**Certificado Profesional de Hacking Ético** (CPHE) – **Security Sentinel**

# **1.- Introducción:**

El **Ciclo de la Auditoría de Seguridad** Ofensiva:

1. Recolección de Información (*Information Gathering*)
2. Escaneo y Enumeración.
3. Análisis de Vulnerabilidades.
4. Explotación de Vulnerabilidades.
5. Post-Explotación
6. Informe
7. *Eliminación Rastro y Huellas* (*Esta fase existe desde el punto de vista de un Ciberdelincuente*)

Podemos diferenciar **3 Tipos de Análisis de Seguridad** **en una Auditoría**, en función del **conocimiento** que tenga el **Pentester** sobre su **Objetivo**:

1. **Black Box**: Donde el Pentester **no tiene ningún conocimiento a cerca del Objetivo**, pero el cliente si tiene conocimiento del análisis, además de saber cuando se va a realizar.
2. **Gray Box**: Donde el Pentester **conoce muy poco del Objetivo** y el cliente, como en el caso anterior, sabe que se va a realizar el análisis y cuando se va a realizar.
3. **White Box**: Donde el **Pentester tienen bastante información sobre su Objetivo** y el cliente, de nuevo, sabe que se va a realizar el análisis y cuando se hará.

¿Qué **Tipos de Auditorías** podemos realizar?:

1. **Auditoría Externa**: Se realiza a un Objetivo (*Servidor, Red, etc.*) **desde el exterior**, es decir, de **forma remota**.
2. **Auditoria Interna**: Se realiza **en las instalaciones del cliente**, es decir, **formamos parte de la propia Red** del cliente.
3. **Auditoria Web**: Se realiza a un **Sitio o Aplicación Web**, evidentemente, este tipo de auditoría también es **externa**.
4. **Auditoria Wifi**: Se realiza a una **Red Wireless (Wifi)**, tanto **desde dentro** como **desde fuera** de las instalaciones del cliente.
5. **Auditoria a una Aplicación**: Se realiza sobre una **aplicación concreta**, se buscan posibles **errores**/**bugs** de seguridad.
6. **Auditoria Móvil**: Se realiza a un **dispositivo** **o** **red de dispositivos móviles** del objetivo.

# **2.- Arquitectura Básica de Redes:**

Por lo general, las **Redes** se dividen en **3 Grupos** principales:

1. **LAN** (*Local Area Network*).
2. **MAN** (*Metropolitan Area Network*).
3. **WAN** (*Wide Area Network*)

Las **Redes** de dispositivos, están basadas en **2 Modelos de Referencia**: **Modelo OSI** y **Modelo TCP/IP**, a su vez, cada uno de estos **Modelos**, está dividido en una serie de **Capas** donde, cada una tiene asociados uno o mas **Protocolos de Red**.

**NOTA**:

Para conocer más en profundidad el tema de las Redes Informáticas y su seguridad, es recomendable realizar el curso de **Seguridad en Redes CPSR**.

|  |  |  |
| --- | --- | --- |
| El **Modelo OSI**: | El **Modelo TCP/IP**: | **Descripción**: |
| 7. Capa de **Aplicación**. | 1. Capa de **Aplicación**. | **Servicios y** **Aplicaciones** que usa un **Usuario**. |
| 6. Capa de **Presentación**. | Se asegura de que la **información** se **transfiera** de un **modo comprensible** por el sistema receptor. |
| 5. Capa de **Sesión**. | Establece, Administra y Termina **las conexiones** entre los **Hosts**. |
| 4. Capa de **Transporte**. | 2. Capa de **Transporte**. | Se encarga de efectuar el **transporte de los datos** del origen al destino. |
| 3. Capa de **Red**. | 3. Capa de **Internet**. | Se encarga de **identificar el enrutamiento** existente entre una o mas **Redes**. |
| 2. Capa de **Enlace de Datos**. | 4. Capa de **Acceso a la Red.** | Se ocupa del **direccionamiento FISICO**, **acceso al medio** y **detección de errores**, **distribución** **ordenada** de **Tramas** y del **control del flujo**. |
| 1. Capa **Física**. | Define las características del **Hardware de Red**. |

**Definiciones Importantes:**

**Dirección IP**: Direcciones binarias de 32 bits que se utilizan en las redes para **identificar un Host** concreto dentro de estas. Son un **conjunto de 4 octetos** (*8 bits/1 byte*) **separados por un punto** (*xxx.xxx.xxx.xxx* / 192.168.1.1).

