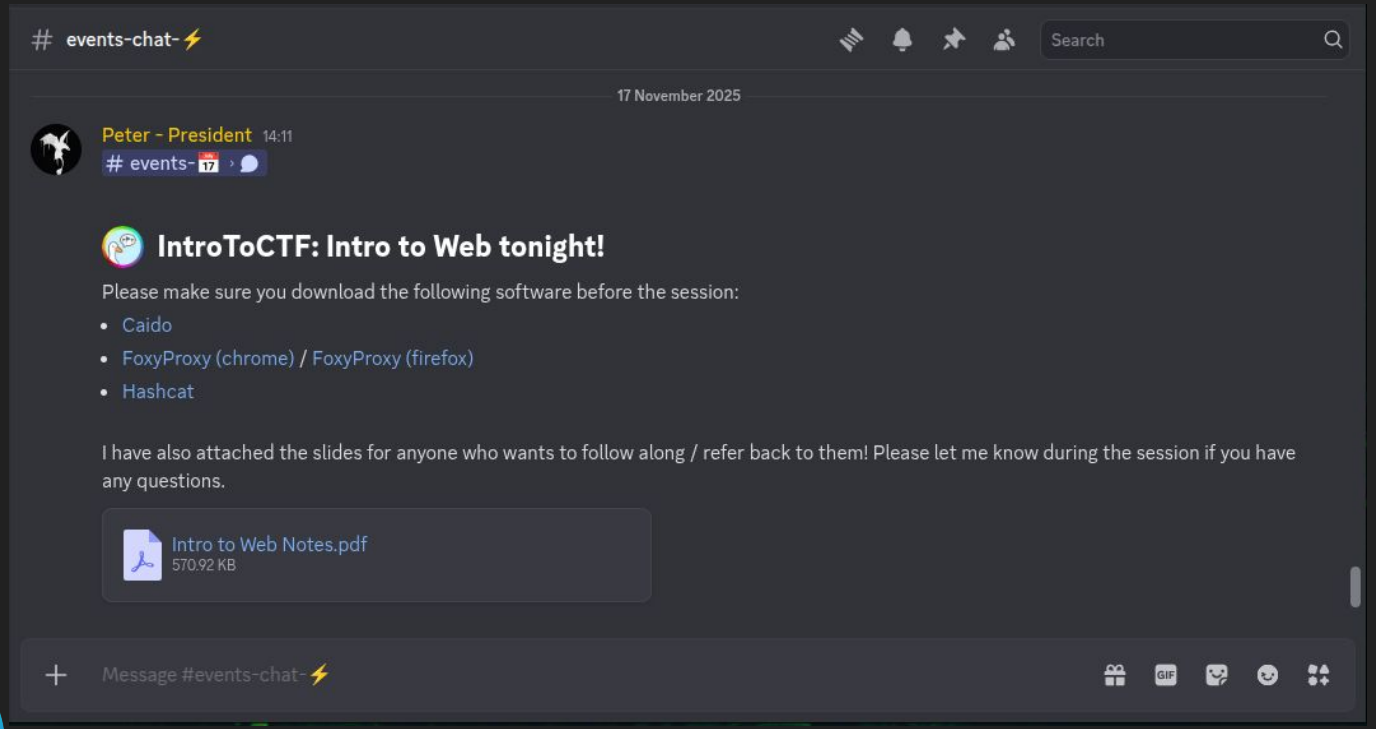
IntroToCTF:

Web Exploitation


events-chat-⚡
Search
17 November 2025
Peter - President 14:11
events-
IntroToCTF: Intro to Web tonight!
Please make sure you download the following software before the session:
Caido
FoxyProxy (chrome) / FoxyProxy (firefox)
Hashcat
I have also attached the slides for anyone who wants to follow along / refer back to them! Please let me know during the session if you have any questions.
Intro to Web Notes.pdf
570.92 KB
Message #events-chat-⚡
GIF

What is ***Web***?

Web exploitation is the process of using misconfigurations in websites to gain unintended permissions to the underlying server.

