Attacking JSON Web Tokens

When "User Input" Doesn't Look Like User Input.

...an excerpt from 4-day course

Modern Webapp Pentesting

First run: July 13-16
Cost: $395
Limit: 30 students

https://wildwesthackinfest.com/online-training/modern-webapp-pentesting/

JSON Web Tokens Are...

These things:

```
eyJ0eXAiOiJKV1QiLA0KICJhbGciOiJIUzI1NiJ9.eyJpc3M
iOiJqb2UiLA0KICJleHAiOjEzMDA4MTkzODAsDQogImh0dHA
6Ly9leGFtcGxlLmNvbS9pc19yb290Ijp0cnVlfQ.dBjftJeZ
4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk
```

JSON Web Tokens Are...

These things...

```
GET /rest/products/search?q= HTTP/1.1
Host: localhost:3000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:76.0) Gecko/20100101 Firefox/76.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm
5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxN
mYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIwLjAuMC4w
IiwicHJvZmlsZUltYWdlIjoiYXNzZXRzL3B1YmxpYy9pbWFnZXMvdXBsb2Fkcy9kZWZhdWx0LnN2ZyIsInRvdHBTZWN
yZXQiOiIiLCJpc0FjdGl2ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiMjAyMC0wNi0wOCAyMjo1NTowNy4zMzMgKzAwOjAwIi
widXBkYXRlZEF0IjoiMjAyMC0wNi0wOCAyMjo1NTowNy4zMzMgKzAwOjAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF0I
joxNTkxNjU3MTIxLCJleHAiOjE1OTE2NzUxMjF9.bHYionAdmkw2Nr9lwNFLW2s0zY9-THTNSA0mMZ9UOz03CzUI4rq
x561_GQL1Tb1MnZPcnMxqy8DvUmrO0UuThOI9PxYaEZNMlX-1R3wFmWqTSXgspc0X5WEXswfUWP0J9BQlfTAZp4e4Du
c5Ug-4T3ABK3QVf8WyZ9KzMpVdl_Y
Connection: close
```

Not All Input Looks Like Input

JSON Web Tokens Are...

"...a compact, **URL-safe** means of representing **claims** to be transferred between two parties.

The **claims** in a JWT are **encoded as a JSON object** that is used as the payload of a JSON Web Signature (JWS) structure or as the plaintext of a JSON Web Encryption (JWE) structure, enabling the claims to be **digitally signed** or integrity protected with a Message Authentication Code (MAC) and/or encrypted."

-- "Abstract" - RFC 7519 - May, 2015

JSON Web Tokens Contain...

JOSE Header:

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

Payload:

```
{"iss":"joe",
"exp":1300819380,
"role":"customer"}
```

JOSE:
J*SON*
O*bject ...*
S*igning and*
E*ncryption*

Payload:
"the claims"

JSON Web Tokens Are...

Base64url encoded, concatenated, signed...

```
base64url(header)
    .
        base64url(payload)
            .
                base64url(signature)
```

Base64 vs Base64URL Encoding

To convert a Base64 string to a Base64URL string...

+ becomes **-**

/ becomes **_**

= becomes nothing (i.e. padding is removed)

Clues

Base64 of JWT often begins with `eyJ0eXAi` or `eyJhbGci`

```
$ echo -n 'eyJ0eXAi' | base64 -d
{"typ"
$ echo -n '{"alg"' | base64
eyJhbGci
```

...and a dot in the first 40 - 60 characters or so...

Aside... Why Base64?

NOT to protect information from malice.

Base64 does not do that.

Base64 ONLY makes them "URL-Safe".

Three Parts: Header, Payload, Signature

Header Says Two Main Things:

1. This is a JWT
2. The signature was computed with *this* algorithm.

Three Parts: Header, Payload, Signature

Payload may say a few standard things...

iss: issuer sub: subject

iat: issued at exp: expires at

nbf: "not before" (start date)

Three Parts: Header, Payload, Signature

Payload may say ... literally anything else

username?

email address?

role?

permissions?

password?

