💚

JWT

Introduction
JWT.io
- Header
- Body
- Key

Common exploits
- Hashing algorithm none
- JWT manipulation
- Token sidejacking
- Weak token secret
- Longlasting tokens

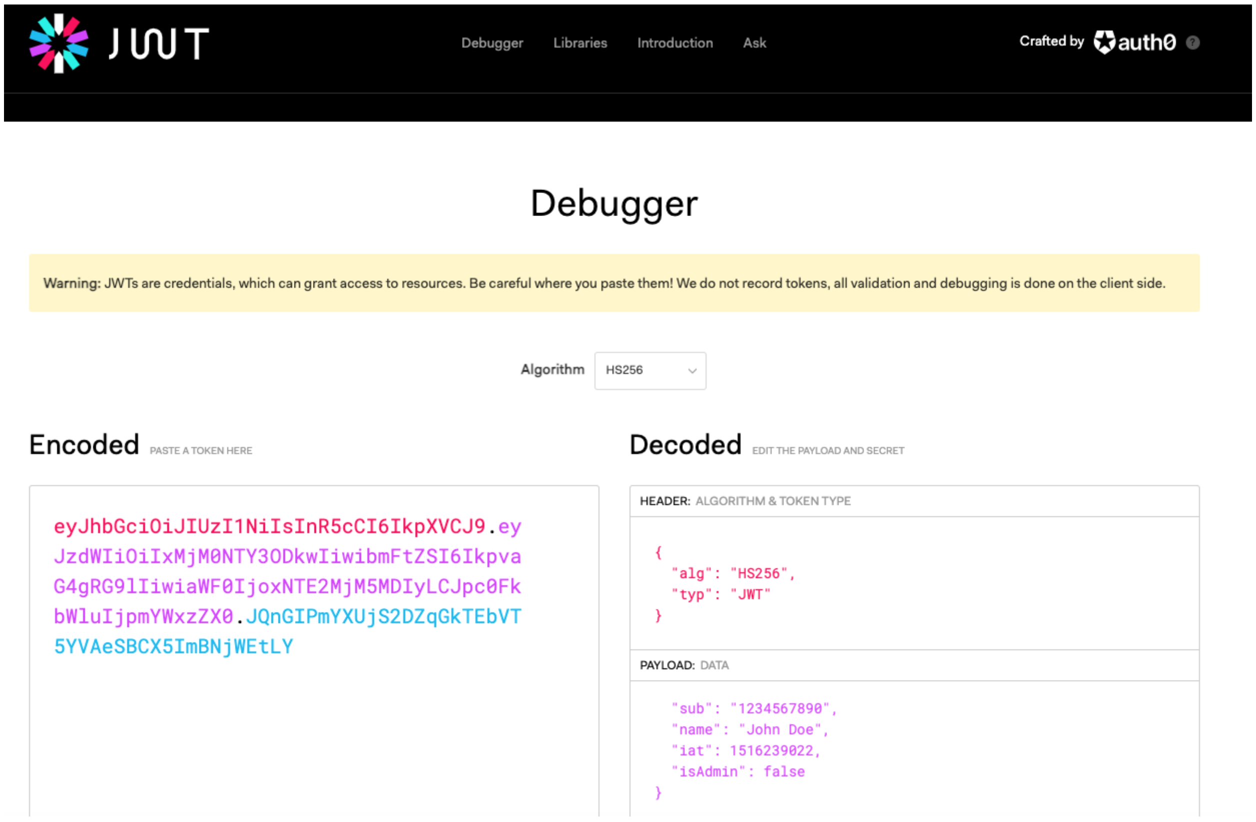
Conclusion

Introduction

JWT or JSON web tokens are widely used these days for authorization purposes so it pays off to learn about what JWT's entail and how we can abuse them. In essence, it all may seem simple but simplicity is often the killer of good judgment and the devil is in the details. Let's explore this wonderful way of authorising users and you will soon see that JWT attacks are an essential tool in any hackers toolbelt.

JWT.io

To analyze, debug and play with JWT tokens in general, we can write our own very simply JWT encoder and decoder in python (https://pyjwt.readthedocs.io/en/stable/) but it's still easier to simply use a WebClient and that is where JWT.io comes into play.



Debugger

Warning: JWTs are credentials, which can grant access to resources. Be careful where you paste them! We do not record tokens, all validation and debugging is done on the client side.

Algorithm HS256

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva
G4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyLCJpc0Fk
bWluIjpmYWxzZX0.JQnGIPmYXUjS2DZqGkTEbVT
5YVAeSBCX5ImBNjWEtLY
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022,
  "isAdmin": false
}
```

A visit to the website shows us an example of JWT. This allows us to see that it's nothing more than a combination of JSON strings in an encoded form.

The structure is always as follows: "header.body.key"

Header

It all starts with the header which is used to describe what encryption algorithm is being used. Several popular options exist and most of them require a key as well.

Body

The body of a JWT usually describes certain properties of a user, for example, the location of a user which might authorize them to access geo-locked content. This can be anything and usually contains some juicy information we might want to change to test the authorization module.

Key

The last part of the JWT is the key and this is the most important part because if you get this wrong, the server will never accept your JWT but if you are able to get the key and encode the value that is equally a very bad problem.

Common exploits

Hashing algorithm none

One very common exploit that is slowly going away thankfully is when the server accepts JWT tokens with the header that indicated the use of no key. This will allow us to easily forge any token we want.

JWT manipulation

Sometimes, user controlled values end up in a JWT token. This should never ever be the case but since some application abuse JWT tokens for authentication rather than authorization, we can still play with the contents of the JWT by changing the username to something that interrupts the current JWT structure and insert our own. This means we might be able to change values and make ourselves admin or even login as other users if the system is built in a bad way.

i.e.

```
{
  "username":"attack",
  "isAdmin":"false"
}

Might turn into
{
  "username":"attack",
  "isAdmin":"false"
}
  "isAdmin":"false"
}
```

And no that is not a type but by inserting the username attack",   "isAdmin":"false"} we might be able to do serious harm.

Token sidejacking

This exploit is basically intercepting the token/key and using it to sign a valid JWT. This can be very basic like a leaked token on the source code of a website.

This can be prevented and we will get more into that into another article.

Weak token secret

Of course just like with any key, this one is vulnerable to being weak and easy to brute force so pick a solid key and make sure it does not leak.

Longlasting tokens

Normally, every token contains an expiration field. Usually it’s called “exp” but it might be called something different. Make sure the server actually disregards any JWTs that are no longer valid.

Conclusion

While the attack surface on JWT is not massive, it does offer some ways we could potentially exploit our target and now you know what to do next time you find the JWT signing key 🙂