**Mascara de Subred**: Permite **delimitar el ámbito de una red**. Indica a los dispositivos, que parte la **Dirección IP** de otro dispositivo es el **identificador de la red/subred** a la que está conectado y que parte identifica a ese dispositivo. Tiene el mismo formato que la **Dirección IP**.

**Puerta de Enlace/Gateway**: Dispositivo localizado en una red, por donde se envían los paquetes que tienen como destino una **red exterior**.

**Clases de Direcciones IP**:

3 tipos principales de **Clases de IP** mas ciertas **Direcciones Especiales**:

1. **Red de Clase A**: Se asigna el **Primer Octeto** (8 primeros bits) para **Identificar la Red**, reservando los **otros Tres Octetos** (24 bits) para que sean asignados **a los Hosts**, de modo que la cantidad máximo de **Hosts** es de: **224 - 2** (\*), aproximadamente **17 millones de Hosts**.
2. **Red de Clase B**: Se asignan los **Dos Primeros Octetos** (16 primeros bits) para **Identificar la Red**, reservando los **Dos Octetos Finales** (16 bits) para asignarlos **a los Hosts**, de modo que la cantidad máxima de **Hosts** es de: **216-2** (\*), es decir **65534** **Hosts**.
3. **Red de Clase C**: Se asignan los **Tres Primeros Octetos** (24 primeros bits) para **Identificar la Red**, reservando los **el Octeto Final** (8 bits) para asignarlos **a los Hosts**, de modo que la cantidad máxima de **Hosts** es de: **28-2** (\*), es decir **254** **Hosts**.

**(\*) NOTA**:

En los tres casos, se excluyen las denominadas **Dirección IP de Broadcast** (Multidifusión) y la **Dirección IP Red**. La **primera Dirección IP de la Clase**, se reserva para **Identificar la Red**, mientras que **la última** se emplea para **Broadcast** o **Difusión.** Por eso, el **número máximo** de **Hosts** en una red es siempre igual al **número de Direcciones IP disponibles** en un rango **menos dos**.

Dicho de otra forma, la **Dirección IP** que tiene los **bits de Hosts iguales a 0**, sirve para definir la red en la que se ubica (**Dirección de Red**). La **Dirección IP** que tiene los **bits correspondientes a Hosts iguales a 1**, sirve para enviar paquetes a todos los **Hosts** de la red en la que se ubica (**Dirección de Broadcast**).

**NOTAS**:

* Las **Direcciones IP 127.x.x.x**, se reservan para **designar a la propia máquina**. Es la denomina **Dirección de Bucle Local** o **LoopBack**.
* La **Dirección IP 0.0.0.0** está reserva por la **IANA** para **Identificación Local.**

**Clases de Direcciones IP** (Resumen):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Clase** | **Bits** | **Intervalo** | **Redes** | **Direcciones** | **Hosts** | **Máscara Red** | **Broadcast** |
| **A** | 0 | 0 – 127 | 126 | 16777216 | 16777214 | 255.0.0.0 | x.255.255.255 |
| **B** | 10 | 128 – 191 | 16384 | 65536 | 65534 | 255.255.0.0 | x.x.255.255 |
| **C** | 110 | 192 – 223 | 2097152 | 256 | 254 | 255.255.255.0 | x.x.x.255 |

**ANEXO: Modos de Configuración Adaptadores de Red:**

Existen diversos modos de configurar la **Conexión de Red** de una **Máquina Virtual**, en función del modo seleccionado, podremos:

* Establecer conexiones **entre** el **Anfitrión** y las **Máquinas Virtuales**.
* Establecer conexiones **entre** las **Máquinas Virtuales**.
* Establecer conexiones **desde** las **Máquinas Virtuales** a **Internet**.
* Establecer conexiones **desde** **Internet** a las **Máquinas virtuales**.

