Unidad 1 – Red Team

Módulo 3
Ataques a Aplicaciones Web

THE BRIDGE DIGITAL TALENT ACCELERATOR

Ataques a Aplicaciones Web

1. Fundamentos de Aplicaciones Web

- Protocolos de comunicaciones
  - ¿Qué son los protocolos de comunicaciones?
  - Ejemplos de protocolos
- Arquitectura de aplicaciones Web
  - ¿Qué es el protocolo HTTP?
  - Esquema de comunicaciones básico
  - Estructura de peticiones / respuestas
  - Principales métodos HTTP
  - Cookies

2. Ataques básicos en arquitecturas Web
  - Entornos de ejecución cliente / servidor
  - OWASP
    - ¿Qué es OWASP?
    - OWASP Top 10

1. Fundamentos de Arquitectura de Aplicaciones Web

Protocolos de comunicaciones

- ¿Qué son los protocolos de comunicaciones?

  Conjunto de normas **obligatorias** que tienen que cumplir todos los programas que intervienen en una **comunicación**.

  EJEMPLOS DE PROTOCOLOS
  Protocolos ampliamente utilizados a día de hoy :
  - TCP/IP: Comunicación en redes.
  - FTP: Transferencia de archivos.
  - HTTP: Intercambios de información entre Cliente Web y Servidor HTTP.
  - …

Arquitectura de aplicaciones web

¿QUÉ ES EL PROTOCOLO HTTP?

- Llamado **H**yper**T**ext **T**ransfer **P**rotocol o protocolo de Transferencia de Hipertexto.
- Base de cualquier intercambio de datos en la Web.
- Protocolo evolutivo : HTTP 1.0, 1.1, HTTP 2.0 (HTTP/2) años 1990-2015.
- Mensajes legibles en texto plano (excepto HTTP/2).
- Modelo Cliente/Servidor.

  Cliente: Solicita los datos y los recibe del Servidor.

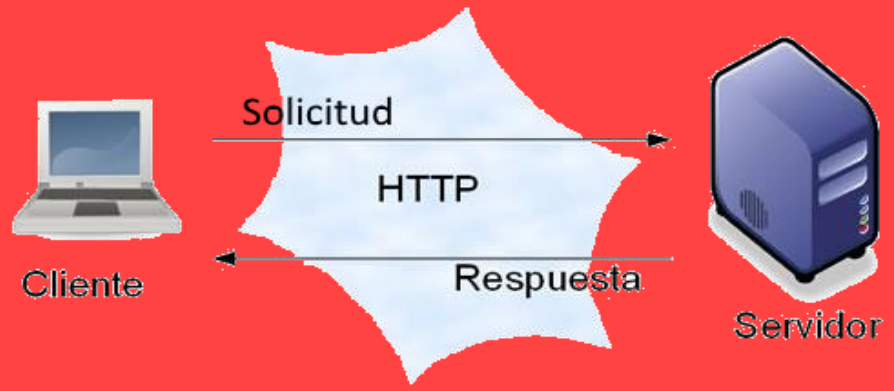
  Servidor: Proporciona los datos al cliente.

Arquitectura de aplicaciones web

ESQUEMA DE COMUNICACIONES BASICO

1. Cliente **solicita** un recurso al servidor (por ejemplo una página web) mediante un **mensaje de petición** HTTP.

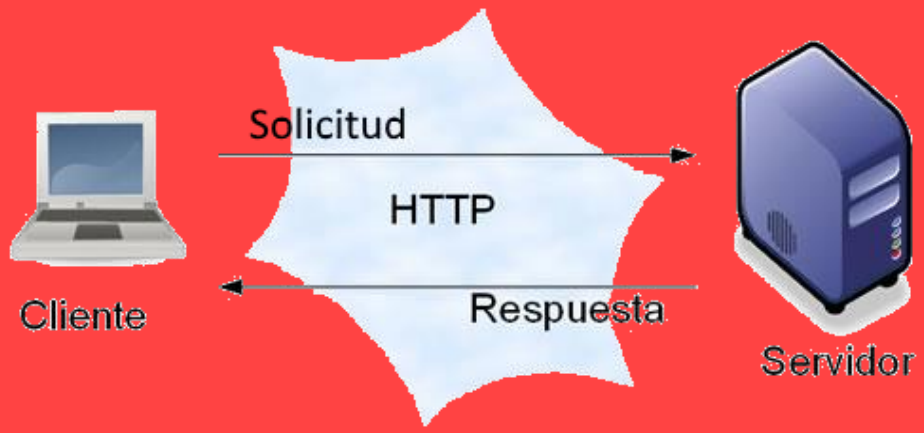
2. El servidor **responde** al cliente mediante un **mensaje de respuesta** HTTP con el contenido solicitado.



