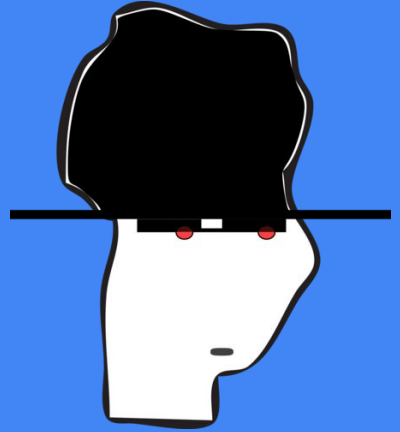
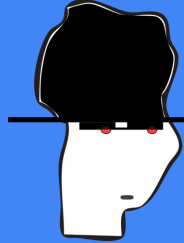


# Seguridad Web

Seguridad Ofensiva





# Seguridad Web

- Tecnologías web
  - Protocolo
  - Server-side
  - Client-side
- Protecciones
  - Controles del lado del cliente
  - Autenticación
  - Sesiones
- Vulnerabilidades
  - Inyecciones (SQL, NOSQL, ...)
  - XSS
  - XXE's
  - CSRF
  - SSRF
  - Deserialización insegura
  - Exposición de datos
  - L/R File Inclusion

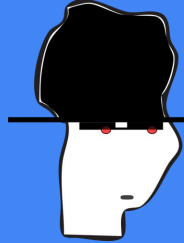
# TOP 10 - OWASP




OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	⊗	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	⊗	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]



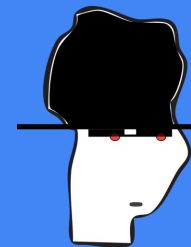
# Tecnologías web



# Tecnologías Web

- Protocolos
  - HTTP/s    WS    QUIC
  - SPDY 
- Server side
  - PHP
  - CGI
  - Node.js
  - XML
  - Ruby - Python - Perl ....?
- Client side
  - telnet
  - browsers / clients
  - html
  - CSS
  - JS
  - ...

# HTTP



- Hyper Text Transfer Protocol
  - Protocolo de comunicación para acceder la www
  - Modelo basado en mensajes (request, response).
  - No orientado a conexión

**International** RFC 1945 [🔗](#) HTTP/1.0 (1996)

**standard**

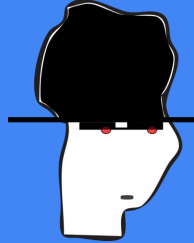
RFC 2616 [🔗](#) HTTP/1.1 (1999)

RFC 7540 [🔗](#) HTTP/2 (2015)

QUIC  
Internet-Draft  
Intended status: Standards Track  
Expires: January 10, 2020

M. Bishop, Ed.  
Akamai  
July 09, 2019

Hypertext Transfer Protocol Version 3 (HTTP/3)  
draft-ietf-quic-http-22



# HTTP: Request

GET /books/search.asp?q=wahh HTTP/1.1

Accept: image/gif, image/xbitmap, image/jpeg, image/pjpeg,  
application/xshockwaveflash, application/vnd.msexcel,  
application/vnd.mspowerpoint, application/msword, \*/\*

Referer: http://wahh-app.com/books/default.asp

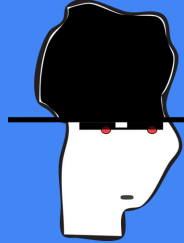
Accept-Language: en-gb,en-us;q=0.5

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)

Host: wahh-app.com

Cookie: lang=en; JSESSIONID=0000tI8rk7joMx44S2Uu85nSWc :vsnlc502



# HTTP: Response

```
HTTP/1.1 200 OK
Date: Sat, 19 May 2007 13:49:37 GMT
Server: IBM_HTTP_SERVER/1.3.26.2 Apache/1.3.26 (Unix)
Set-Cookie: tracking=tI8rk7joMx44S2Uu85nSWc
Pragma: no-cache
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Type: text/html; charset=ISO-8859-1
Content-Language: en-US
Content-Length: 24246
```