Three Parts: Header, Payload, Signature

Signature is ... a digital signature

(...of the encoded header and payload, using the algorithm named in the header)

Common Use: Federated Authentication and Authorization

Often: HTTP "Authorization" Header

Remember this?

```
GET /rest/products/search?q= HTTP/1.1
Host: localhost:3000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:76.0) Gecko/20100101 Firefox/76.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm
5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxN
mYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIwLjAuMC4w
IiwicHJvZmlsZUltYWdlIjoiYXNzZXRzL3B1YmxpYy9pbWFnZXMvdXBsb2Fkcy9kZWZhdWx0LnN2ZyIsInRvdHBTZWN
yZXQiOiIiLCJpc0FjdGl2ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiMjAyMC0wNi0wOCAyMjo1NTowNy4zMzMgKzAwOjAwIi
widXBkYXRlZEF0IjoiMjAyMC0wNi0wOCAyMjo1NTowNy4zMzMgKzAwOjAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF0I
joxNTkxNjU3MTIxLCJleHAiOjE1OTE2NzUxMjF9.bHYionAdmkw2Nr9lwNFLW2s0zY9-THTNSA0mMZ9UOz03CzUI4rq
x561_GQL1Tb1MnZPcnMxqy8DvUmrO0UuThOI9PxYaEZNMlX-1R3wFmWqTSXgspc0X5WEXswfUWP0J9BQlfTAZp4e4Du
c5Ug-4T3ABK3QVf8WyZ9KzMpVdl_Y
Connection: close
```

Aside... one "obvious" reason for base64

JSON can have newlines.

HTTP headers can't.

RFCs of Interest

7519: JWT (...Tokens)

7518: JWA (...Algorithms)

7515: JWS (...Signatures)

7516: JWE (...Encryption)

Most JWTs are JWSes...

Encoded, not *encrypted*.

...therefore readable. *Always*.

Most JWTs are JWSes...

A good signature allows tampering to be detected.

Signing algorithm is part of the header

...therefore attacker-controllable. ...*Always*.

Most JWTs are JWSes...

So...

Servers need to be careful.

More "bouncer" than "concierge"

1. Do Not Trust User Input

2. Everything is User Input

How Many Issues in the OWASP Top Ten?

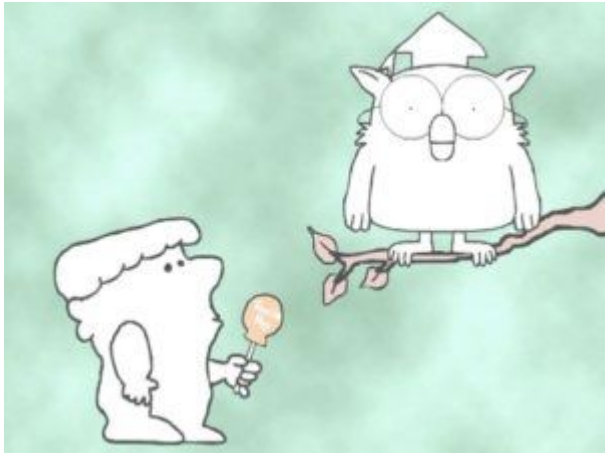
1: Unexpected User Input

2: Unexpected User Input

3: Sensitive Data Exposure (leaks)

4-9: Unexpected User Input

10. Insufficient Logging & Monitoring



Signature Algorithms (RFC 7518)...

3.1. "alg" (Algorithm) Header Parameter Values for JWS

"alg" Value	Digital Signature or MAC Algorithm	Implementation Req'ts
HS256	HMAC using SHA-256	Required
RS256	RSASSA-PKCS1-v1_5 using SHA-256	Recommended
none	No digital signature or MAC performed	Optional

That "none" option looks dangerous...