Arquitectura de aplicaciones web

- MENSAJE DE PETICION (CLIENTE)

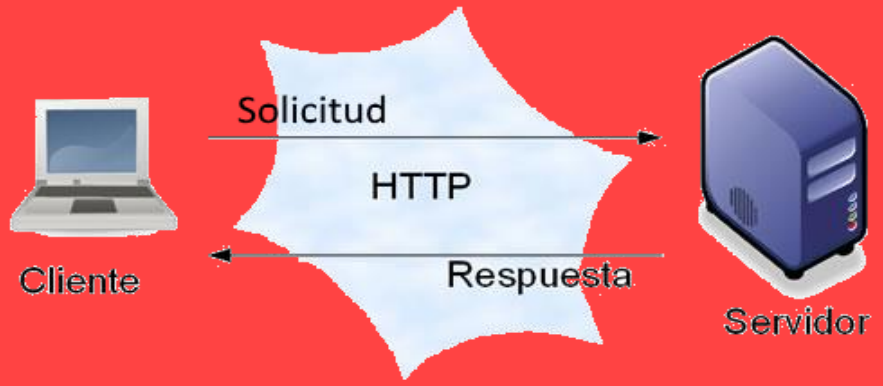
```
GET / HTTP/1.1
Host: www.eltiempo.es
Accept-Language: es
```



Arquitectura de aplicaciones web

- MENSAJE DE PETICION (CLIENTE)

```
GET / HTTP/1.1
Host: www.eltiempo.es
Accept-Language: es
```



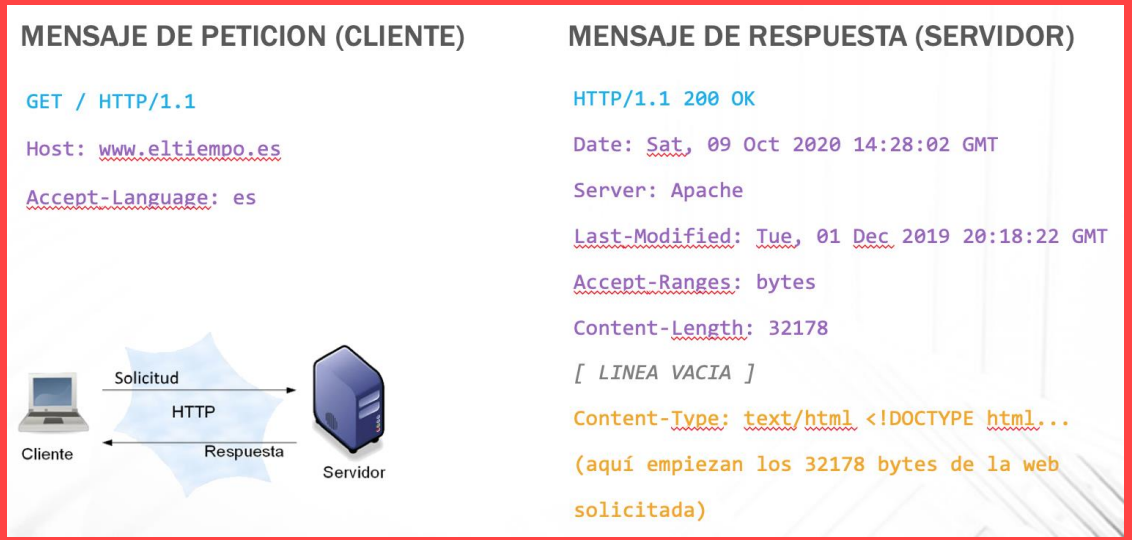
- MENSAJE DE RESPUESTA (SERVIDOR)

```
HTTP/1.1 200 OK
Date: Sat, 09 Oct 2020 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2019 20:18:22 GMT
Accept-Ranges: bytes
Content-Length: 32178
[ LINEA VACIA ]
Content-Type: text/html <!DOCTYPE html...
(aquí empiezan los 32178 bytes de la web
solicitada)
```

