Hacker Tools: JWT_Tool – The JSON Web Token Toolkit

BY ANNA HAMMOND · JULY 27, 2021 · LAST UPDATED ON MARCH 6, 2025

When you log in to a website and start surfing, why don't you need to type in your password for every subsequent request? JWT is a very likely reason for that. It allows information transmission and authorization in a simple format. However, sometimes it is implemented incorrectly and that can lead to security vulnerabilities. Luckily, there are tools out there to help us with that.

This week, we're going to check out JWT_Tool and how we can use it to find misconfigurations in JWT implementations. Are you ready to take a deep dive into this amazing tool?

What is JWT?

When looking at HTTP requests, you've most likely seen something like this in the headers:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

This is a JWT token. Every token consists of three parts delimited by a `.`, where every part is base64 encoded data. The first section is the header, which contains claims about the algorithm being used for signing or encrypting and much more. If we decode a header, it may look like this: `{"alg":"HS256","typ":"JWT"}`.

The body, on the other hand, contains the actual data we're trying to pass on to the server. It may contain things like username, permissions, and more. Note that these can be read very easily but can't be modified by the user. This is because the last part of the token contains a signature. This last part is used to ensure that no changes were made to the token. If the server receives a token, it can recalculate the signature and compare it to the one provided, only if they match, will the server know that no modifications occurred and that it can continue processing the request.

What is JWT_Tool?

JWT_Tool is a toolkit for validating, forging, scanning, and tampering with JWT tokens written in Python. We can use it to perform some reconnaissance by viewing the token's claims, run an active scan against the endpoint to try and find misconfigurations, fuzz applications, crack weak secrets, and much more!

In simpler terms, JWT_Tool is the way to go when assessing the security of systems using JWT. Continue reading to find out how to run JWT_Tool and how to find vulnerabilities.

JWT_Tool logo

How to install?

Let's start playing around with this amazing tool. To get a local installation up and running, all you have to do is clone the repository from here and install the Python dependencies using **python3 -m pip install termcolor cprint pycryptodomex requests**

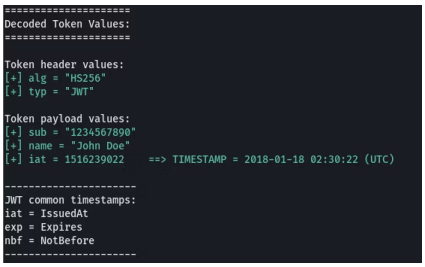
It truly is as easy as that and you've set everything up to get hunting!

Playing around with the tool

Viewing the token

The easiest action to perform is to view the headers and payload values of your token. This can very easily be done through

```
python3 jwt_tool.py <<JWT_TOKEN>>
```



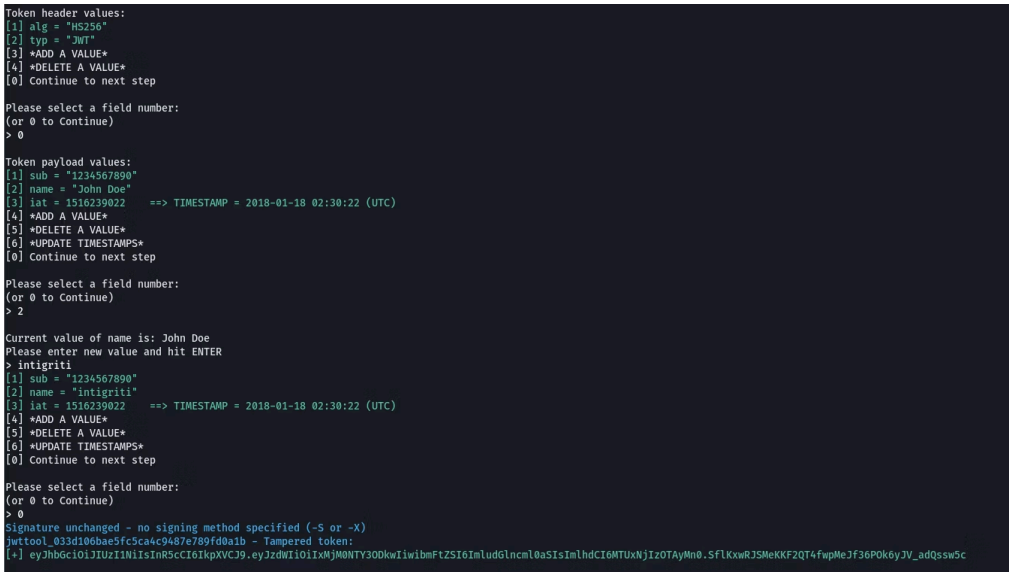
Decoding token

Tampering with the token

Now that we can view the token, we can also start to tamper with it by changing values. The following command allows us to do this.

```
python3 jwt_tool.py <<JWT_TOKEN>> -T
```

Note that in this case, we're not resigning the token. However, use the `-S` option to add a signature if you have the required information available. Using the `-X` option, we can attempt some popular exploits such as injecting or spoofing the JWKS file, setting the algorithm to none, or performing a key confusion attack.



Tampering with a token

Cracking the secret key

If you've identified that the algorithm being used is HMAC, you may have some luck in attempting to crack the secret key. JWT_Tool can do just that using the following command.

```
python3 jwt_tool.py <<JWT_TOKEN>> -C -d <<DICT_FILE>>
```

None algorithm attack – CVE-2015-9235

This attack targets an option in the JWT standard for producing unsigned keys. The output literally omits any signature portion after the second *dot*. Due to weaknesses in some libraries or server configurations, a service may read our tampered request, see that it does not need to be signed, and then just accept it on trust. This can be exploited using JWT_Tool with the `-X a` option.

RS256 to HS256 Key Confusion Attack – CVE-2016-5431

This attack plays around with the fact that some libraries use the same variable name for the *secret* that signs/verifies the HMAC symmetric encryption and the *secret* that contains the Public Key used for verifying an RSA-signed token. By tweaking the algorithm to an HMAC variant (HS256/HS384/HS512) and signing it using the publicly available Public Key we can trick the service into verifying the HMAC token using the hard-coded Public Key in the *secret* variable.

In JWT_Tool this can be very easily tested by providing the token and a public key.

```
python3 jwt_tool.py <<JWT_TOKEN>> -X k -pk <<PUBKEY.PEM>>
```

Conclusion

JWT_Tool is an extremely powerful toolset that will greatly increase your efficiency while searching for vulnerabilities in JWT implementations.

If you would like to recommend a tool for us to cover next week, then be sure to let us know down below. Also be sure to check out all the previous Hacker Tools articles, such as the last one on Aquatone.

Did you know that there is a video accompanying this article? Check out the playlist!