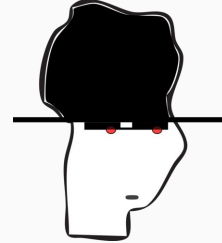
```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1">
...
```

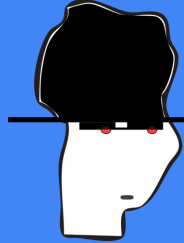


# HTTP: Métodos

- GET
- POST
- HEAD
- OPTIONS
- TRACE
- PUT

- GET:  
Diseñado para la obtención de un recurso
- POST  
Diseñado para efectuar acciones
- HEAD  
Similar a GET, solo devuelve info de los headers
- OPTIONS  
Consulta al servidor los métodos disponibles
- TRACE  
Diagnóstico, el servidor responde con el contenido del request recibido
- PUT  
Permite el upload de un recurso.



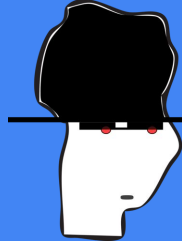


# HTTP: PUT

```
PUT /hello.htm HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.tutorialspoint.com
Accept-Language: en-us
Connection: Keep-Alive
Content-type: text/html
Content-Length: 182
```

```
<html>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```

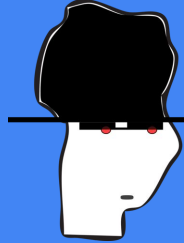
# HTTP/2



```
joe@zoidberg:~$ curl -vso /dev/null --http2 https://www.cloudflare.com/  
* Trying 2606:4700::6811:d109:443 ...  
* TCP_NODELAY set
```

```
* Using HTTP2, server supports multi-use  
* Connection state changed (HTTP/2 confirmed)  
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0  
} [5 bytes data]  
* Using Stream ID: 1 (easy handle 0x558e254aaff0)  
} [5 bytes data]  
> GET / HTTP/2  
> Host: www.cloudflare.com  
> user-agent: curl/7.68.0  
> accept: */*
```

```
HTTP/2 200  
date: Sat, 29 Aug 2020 00:37:34 GMT  
content-type: text/html; charset=utf-8  
set-cookie: __cfduid=d5bb6a5b686807603873b56a77ee5e  
cure  
cf-ray: 5ca234456fc6d2c4-EZE  
cache-control: public, max-age=30  
set-cookie: __cflb=02DiuJFZNR1vT983reJ7dGD57ieW7F4C  
strict-transport-security: max-age=31536000  
vary: Accept-Encoding  
cf-cache-status: EXPIRED
```



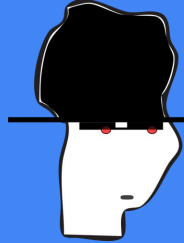
# HTTP: URL

## UNIFORM RESOURCE LOCATOR

`<protocol>://<hostname>[:port][ /path ][?(param=value&)+]`

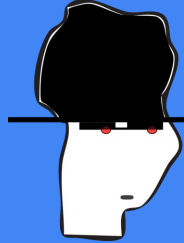
Better desc:

```
URI = scheme:[//authority]path[?query][#fragment] ,  
authority = [userinfo@]host[:port]
```



# HTTP: Cookies

- Permiten al servidor enviar datos (específicos) al cliente
- El cliente guarda datos y los reenvía al servidor en posteriores requests.
- No hay intervención del usuario.
- En general consisten de un par (clave, valor)



# HTTP: Cookies

Se establecen mediante el campo “Set-cookie” en el header del response:

```
content-type: text/html
```

```
Set-cookie: tracking=tI8rk7joMx44S2Uu85nSWc
```

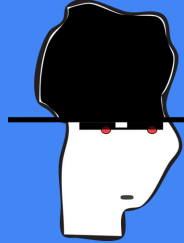
El browser automáticamente agregara a los futuros requests al mismo servidor:

```
Cookie: tracking=tI8rk7joMx44S2Uu85nSWc
```



# HTTP: Status Codes

- ❑ 1xx - Informativos
- ❑ 2xx - Exitoso
- ❑ 3xx - Redirección
- ❑ 4xx - Error en el request
- ❑ 5xx - Error en el servidor



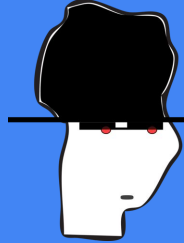
# Server side

Mucho del contenido que se presenta al usuario es generado dinámicamente.

Hay código ejecutándose en el servidor: script, programas, binarios, etc.

Los request agregan parámetros permitiendo al servidor producir una respuesta personalizada(query strings, cookies, POST requests, user-agent, etc)



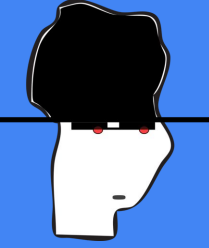
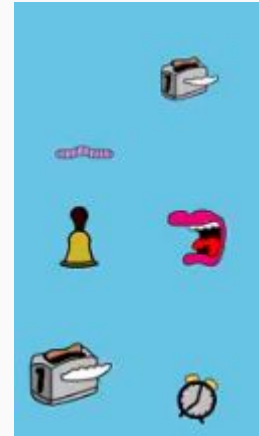
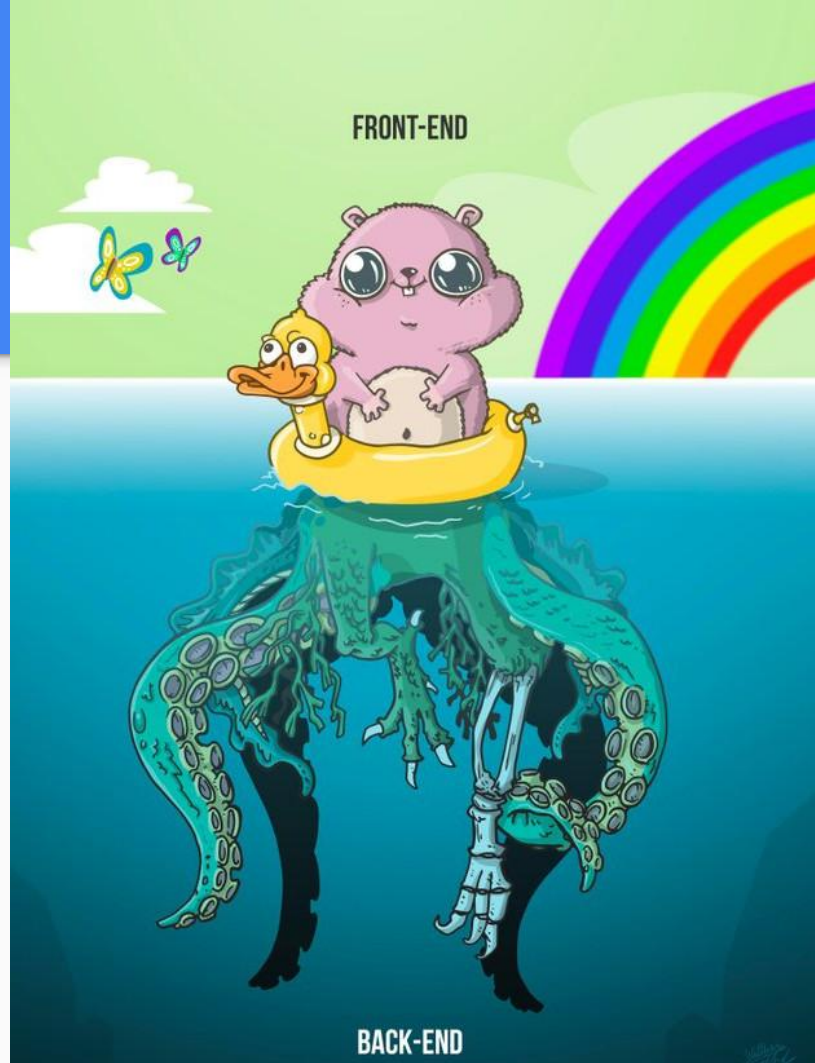
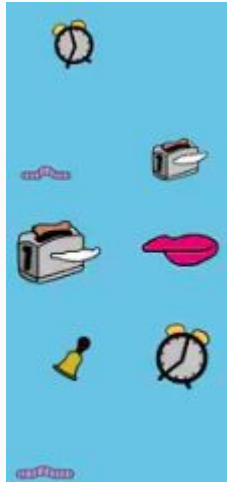


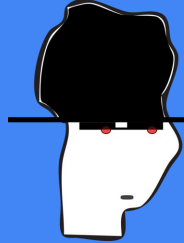
# Server side

Del lado del servidor se emplea un amplio abanico de tecnologías para procesar la respuesta:

- Lenguajes de scripting: php, VBScript, Perl, Python, Ruby.
- Plataformas web: ASP,.NET, Java, Nodejs
- Servidores web: Apache, Nginx, IIS, Netscape Enterprise.
- Motores de bases de datos: Oracle, MySQL, PostgreSQL, Mongo, Maria, MS-SQL.
- Otros componentes: Servicios SOAP, cache, etc.
- .....

# Server side





# Client side: HTML

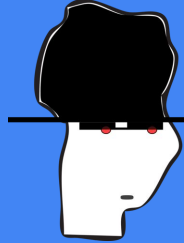
- Lenguaje Tag-based
- Describe la estructura de los documentos a renderizar en el browser
- Flujo de la aplicación dado por los links:

```
<a href="news/showStory?newsid=6466&lang=en">Sale now  
on!</a >
```

Si el usuario sigue este link, se realiza un request:

```
GET /news/showStory?newsid=6466&lang=en HTTP/1.1  
Host: wah-app.com ...
```

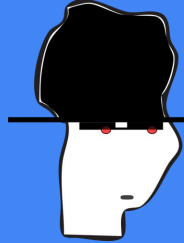
A partir de los parámetros en el query string (newsid, lang) el servidor determina el contenido a mostrarle al usuario.



# Client side: Forms

También se necesita poder ingresar datos para determinadas acciones.  
El mecanismo usual es usar forms, que permiten la entrada arbitraria de datos de los usuarios.

```
<form action="/sec/login.php?app=quotations" method="post">  
username: <input type="text" name="username"><br>  
password: <input type="password" name="password">  
<input type="hidden" name="redir" value="/sec/home.php">  
<input type="submit" name="submit" value="log+in">  
</form>
```

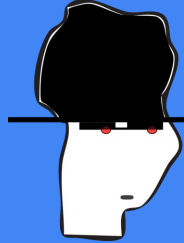


# Client side: Forms

Cuando el usuario ingresa los valores y envía el form, se produce un request:

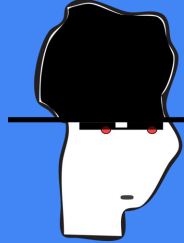
```
POST /sec/login.php?app=quotations ...  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 39  
Cookie: SESS=GTnrpx2ss2tSWSnhXJGyG0LJ47MXRsjcFM6Bd  
username=daf&password=foo&redir=/sec/home.php&  
submit=log+in
```

- Los datos se pasan en el body del request
- Se incluyen el campo oculto y el parámetro submit
- La url de target incluye un parámetro app
- Se incluye las cookies.



# Client side: Javascript

- Lenguaje simple y potente: event-driven, funct, imperative, oo.
- En gral, corre del lado del cliente.
- Permite extender las interfaces web (por ej: validar los datos ingresados por el usuario, modificar la interfaz dinámicamente en respuesta a acciones del usuario, etc)
- Tiene muchas bibliotecas jquery, react, gojs



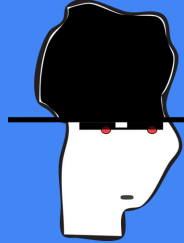
# Both sides: DATA (JSON, XML, ...)

## Response