Arquitectura de aplicaciones web

ESTRUCTURA DE PETICIONES/RESPUESTAS

- Línea de inicio (Petición)
  - Método HTTP: GET, PUT, POST…
  - Objetivo: Dirección URL
  - Versión del protocolo HTTP utilizado
- Línea de estado (Respuesta)
  - Versión del protocolo HTTP utilizado
  - Código de estado: 200, 404…
  - Texto descripción de estado : “Ok”, “Not found”…



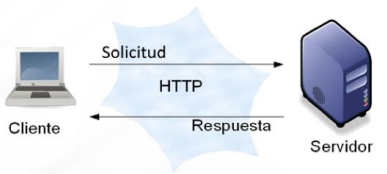
Arquitectura de aplicaciones web

ESTRUCTURA DE PETICIONES/RESPUESTAS

- Cabeceras (Opcionales)
- “Cadena de texto”: “valor” (son variables)
    - Generales: Afectan a todo el mensaje.
    - Petición: Añaden detalle a la petición.
    - Entidad: Afectan al cuerpo.
    - Cookies

MENSAJE DE PETICION (CLIENTE)

```
GET / HTTP/1.1
Host: www.eltiempo.es
Accept-Language: es
```



MENSAJE DE RESPUESTA (SERVIDOR)

```
HTTP/1.1 200 OK
Date: Sat, 09 Oct 2020 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2019 20:18:22 GMT
Accept-Ranges: bytes
Content-Length: 32178
[ LINEA VACIA ]
Content-Type: text/html <!DOCTYPE html...
(aquí empiezan los 32178 bytes de la web
solicitada)
```