1. **Modo BRINDGE** (Puente):

La **MV** se conecta a la **misma red** que el **Anfitrión**, la MV se comporta como **un equipo más conectado a la red** **‘*real’***, **permite conexión entre MV’s**, **con el Anfitrión y con Internet**.

1. **Modo HOST-ONLY**:

Permite la **conexión entre las MV’s y el Anfitrión** y la **conexión entre las MV’s**, las MV’s no dispondremos de conexión a Internet en ningún sentido.

1. **Modo NAT** (Network Address Translation):

**Lo contrario al modo anterior**, las **MV’s tendrán conexión a Internet**, pero para poder conectar con las MV’s desde el exterior (Internet), es necesario **MAPEAR PUERTOS mediante NAT**.

1. **Modo** **RED** **NAT**:

**Igual que en el modo NAT**, pero **permite la conexión entre MV’s**.

1. **Modo Interno**:

**El mas restrictivo**, sólo **permite la conexión entra las MV’s**, no permite la conexión con el Anfitrión, ni la conexión con Internet.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **MV 🡨🡪 Host** | **MV 🡨🡪 MV** | **MV 🡪 Internet** | **MV 🡨 Internet** |
| **Host-Only** | **SI** | **SI** | NO | NO |
| Modo **Interno** | NO | **SI** | NO | NO |
| Modo **Bridge** | **SI** | **SI** | **SI** | **SI** |
| Modo **NAT** | NO | NO | **SI** | Reenvío de Puertos. |
| Modo **Red NAT** | NO | **SI** | **SI** | Reenvió de Puertos. |

Cuadro resumen **Modos Conexión** Virtualbox.

**Conexión Directa y Conexión Reversa:**

1. Conexiones **Directas**:

En este tipo de conexiones entre Hosts, el **Host\_Origen** se conecta a un **Host\_Destino** a través de un **puerto abierto** en el **Host\_Destino**.

1. Conexiones **Reversas**:

En este caso, el **Host\_Origen** tiene un **puerto abierto ‘*escuchado’*** para cuando el **Host\_Destino** este operativo se produzca la conexión.

# **3.- Recolección de Información** (*Sitios Web/Redes Internas*)**:**

En este punto vamos a ver algunas técnicas de recopilación de información que posteriormente van a constituir la base para las fases de Explotación y Post-Explotación. Hay que tener en cuenta que la **Fase** **de Recolección de Información** o **Information Gathering** puede considerarse la **más importante** de las fases de un **Proceso de Pentesting**, de hecho, el éxito de un futuro ataque, dependerá en gran medida de cómo se haya realizado esta recolección de información.

La **recolección de información** también recibe el nombre de ‘**Reconocimiento’** (*Reconnaisse*), es en esta fase donde el **Pentester**/**Auditor**, **debe reunir de forma metodológica toda la información posible sobre un objetivo** con el propósito de posteriormente poder lanzar un ataque. Una **ecuación simple** **que hay que tener en cuenta es**:

**MAS INFORMACIÓN RECOLECTADA = mayor probabilidad de ÉXITO en el ataque**

**NOTA IMPORTANTE**:

En muchas ocasiones, esta fase de recolección se deja, **SOLO**, en manos de **herramientas ‘AUTOMATICAS’** y esto **no es una buena metodología**, esta bien utilizar estas herramientas, pero siempre como complemento de **un profundo trabajo de recolección de información ‘MANUAL’**.

Algunas de las características del proceso de **Recolección de Información** serán:

* No se lleva a cabo **Ningún tipo de Escaneo ni Contacto** con la máquina objetivo.
* Permite elaborar un **Perfil del Objetivo** **Sin Interactuar** son el.
* Disponemos de **Menos Herramientas** para su realización que en otras fases del proceso.
* Se basa principalmente en la recolección de **Información Pública** (*OSINT*).