For example:

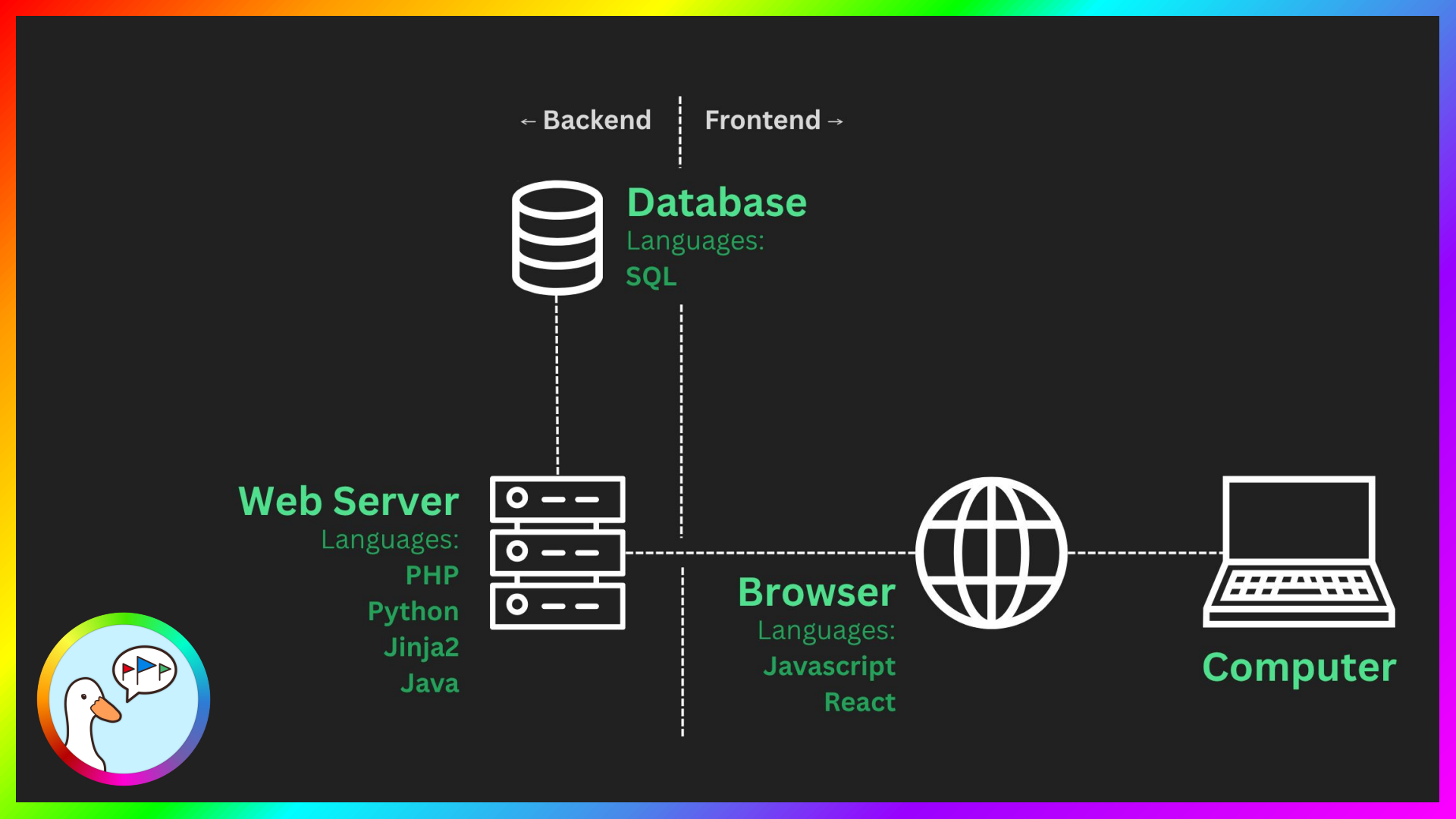
- Running arbitrary commands against a database using SQL.
- Running system commands on a system hosting a web server.
- Reading the contents of files on the system hosting a web server.
- Unauthorised access to subsections of websites by cookie manipulation.

