

Jwt tool

Description

jwt_tool est une boîte à outils open source conçue pour l'analyse, la manipulation et la génération de JSON Web Tokens (JWTs) au travers de l'utilisation de la ligne de commande.

Installation

L'installation nécessite d'avoir les outils suivants installés sur son système : **git**, **python3**, **pip3**.

```
git clone https://github.com/ticarpi/jwt_tool
cd jwt_tool
python3 -m pip install -r requirements.txt
```

Cas d'utilisation

- Vérification de l'implémentation correcte des JWT sur une plateforme web ;
- Validation de l'absence de CVE.

Fonctionnalités principales

- Manipulation/génération de tokens ;
- Requêtes HTTP spécifiques ;
- Cassage de clés secrètes.

Manuel des options principales

```
-t TARGETURL, --targeturl TARGETURL : URL vers laquelle envoyer le JWT.
-rc COOKIES, --cookies COOKIES : cookies à envoyer avec la requête HTTP.
-rh HEADERS, --headers HEADERS : headers HTTP à envoyer avec la requête HTTP.
-pd POSTDATA, --postdata POSTDATA : données POST à envoyer avec la requête HTTP.
-M MODE, --mode MODE Scanning mode :
    pb = audit du playbook
    er = fuzz les claims connus pour provoquer des erreurs
    cc = fuzz les claims communs
    at - tous les tests
-X EXPLOIT, --exploit EXPLOIT :
    eXploit known vulnerabilities:
    a = algorithme à none
    n = signature nulle
    b = mot de passe vide
    s = spoofing du JWKS
    k = confusion de clé (spécifier la clé avec -pk)
    i = injection d'un JWKS
-C, --crack : casser la clé pour un token HMAC-SHA (specifier -d/-p/-kf).
-d DICT, --dict DICT : chemin vers le dictionnaire pour le cassage de clé.
-p PASSWORD, --password PASSWORD : mot de passe du token.
```

Exemple d'exploitation ou d'utilisation

Exemple 1 - Lecture de token

La commande suivante permet de lire un token et d'en afficher les informations :

```
python3 jwt_tool.py eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJsb2dpbiI6InRyY2FycGkifQ.bsSwqj2c2uI9n7-ajmi3ixVGhPUiY7j09SUn9dm15Po
```

Nous obtenons le résultat suivant :

```
Original JWT:
=====
Decoded Token Values:
=====
Token header values:
[+] typ = "JWT"
[+] alg = "HS256"
Token payload values:
[+] login = "ticarpi"
-----
JWT common timestamps:
iat = IssuedAt
exp = Expires
nbf = NotBefore
-----
```

Grâce à cela, nous pouvons déterminer l'algorithme utilisé pour la signature du token ainsi que les informations du header.

Exemple 2 - Cassage de clé secrète

La commande suivante permet de casser une clé secrète faible grâce à une attaque par dictionnaire :

```
python3 jwt_tool.py eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJsb2dpbiI6InRyY2FycGkifQ.bsSwqj2c2uI9n7-ajmi3ixVGhPUIY7jO9SUn9dm15Po -C -d /usr/share/wordlists/rockyou.txt
```

Nous obtenons le résultat suivant :

```
Original JWT:
[+] secret is the CORRECT key!
```

La clé utilisée pour signer le token est donc **CORRECT**, nous pouvons maintenant reforcer le token avec des paramètres personnalisés.

Exemple 3 - Test d'une CVE connue

Nous pouvons aussi utiliser **jwttool** pour tester des CVEs connues. Dans cet exemple, nous allons utiliser **jwt_tool** sur la **CVE-2020-28042**. Celle-ci permet de forger des tokens dont la signature est nulle mais qui sont quand même valides. Avec **jwt_tool** nous forgeons un token à signature nulle : `bash` `python3 jwt_tool.py eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJsb2dpbil6InRyY2FycGkifQ.bsSwqj2c2uI9n7-ajmi3ixVGhPUIY7jO9SUn9dm15Po -X n` Nous obtenons le résultat suivant : `jwttool_ed591ad5ea32583e84648d60a95626f5 - EXPLOIT: null signature (This will only be valid on unpatched implementations of JWT.)` Si le système que nous essayons de tester est vulnérable à la CVE-2020-28042, c'est à dire qu'il accepte le token que nous venons de forger, alors tout token forgé avec une signature nulle sera valide. ===== Retour fiches outils ===== * [Cyber fiches outils](#)

From: / - Les cours du BTS SIO

Permanent link: /doku.php/cyber/outils/jwt-tool?rev=1750434056

Last update: 2025/06/20 17:40

