CLI TOOL FOR JSON TOKENS

STUDENT NAMES: ANURAG SOURAV, VINAYAK AGRAWAL, KHUSHI GARG, SHAILI GUPTA
GUIDE: Mr. Vikas Kumar Jain

Project Group ID:- 8

ABSTRACT

Authentication and authorization are used in security, particularly when it comes to getting access to a system .Authentication is the process of verifying a user's identification through the acquisition of credentials and using those credentials to confirm the user's identity.
Authorization is the process of allowing authenticated users access to resources by determining whether they have system access permissions.

JWT are an open industry standard for representing claims between two parties. JWT specifies a compact and self-contained method for communicating information as a JSON object between two parties. Because it is signed, this information can be checked and trusted. JWTs can be signed using a secret an RSA or ECDSA public/private key combination.

Our Command line JWT Tool help to secure endpoints and even authenticate users.

INTRODUCTION

Our tool helps in session management and secure authentication . It is encoded representation of a JSON object. Session IDs are vulnerable to session fixation attacks.
Proper session authentication is required which can be done by JWT tokens.

RESULTS

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.

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.

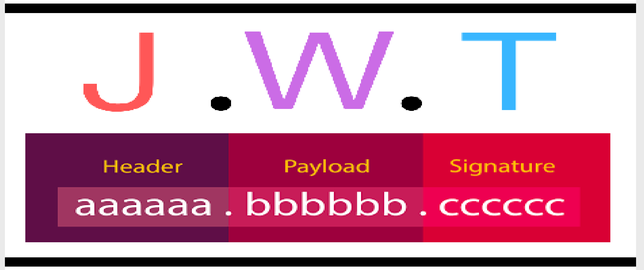
MODULES

- **Encryption Module:**

The desired information is encoded using a specified algorithm and secret phrase. Special information may be added, like expiration time.

- **Decryption Module:**

The generated token is passed to a client. 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.




```
┌──(root㉿kali)-[~/ProjectExhibition/jwt-cli]
└─# jwt sign '{"user": "Vinayak Agrawal"}' "Super password key"

copied to clipboard:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoiVmluYXlhayBBZ3Jhd2FsIiwiaWF0IjoxNjUwNTQwNTY5LC
fq5_gfDGE4XHkopHu06Zrl0qkM

┌──(root㉿kali)-[~/ProjectExhibition/jwt-cli]
└─# jwt verify "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoiVmluYXlhayBBZ3Jhd2FsIiwiaWF0Ij
DjSHo3_16EMb1ZsAfq5_gfDGE4XHkopHu06Zrl0qkM" "Super password key"

valid!

┌──(root㉿kali)-[~/ProjectExhibition/jwt-cli]
```

```
 $$$$$\  $$\       $$\ $$$$$$$$\  $$$$$$$$\                    $$\
 \__$$ | $$ |  $\  $$ |\__$$  __| \__$$  __|                   $$ |
    $$ | $$ | $$$\ $$ |   $$ |       $$ | $$$$$$\   $$$$$$\  $$ |
    $$ | $$ $$ $$\$$ |   $$ |       $$ |$$  __$$\ $$  __$$\ $$ |
$$\ $$ | $$$$  _$$$$ |   $$ |       $$ |$$ /  $$ |$$ /  $$ |$$ |
$$ | $$ | $$$  / \$$$ |   $$ |       $$ |$$ |  $$ |$$ |  $$ |$$ |
\$$$$$$ /|$$ /     \$$ |   $$ |       $$ |\$$$$$$ |\$$$$$$ |$$ |
 \______/ \__/       \__|   \__|       \__| \______/  \______/ \__|
 Version 1.3
```

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.