"Resistance to tampering"

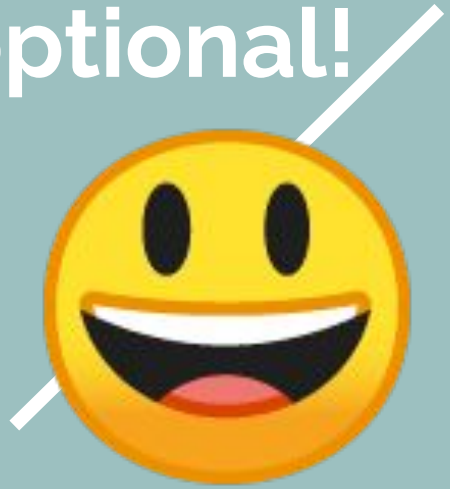
3.6. Using the Algorithm "none"

JWSs MAY also be created that do not provide integrity protection.

Such a JWS is called an Unsecured JWS. **An Unsecured JWS uses the "alg" value "none"** and is formatted identically to other JWSs, but MUST use the **empty octet sequence as its JWS Signature** value. Recipients MUST verify that the JWS Signature value is the empty octet sequence.

If it's empty...
...how do you know it's an octet sequence?

RFC 7518 (J.W. Algorithms)
The "none" algorithm is optional!



Signature Algorithms (RFC 7519)

8. Implementation Requirements

This section defines which algorithms and features of this specification are mandatory to implement. . . .

Of the signature and MAC algorithms specified in JSON Web Algorithms [JWA], only HMAC SHA-256 ("HS256") and **"none" MUST be implemented** by conforming JWT implementations.

RFC 7519 (JWT):
The "none" algorithm is a "MUST"

JWT's Stance on Privacy

12. Privacy Considerations

A JWT may contain privacy-sensitive information. When this is the case, measures MUST be taken to prevent disclosure of this information to unintended parties.

... [*Encrypt the JWT and/or use TLS*] ...

Omitting privacy-sensitive information from a JWT is the simplest way of minimizing privacy issues.

https://tools.ietf.org/html/rfc7519#section-12

Encrypt the JWT and/or use TLS

JWS's Stance on Security

(A JWS is the kind of JWT you normally see: one whose claims are not encrypted)

Keep it secret

Keep it safe

Cryptography Is Hard

...leaving us with...

No privacy protections in a JWT (*JWS*)

The "none" algorithm disables signing completely

The "none" algorithm is ... required? optional? ...both?

Attacks, then:

1. Information disclosure (just decode the payload)
2. Potential for forgery (if the "none" algorithm is supported)
3. Cracking (guess the "secret" if 1 & 2 don't work)

On Cracking...

- HMAC only as secure as the secret.
- JWT is self-contained.
- Sample code uses bad examples.
- Guess all day long on your own system.

https://github.com/auth0/node-jsonwebtoken

Synchronous Sign with default (HMAC SHA256)

```
var jwt = require('jsonwebtoken');
var token = jwt.sign({ foo: 'bar' }, 'shhhhh');
```

Where To Practice?

I Love Juice Shop. And OWASP. And bkimminich.

Working from home

Björn Kimminich
bkimminich

Follow

Product Group Lead Architecture Governance + Application Security **@kuehne-nagel**, Project Leader **@OWASP** Juice Shop, IT Security Lecturer **@Nordakademie**.