```
HTTP/1.1 200 OK
Server: Cowboy
Connection: close
Content-Type: text/xml; charset=utf-8
Date: Mon, 23 Jan 2017 18:15:18 GMT
Via: 1.1 vegur

<response>
  <result outcome="rejected">
    <errors>
      <error>Value out of range</error>
      <error>Must be older than 18 years</error>
    </errors>
  </result>
</response>
```

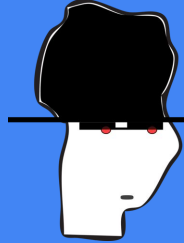
```
{ "3":
  { "1":1,
    "2":"31425826",
    "5":{"2":0}, "7":1},
    "4":{"2":0, "4":0, "5":220}
  },
  "5":2,
  "user": "somebody"
}
```



# Sesiones

- Para implementar funcionalidades útiles, las aplicaciones necesitan poder registrar el estado de las interacciones del usuario a través de múltiples requests
- HTTP es stateless
- Esta info se mantiene en una estructura del lado del servidor denominada sesión.
- Permiten re-identificar al usuario en distintos requests (la mayoría de las veces entregando un token que identifica la sesión mediante cookies).

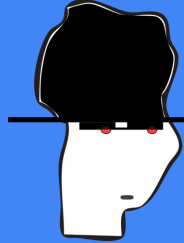




# Sesiones

Pedidos subsecuentes de un usuario están vinculados al mismo a través de la sesión establecida durante la autenticación .

Si el manejo de sesiones es vulnerable, un atacante puede comprometer a los usuarios por más robusto que sea el sistema de autenticación.



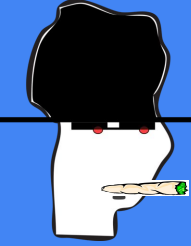
# Principal Problema

Entrada arbitraria de datos por parte de los usuarios.

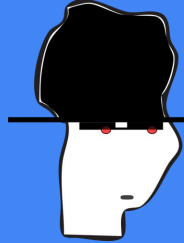
- Los usuarios pueden interferir con cualquier dato que se transmite entre cliente y servidor.
- Los usuarios pueden enviar requests en cualquier orden, con diferentes parámetros (esperados o no), etc
- Los usuarios no están limitados a usar un navegador web para interactuar con una aplicación web

(algunas) Vulnerabilidades web

# Vulnerabilidades

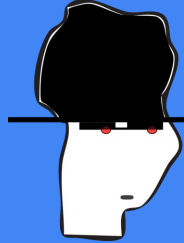


OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	⊗	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	⊗	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]



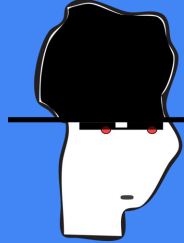
# Enfoque ofensivo

- Un ejercicio “artesano”, involucra entender/deducir la aplicación
- DAST (Dynamic App Security Testing)
- SAST? (Puede incluir code-review)
- Puede incluir herramientas automáticas bajo algún riesgo.
- Es iterativo.



# Command Injection

Son vulnerabilidades que parten de ejecutar comandos de sistema con fines específicos pero delegando parte del control de los comandos a ejecutar a datos controlables por un atacante.



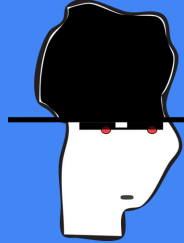
# Command Injection