Notes

If you didn't manage to download the tools from Discord, they are also here:

- **Caido** (caido.io/)
- **FoxyProxy** (chromewebstore.google.com/, addons.mozilla.org/)
- **Hashcat** (hashcat.net/hashcat/)

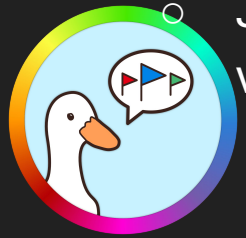
Today, we will run the session as three sets of: mini-lecture, lab. Please ask any questions throughout, since there is no specific time for questions.


← Backend
Frontend →
Database
Languages:
SQL
Web Server
Languages:
PHP
Python
Jinja2
Java
Browser
Languages:
Javascript
React
Computer

Server-Side Template Injection (SSTI)

- Some websites render dynamic content through templates.
- Templating is when the backend directly puts the content of variables into the website code it sends to your computer.
- It can also pass simple code to the templating language, such as 7 * 7, which will render as 49.

- A very common templating language is **Jinja2**, which is used by Flask apps.
  - Jinja2 uses {{ ... }} to denote where the template should calculate the value.



Netkit 2

Sign up for our mailing list, and we'll ensure that
that you hear the latest and greatest news about Netkit 2.

{{ 7 * 7 }}

Sign Up

Thank you for signing up!

We'll send an email to 49 when we're ready to launch.

Server-Side Template Injection (SSTI)

- Once you've found an input (textbox etc.) vulnerable to SSTI, you can execute arbitrary code on the web server.
- Assuming the web server is running Python:
  - Python executes system commands using *import os; os.system("ls")*
  - However, Jinja2 doesn't support direct Python execution.

{{ request.application.__globals__.__builtins__.__import__('os').popen('id').read() }}

Python application

Access global variables / libraries

Import default library *'os'*

Run '*id*' command

Read command output

https://onsecurity.io/article/server-side-template-injection-with-jinja2/

Netkit 2

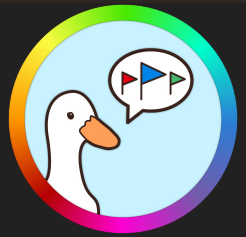
Sign up for our mailing list, and we'll ensure that
that you hear the latest and greatest news about Netkit 2.

{{ _.__import__('os').popen('id').read() }}

Sign Up

Thank you for signing up!

We'll send an email t uid=33(www-data) gid=33(www-data) groups=33(www-data)



uid=33(www-data) gid=33(www-data) groups=33(www-data)

SSTI1 (play.picoctf.org)

Netkit 2 Launch (gym.lilypadd.com)

SSTI2 (play.picoctf.org)

Spookifier (app.hackthebox.com)

Request Manipulation

- When your browser accesses a website, a HTTP request must be sent to retrieve the code for the website.