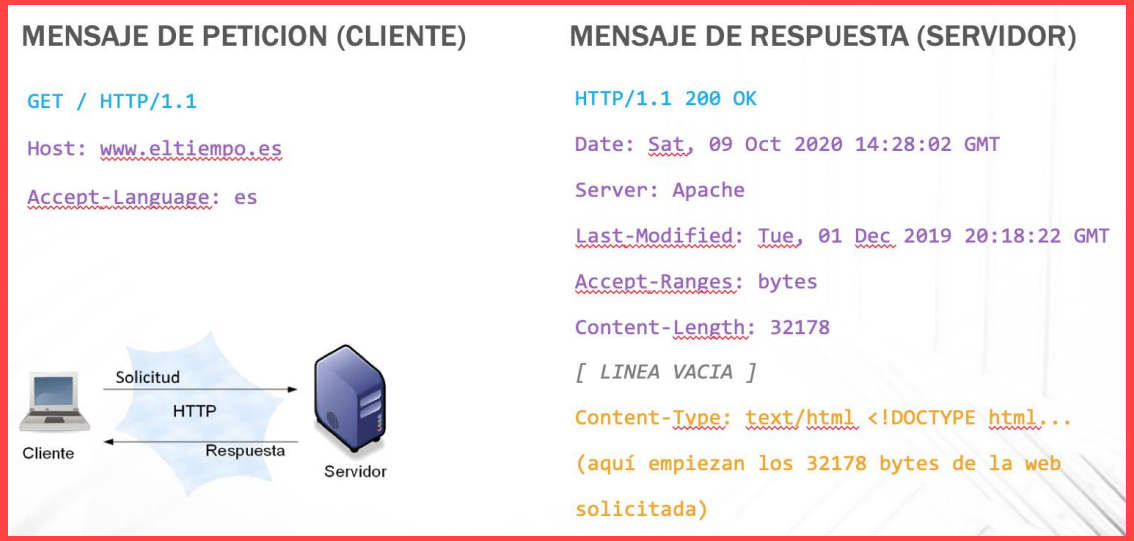
Arquitectura de aplicaciones web

ESTRUCTURA DE PETICIONES/RESPUESTAS

- Cuerpo (Opcional)
  - Cuerpos con un único dato

    Content-Type / Content-Length
  - Cuerpos con múltiples datos

    Formularios HTML



Arquitectura de aplicaciones web

PRINCIPALES MÉTODOS HTTP

- GET
- POST
- PUT
- DELETE
- HEAD
- …

Arquitectura de aplicaciones web

PRINCIPALES MÉTODOS HTTP

- GET

  Obtiene información del servidor, trayendo los datos que están en el servidor al cliente.

  - Si la petición es **correcta**, tendremos en la línea de estado la combinación (200 OK)
  - Si la petición es **errónea**, tendremos en la línea de estado el mensaje de error del servidor especificado (404 Not Found, 400 Bad Request…)

Arquitectura de aplicaciones web

PRINCIPALES MÉTODOS HTTP

- GET

  Ejemplo 1: Solicitamos información (un enlace web) desde nuestro navegador al servidor, y ésta es devuelta como respuesta.

  Línea de inicio

  ```
  GET /test/web.html HTTP/1.1
  ```

  Ejemplo 2: Noticia concreta

  Solicitamos información (un enlace web) desde nuestro navegador al servidor enviándole el tipo de noticia que queremos y el identificador de la noticia.

  Línea de inicio

  ```
  GET /test/formulario.php?tipo_noticia=valor1&id=valor2 HTTP/1.1
  ```

Arquitectura de aplicaciones web

PRINCIPALES MÉTODOS HTTP

- POST

  Envía información desde el cliente para actualizar o añadir en el servidor.

  Habitualmente se utilizan formularios.

  - Si la petición es **correcta**, tendremos en la línea de estado la combinación (201 Created)
  - Si la petición es **errónea**, tendremos en la línea de estado el mensaje de error del servidor especificado (404 Not Found, 400 Bad Request…)

Arquitectura de aplicaciones web

PRINCIPALES MÉTODOS HTTP

- POST

  Ejemplo 1 : Formulario con campos Nombre y Email.

  Línea de Inicio :

  ```
  POST /formulario_post.php HTTP/1.1
  ```

  Cabeceras :

  ```
  Content-Type: application/x-www-form-urlencoded
  Content-Length: 43
  ```

  Cuerpo :

  [Línea en blanco]

  ```
  formulario_post.php?v_nombre=usuario&v_email=usuario%40thebridge.tech
  ```

```
<form action=”formulario_post.php" method="post">
  Nombre: <input type="text" name=”v_nombre"><br>
  Email: <input type="text" name=”v_email"><br>
  <input type="submit" value="Enviar">
</form>
```

Arquitectura de aplicaciones web

PRINCIPALES MÉTODOS HTTP – COMPARATIVA GET VS POST

- GET

- Se utiliza para **recuperar** información
- Los parámetros se quedan en el histórico del navegador (están en la dirección)
- Pueden añadirse a direcciones favoritas
- Se guardan en caché de navegador
- Tamaño de variables depende del máximo de tamaño de la dirección

- POST

- Se utiliza para **actualizar** información
- Los parámetros **no** se quedan en el histórico del navegador (están en el Cuerpo)
- **No** pueden añadirse a direcciones favoritas
- **No** se guardan en caché de navegador
- **No** hay tamaño máximo de variables

https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/get-vs-post/

Arquitectura de aplicaciones web

COOKIES

- Información enviada y recibida en las Cabeceras HTTP.
- Queda almacenada en el lado cliente durante un tiempo determinado.
- Identifica de manera unívoca un usuario durante ese periodo de tiempo.

```
GET /ejemplo.html HTTP/1.1

Host: www.mysite.com

Cookie: colorPreference=blue;
sessionToken=48745487
```

NUEVA PETICION CLIENTE

```
HTTP/1.0 200 OK

Content-type: text/html

Set-Cookie: colorPreference=blue

Set-Cookie: sessionToken=48745487;
Expires=Thu, 01 Jan 2031 19:22:10 GMT
```