```
readfile.c  x
#include <stdio.h>
#include <unistd.h>

int main(int argc, char **argv) {
    char cat[] = "cat ";
    char *command;
    size_t commandLength;

    commandLength = strlen(cat) + strlen(argv[1]) + 1;
    command = (char *) malloc(commandLength);
    strncpy(command, cat, commandLength);
    strncat(command, argv[1], (commandLength - strlen(cat)) );

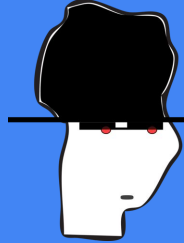
    system(command);
    return (0);
}
```



# Command Injection

```
delete.php x
<?php
    print("Please specify the name of the file to delete");
    print("<p>");
    $file=$ GET['filename'];
    system("rm $file");
?>
```

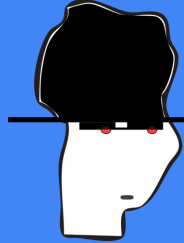




# IDOR (Insecure Direct Object Reference)

Insecure direct object references (IDOR) are a type of **access control** vulnerability that arises when an application uses user-supplied input to access objects directly.

```
https://insecure-website.com/customer_account?customer_number=132355
```

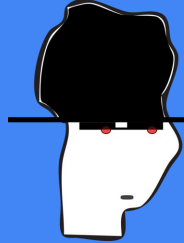


# IDOR (Insecure Direct Object Reference)

Insecure direct object references (IDOR) are a type of **access control** vulnerability that arises when an application uses user-supplied input to access objects directly.

```
https://insecure-website.com/customer_account?customer_number=132355
```

```
https://insecure-website.com/static/12144.txt
```

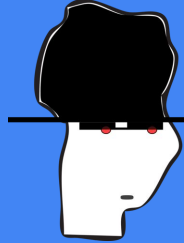


# IDOR (Vimeo2015)

`http://vimeo.com/api/v2/kerrytrainor/info.xml`

```
<users>
  <user>
    <id>10272636</id>
    <display_name>Kerry Trainor</display_name>
    <created_on>2012-02-02 14:12:45</created_on>
    <is_staff>1</is_staff>
    ...
  </user>
</users>
```

Finally, `vimeo.com/user10272636`



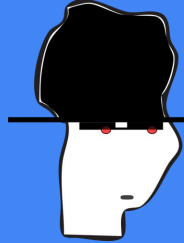
# IDOR (Vimeo2015)

First, an attacker signup for an account and request "forgot password".



**Pas de panique — on s'occupe de tout.** Donnez-nous l'adresse e-mail que vous utilisez pour vous connecter à Vimeo et nous vous enverrons des instructions pour réinitialiser votre mot de passe.

[Help Me](#)



# IDOR (Vimeo2015)

`https://vimeo.com/forgot\_password/[user id]/[token]`



**Saisissez votre nouveau mot de passe.** Un long mot de passe constitué d'une série de chiffres, de symboles et de lettres majuscules et minuscules est plus sécurisé.

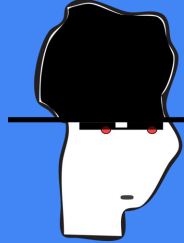
Nouveau mot de passe :

Ressaisissez votre nouveau mot de passe :

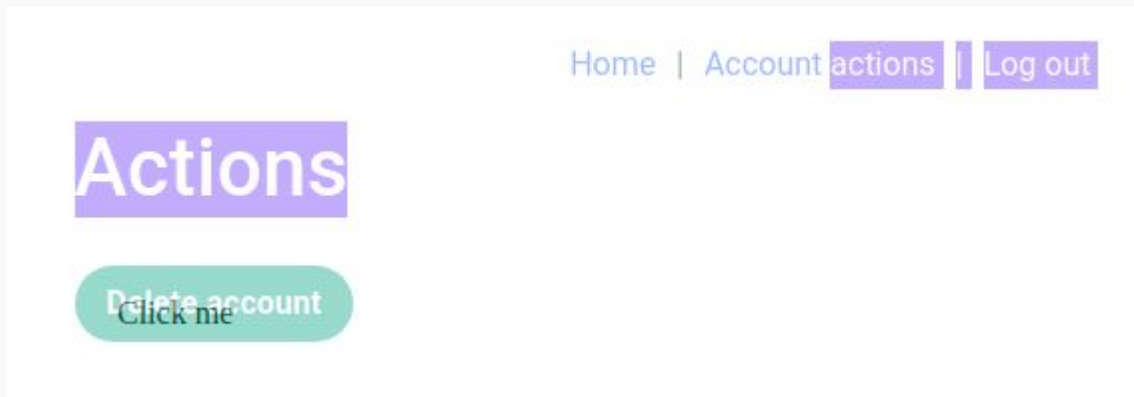
Réinitialiser le mot de passe



