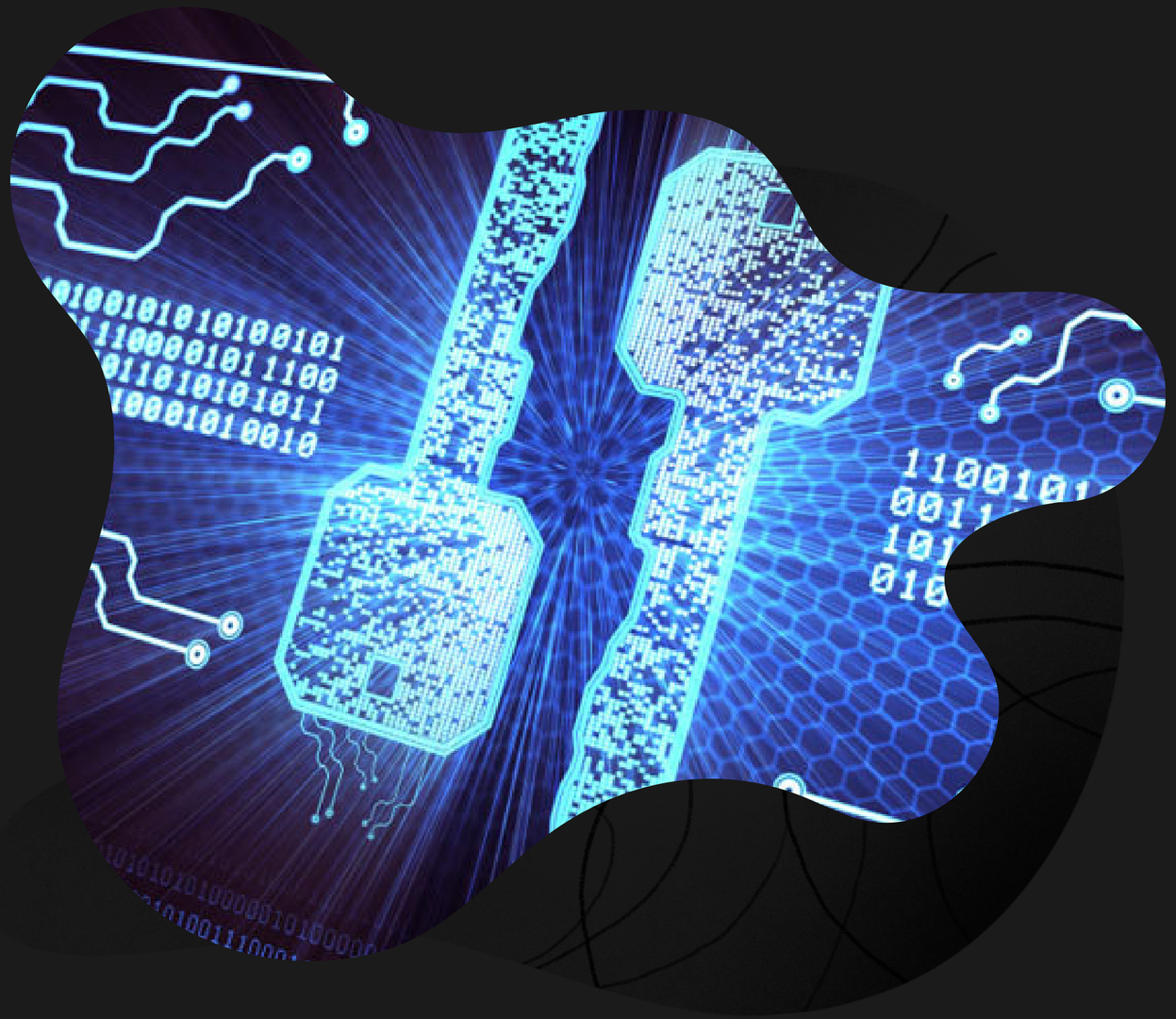


Snap of Project

```
┌──(root🐼kali)-[~/ProjectExhibition/jwt-cli]
└─# jwt verify "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoiVmluYXlhayBBZ3Jhd2FsIiwiaWF0IjoxNjUwNTQwNTY5LCJleHAiOjE2NTA1NDQxNjl
DjSHo3_16EMb1ZsAfq5_gfDGE4XHkopHu06Zrl0qkM" "Super"

 invalid signature


┌──(root🐼kali)-[~/ProjectExhibition/jwt-cli]
└─# jwt decode "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoiVmluYXlhayBBZ3Jhd2FsIiwiaWF0IjoxNjUwNTQwNTY5LCJleHAiOjE2NTA1NDQxNjl
DjSHo3_16EMb1ZsAfq5_gfDGE4XHkopHu06Zrl0qkM"
```

Snap of Project





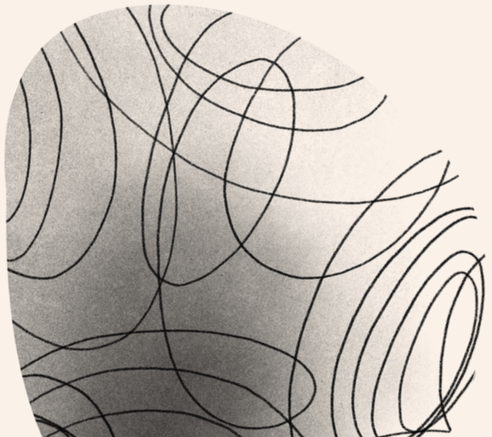
RESULT

The tool gives a command line interface to a cyber security enthusiast or anyone related to the field. It provides a way for securely transmitting information between different parities in the form of JSON object .The sent or received information can be verified because they are digitally signed.