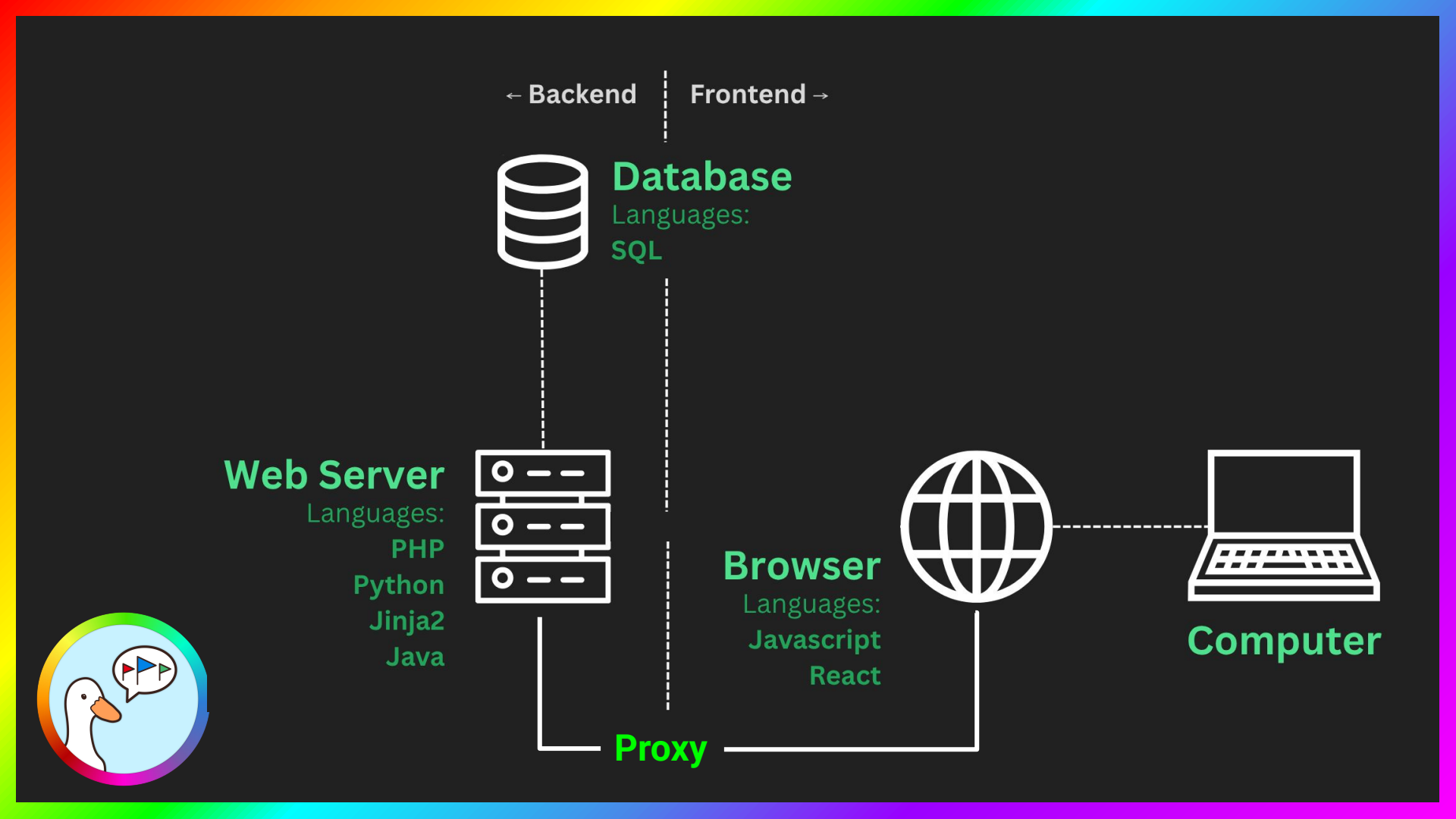
GET	POST	PUT	HEAD
Retrieve data from server	Send data to the server	Modify the data on the server	A GET request, but without a response body

- You can send custom requests using command-line tools, such as cURL, or modify live requests using tools like Burp Suite or Caido.

```
GET /tags/ref_httpmethods.asp HTTP/2
Host: www.w3schools.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64;
rv:140.0) Gecko/20100101 Firefox/140.0
Accept:
text/html,application/xhtml+xml,application/xm
l;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate, br, zstd
Referer: https://www.google.com/
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
Sec-Fetch-User: ?1
Priority: u=0, i
```

```
POST /$rpc/google.internal.onegoogle.asyncdata
HTTP/2
Host: ogads-pa.clients6.google.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64;
rv:140.0) Gecko/20100101 Firefox/140.0
Accept: */*
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate, br, zstd
Referer: https://www.google.com/
Content-Type: application/json+protobuf
X-User-Agent: grpc-web-javascript/0.1
Content-Length: 86
Origin: https://www.google.com
Connection: keep-alive
Cookie:
auth=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJh
ZG1pbiI6dHJ1ZX0.ZcCK4LWJ28-R1wbOgrgVg-kzAcjDE6
jBD_dn2nFa4fs
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors

id=413&input=hello%20world
```


← Backend
Frontend →
Database
Languages:
SQL
Web Server
Languages:
PHP
Python
Jinja2
Java
Browser
Languages:
Javascript
React
Computer
Proxy

Request Manipulation

- By intercepting these requests, you can view them and/or modify them.