```
GET /forgot_password/35771311/zeamn4zkwisrbjqt HTTP/1.1
Host: vimeo.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:36.0) Gecko/20100101
Firefox/36.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: fr,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Cookie: vuid=1002582892; __utma=18302654.4509985; __utmb=18302654.46.9.14; __utmc=18302654; __utmz=18302654.1419898371.1.1.utmcsr=google|utmccn=(organic)|utmcmd=organic|utmcctr=(not%20provided)|2=us; site_settings=%7B%22%3A%22%3Anull%7D; has_logged_in=1; stats_start_date=2014%2F12%2F26; stats_end_date=2014%2F12%2F30; notification_53=1419898371; notification_language_optin=1419898373
Connection: keep-alive
```

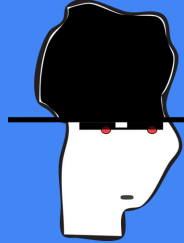


# Clickjacking



Un ataque de clickjacking puede tomar la forma de código embebido o script que se ejecuta sin el conocimiento del usuario.

Por ejemplo: aparentando ser un botón para realizar otra función.

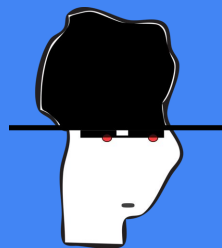


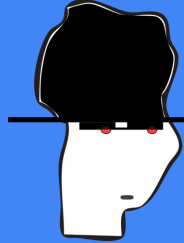
# Clickjacking (example)

## Clickjacking on Google MyAccount Worth 7,500\$

Recently, I got surprised from google, I found bug Clickjacking On Google My account. And they reward me 7,500\$ for single bug. Amazing, right?. This bug I've found on March 2018, but the clickjacking is just blocked by CSP, and on August, I've found way to bypass it.

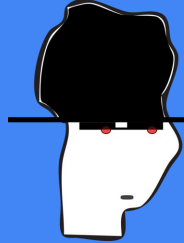






# Mecanismos de defensa

- Controles del lado del cliente
- Autenticación
- Sesiones
- ~~De infraestructura~~



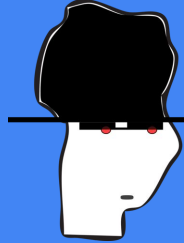
# Mecanismos de defensa

- Controles del lado del cliente
  - Hidden fields
  - Referer header
  - Ofuscación de datos
  - Validación basada en scripts

Pero todo lo que se envía al servidor es controlable por el usuario!

¿Por qué se hace esto?

Simple para el desarrollador (no sesión, performance, usuarios interactuando con múltiples servidores)



# Mecanismos de defensa

- Autenticación

- Basada en forms
- Multi-factor
- Certificados SSL
- Basada en HTTP
- Servicios de terceros.

**Defectos:**

- Mensajes de error demasiado informativos.
- Transmisión de credenciales vulnerables
- Defectos en el proceso de cambio de password
- Defectos en el proceso de recuperacion de password
- Almacenamiento inseguro de credenciales
- Nombres de usuario predecibles

# Árbol de tecnologías web