RESPUESTA DEL SERVIDOR

Herramientas de Desarrollador

Elements: Aquí podemos visualizar el código HTML, además de poder eliminar o modificar partes del mismo de forma local, para así ver cómo queda la página.
Console: En esta pestaña se nos habilita una consola en la que podemos insertar código JavaScript para realizar todo tipo de pruebas.
Sources: Aquí podemos ver todos los archivos que se han descargado para la página correspondiente, ya sean archivos JavaScript, HTML entre otros tipos.
Network: En esta pestaña se nos muestran las cabeceras y lo que han tardado en cargarse entre otras cosas.
Performance: Aquí podemos ver datos del rendimiento, como por ejemplo hacer un perfil de cuánto tarda en cargar la página.
Memory: Desde aquí podemos comprobar la memoria que ocupa la web que se ha cargado.
Application: Aquí podemos ver, por ejemplo, las cookies que tenemos.
Security: Dónde podemos ver todo lo relacionado con la seguridad de la web.
Audits: En esta pestaña podemos realizar auditorías de la página web.

Ejemplo:

http://www.brainjar.com/java/host/test.html

2. Ataques básicos en arquitecturas web

Ataques en arquitecturas web

ENTORNOS DE EJECUCIÓN CLIENTE/SERVIDOR

- Código servidor: Código que se ejecuta en el lado del servidor Web una vez se ha realizado una petición.
- Código cliente: Código que se ejecuta en el lado del cliente (navegador Web) cuando el navegador carga una página web.

En aplicaciones Web modernas cada vez hay más código cliente (sobre todo en lenguaje JavaScript) que se utiliza para gestionar contenido interactivo sin realizar sucesivas consultas al servidor por cada interacción.

Esto abre puertas a nuevas superficies de ataque y tipos de vulnerabilidades que pueden ser aprovechadas por terceros.

Ataques en arquitecturas web

OWASP

- **O**pen **W**eb **A**pplication **S**ecurity **P**roject (Proyecto Abierto de Seguridad de Apps Web).
- Organización sin ánimo de lucro.
- Proporcionan guías, herramientas, conferencias, documentación…
- OWASP “Top 10” : 10 riesgos de seguridad más importantes (Web, Apps, IoT…)

https://owasp.org/

Ataques en arquitecturas web

OWASP TOP 10 HISTÓRICO

OWAPS TOP 10 - 2007	OWAPS TOP 10 - 2010	OWAPS TOP 10 - 2013	OWAPS TOP 10 - 2017
A1 - Cross Site Scripting (XSS)	A1 – Injection	A1 – Injection	A1 – Injection
A2 - Injection Flaws	A2 – Cross Site Scripting (XSS)	A2 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Management
A3 - Malicious File Execution	A3 – Broken Authentication and Session Management	A3 – Cross-Site Scripting (XSS)	A3 – Sensitive Data Exposure
A4 - Insecure Direct Object Reference	A4 – Insecure Direct Object References	A4 – Insecure Direct Object References	A4 – XML External Entity (XXE)
A5 - Cross Site Request Forgery (CSRF)	A5 – Cross Site Request Forgery (CSRF)	A5 – Security Misconfiguration	A5 – Broken Access Control
A6 - Information Leakage and Improper Error Handling	A6 – Security Misconfiguration (NEW)	A6 – Sensitive Data Exposure	A6 – Security Misconfiguration
A7 - Broken Authentication and Session Management	A7 – Insecure Cryptographic Storage	A7 – Missing Function Level Access Control	A7 – Cross-Site Scripting (XSS)
A8 - Insecure Cryptographic Storage	A8 – Failure to Restrict URL Access	A8 – Cross-Site Request Forgery (CSRF)	A8 – Insecure Deserialization
A9 - Insecure Communications	A9 – Insufficient Transport Layer Protection	A9 – Using Components with Known Vulnerabilities	A9 – Using Components with Known Vulnerabilities
A10 - Failure to Restrict URL Access	A10 – Invalidated Redirects and Forwards (NEW)	A10 – Invalidated Redirects and Forwards	A10 – Using Components with Known Vulnerabilities

Ataques en arquitecturas web

OWASP TOP 10 2021