- Maybe an application "*requires*" a specific browser? **User-Agent.**
- Maybe you need to have been redirected from somewhere? **Referer.**
- Maybe you need to edit a hidden input field? **POST variables.**
- Maybe you need different authorisation? **Authorization** or **Cookie.**

- You can also view more information about a response than a website will normally tell you - exact status code, error message etc.

Caido demo w/ **FoxyProxy**

GET aHEAD (play.picoctf.org)

Who are you? (play.picoctf.org)

Request Rumble (gym.lilypadd.com)

Cookies (play.picoctf.org)

Cookie Manipulation

- *Cookies* are small amounts of data stored locally by websites on the user's machine.
- They are very commonly used for authentication.
- However, sometimes they may contain other data, such as XML.

- Cookies are usually encoded (e.g. base64) or encrypted (or at least signed!)
- You can see them by opening `Developer Tools > Storage > Cookies`
  - You can also edit them in this interface.

Base64

- A very common encoding for cookie values.
- Used to ensure cookies containing arbitrary data are in an ASCII-string, safe format.

- Uses only the characters A-Z, a-z, 0-9, +, / to form the string.
- May contain ending padding in the form of a single, or double, equals sign.

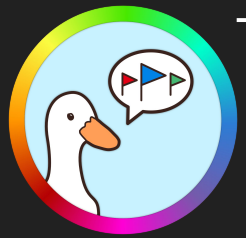
For example: `d2h5IGRpZCB5b3UgYm90aGVyCg==`

Cookie Types

- A lot of cookies will be readable once decoded from base64 - but you may find some that aren't fully.

`eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJhZG1pbiI6dHJ1ZX0.ZcCK4LWJ28-R1wbOgrgVg-kzAcjDE6jBD_dn2nFa4fs:`

`{"typ":"JWT","alg":"HS256"} {"admin":true} p"-bvu`Lr00CV~`



These are called JWTs (JSON Web Tokens).

JSON Web Tokens

- A subsection of cookies.
- Split into three sections: header, payload, and signature.
- They are split up with a "." (period)

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJhZG1pbiI6dHJ1ZX0.ZcCK4LWJ28-R1wbOgrgVg-kzAcjDE6jBD_dn2nFa4fs
```

```
{"typ":"JWT","alg":"HS256"} {"admin":true} p"-bvu`Lr00CV~
```

JSON Web Tokens

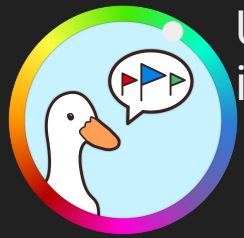
Header: *{"typ":"JWT","alg":"HS256"}*

- Must contain the type of cookie i.e. *JWT*
- Also the signing algorithm: *HS256* (*HMAC-SHA256*) used to authenticate the payload

Payload: *{"admin":true}*

- Contains all the information you want to store in the cookie

Signature: p"-bvu`Lr00CV~



Unintelligible data that verifies the authenticity of the payload by signing it with some secret key.

JWT Vulnerabilities

```
{"typ":"JWT", "alg":"none"}
```

- If the server doesn't have the signing algorithm hardcoded, this can be used to bypass any signature verification. JWTs in this form do not have a third block, but still end with a period.

Weak signing Key

- If the secret key used to sign the JWT is weak, it may be vulnerable to a wordlist/bruteforce attack.

```
hashcat -a 0 -m 16500 jwt.txt rockyou.txt
```

Bruteforce/Wordlist attacks

- A wordlist is a list of commonly-used passwords.
- A wordlist attack uses this list, with a known username (if applicable), to try to authenticate.

- In our case, a signature will be created based on the data and the secret key (password) from the wordlist - and maybe it'll match the signature we know!

JAuth (play.picoctf.org)

Join the Wild Hunt (gym.lilypadd.com)

JaWT Scratchpad (play.picoctf.org)