@kuehne-nagel
Hamburg, Germany
github.com@kimminich.de
http://kimminich.de

Overview | Repositories 83 | Projects 0 | Stars 1.1k | Followers 631 | Following 216

Pinned

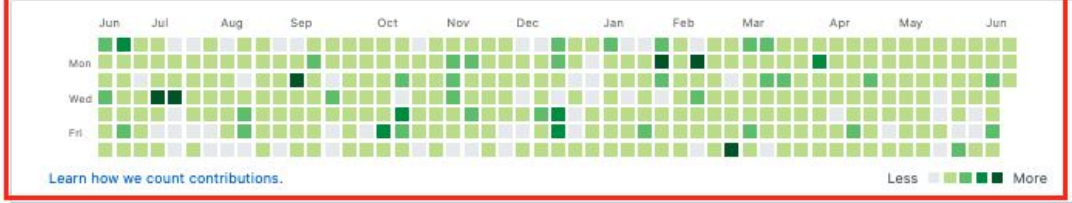
juice-shop
OWASP Juice Shop: Probably the most modern and sophisticated insecure web application
JavaScript ☆ 3.4k ⑂ 2.2k

juice-shop-ctf
Capture-the-Flag (CTF) environment setup tools for OWASP Juice Shop
JavaScript ☆ 188 ⑂ 48

pwning-juice-shop
GitBook markdown content for the eBook "Pwning OWASP Juice Shop"
☆ 103 ⑂ 68

it-security-lecture
University lecture on "IT Security" as Open Educational Resources material
☆ 80 ⑂ 50

5,461 contributions in the last year



...Digi.Ninja, too!

https://digi.ninja/

Authentication Labs!

JWT included!

Decoding the Payload (information disclosure)

Juice Shop JWT!

Prerequisites:

1. Register any account
2. Notice: "admin@juice-sh.op" is an account (*via review on Apple Juice product*)
3. Notice this exists: `http://localhost:3000/rest/user/whoami`
4. Notice there's a JWT involved in that "whoami" thing.

User Secrets in Juice Shop

0. Burp Suite
1. Log in as normal user
2. Send "whoami" request to Repeater & re-send it
3. Trim out extra junk to simplify (which JWT is the important one?)
4. Decode: Look at the payload - anything interesting?

Snooping (stealing poorly-protected secrets)

- Burp Suite Decoder
  - ... gibberish isn't always gibberish
- CyberChef
  - ... what you see is NOT all there is
- Command Line
  - ... "base64url" is not the same as "base64"

Using the "none" algorithm (plain old forgery)

Forgery For Non-Forgers

- Burp Suite Intruder
  - ...with *Advanced Payload Processing* (tm)
- Burp Extension: JOSEPH
  - ...same, but purpose-built
- Command Line
  - Remember it's not *regular* Base64
- CyberChef

Juice Shop Walk-Through

3. Trim out extra junk to simplify (which JWT is the important one?)

4. Decode: Look at the payload - anything interesting?

5. Test for algorithm type of "none" (via Intruder and payload rules)

6. Look at JWT payload to see what might be profitably messed with...

"There is never enough time.
Thank you for yours."

-- Dan Geer

For Further Study

For Further Study

- Discussion of JWT Attacks:
  - https://medium.com/swlh/hacking-json-web-tokens-jwts-9122efe91e4a
- Digi.Ninja's AuthLab:
  - https://authlab.digi.ninja/
- JWT "Best Current Practices" (Feb, 2020)
  - https://tools.ietf.org/html/rfc8725
- Auth0 Information and Tools
  - https://jwt.io/
- "JSON Web Tokens Suck" (Randall Degges)
  - https://www.youtube.com/watch?v=pYeekwv3vC4

For Further Study

- https://owasp.org/www-project-juice-shop/
- https://tools.ietf.org/html/rfc7515
- https://tools.ietf.org/html/rfc7518
- https://tools.ietf.org/html/rfc7519
- https://www.rfc-editor.org/errata_search.php?rfc=7519
- https://www.rdegges.com/2018/please-stop-using-local-storage/
- https://npm.runkit.com/jsonwebtoken

WILD WEST HACKIN' FEST

www.wildwesthackinfest.com

Watch Past WWHF Talks on YouTube

SERVICES

bhis.co

BLACK HILLS
Information Security

Backdoors &Breaches