Coniribuiion

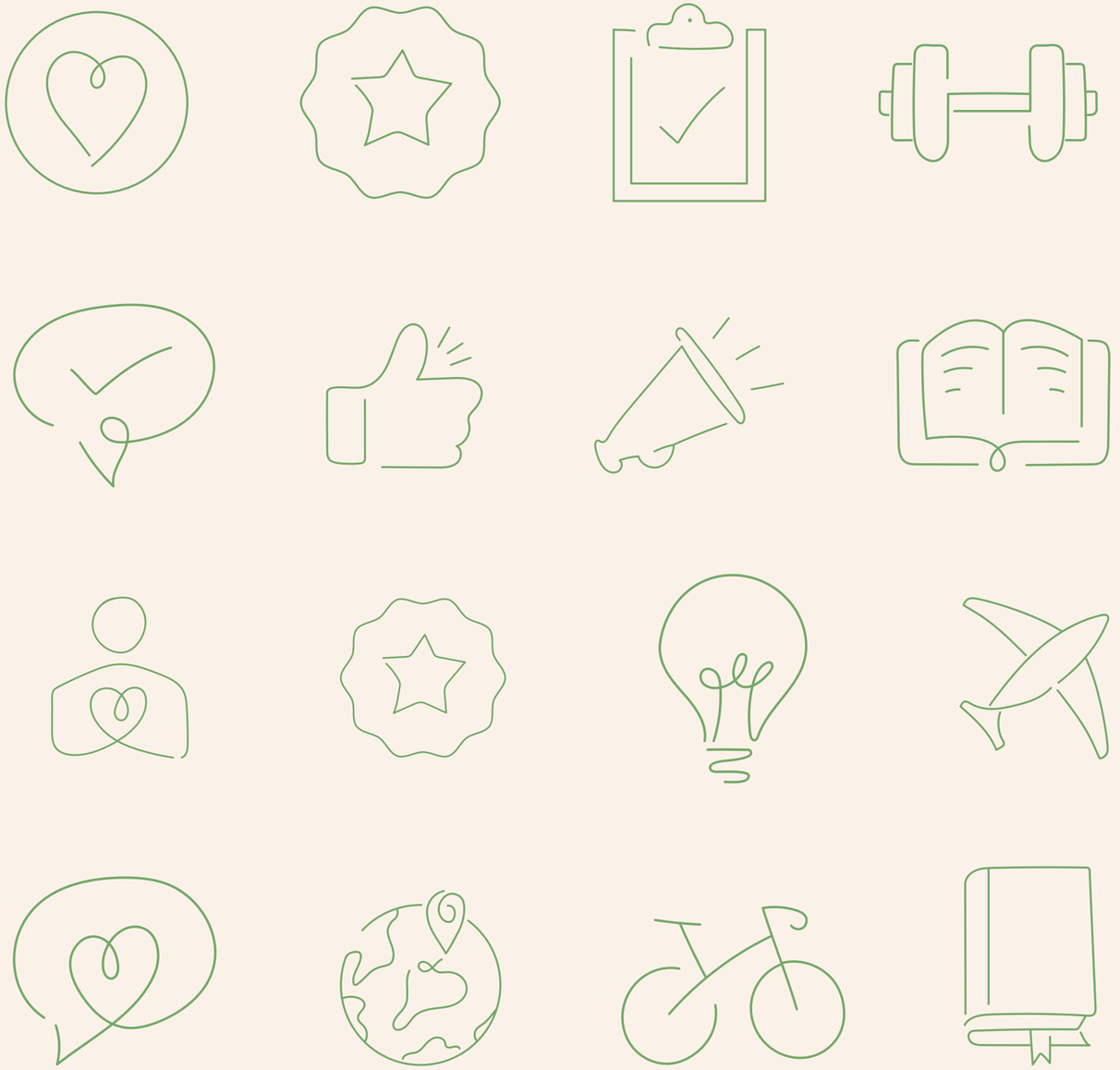
- VINAYAK AGRAWAL- RESEARCH OF JWT CODING PART
- ANURAG SOURAV - RESEARCH AND CODE ANALYSIS & PPT MAKING
- SHAILI GUPTA- ENCRYPTION CODING &RESEARCH
- KHUSHI GARG - ERROR MODIFICATION OUTPUT ANALYSIS

Conclusion



JWT is a great alternative to cookie-based authentication approach. It can have different usages: authentication mechanism, url-safe encoding, securely sharing private data, interoperability, data expiration, etc .JWTs can be used by a server to tell the client app what actions the user is allowed to execute.

References



- Portswigger
- HackerOne(reports)
- Bugcrowd(reports)
- Stackoverflow
- PentesterAcademy

Thank you!