**Técnicas para la Recolección de Información de un SITIO WEB/DOMINIO:**

Son muchas las técnicas que se pueden utilizar en el proceso de **Information Gathering**, algunas de las más utilizadas son:

1. Herramientas a través de la **Consola de Comandos**. (***Kali Linux***).
2. **Sitios Web Corporativos** (***robots.txt*** */* ***sitemap.xml***).
3. Google Hacking (***Dorks***). (\*0xWord)
4. **Plugins/Complementos** Recolectores de Información (para *Firefox*, *Chrome*, etc.).
5. Los **Metadatos**. (\*0xWord)
6. **Ingeniería Social**.

**NOTA**: Alguna de las técnicas de recolección que se verán en este punto, también pueden utilizarse para reunir información en una **Red Corporativa**.

**a. Herramientas de Recolección utilizando la Consola de Comandos:**

Son muchas las **herramientas y protocolos de red** disponibles para realizar recolección de información, la combinación de estas herramientas, permiten obtener información importante de nuestro objetivo.

1. **WhoIs**:

Aunque antiguo, **WhoIs** sigue ofreciendo información importante sobre el **propietario** de un **Nombre de Dominio** o **Dirección IP**. Esta presente en muchos S.O.:

***whois*** *‘nombre\_dominio\_objetivo’*

**NOTA/RECOMENDACION**: La información ofrecida por WhoIs puede ‘ocultarse’ para evitar su exposición pública.

1. **Ping** (*Mensajes* *ICMP*):

Muy sencillo, envía mensajes a un **Host** determinado solicitando una respuesta y mide el tiempo de ida y vuelta de esas peticiones/respuestas. Nos permite saber si un ***Host*** objetivo se encuentra operativo y su **Dirección IP**.

***ping*** *‘nombre\_dominio\_objetivo/dirección\_ip\_objetivo’*

1. **Tracert**/**Traceroute**:

Nos ofrece información a cerca de la **ruta que toma un Paquete de Red** desde nuestro ***Host*** hasta el ***Host*** objetivo. Nos muestra los ‘**saltos**’ que hace el paquete a través de los distintos **Routers** por los que va pasando hasta llegar a su destino.

Esta información puede sernos útil para averiguar el ***ISP*** (***Internet Service Provider***) de nuestro objetivo y a partir de aquí extraer información como su **país de origen**, **localización** de los **Servidores**, etc.

***traceroute*** *‘nombre\_dominio\_objetivo/dirección\_ip\_objetivo’ 🡪 Sistemas Linux/Unix/OSX*

*o*

***tracert*** *‘nombre\_dominio\_objetivo/dirección\_ip\_objetivo’ 🡪 Sistemas Windows*

**¿Cómo leer el resultado e Tracert/Traceroute?:**

Cada una de las líneas que aparecen, representan un ***router*** por el que pasan los paquetes enviados hasta llegar al destino. Cada una de estas líneas nos muestra la siguiente **información**:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Salto** | **Nombre Router** | **IP Router** | **Tpo. Respuesta 1** | **Tpo. Respuesta 2** | **Tpo. Respuesta 3** |

**NOTA**: En muchas ocasiones la línea solo mostrara una serie de **asteriscos seguidos** (\* \* \*), esto puede significar que el ***router*** por el que pasan los paquetes se encuentra detrás de un ***Firewall*** u otro sistema de seguridad que bloquea este tipo de paquetes de red.

1. **Consultas DNS:**

Este tipo de consultas pueden ofrecernos mucha información importante de nuestro objetivo y consisten básicamente en preguntar a los **Servidores de Nombre de Dominio** (***DNS***) que saben de nuestra ‘victima’.

Para realizar este tipo de consultas utilizamos el comando ‘***nslookup’***:

La principal función de ‘***nslookup’*** es **consultar**, **obtener información** y **solucionar problemas** de los **Servidores DNS** que su utilizan en una conexión, así como realizar la **Resolución Inversa de Direcciones IP**.

Cuando se utiliza ‘***nslookup’***, veremos de forma frecuente estos dos términos:

* + **Authoritative Answer**: La respuesta **DNS** se ha producido desde un **Servidor DNS** que tiene **TODO** el archivo (*Archivo de Zona*) de información disponible para esa **Zona** (*Zona DNS*).
  + **Non Authoritative Answer**: La respuesta **DNS** se ha producido desde un **Servidor DNS** que tiene en su caché una **COPIA** de las consultas realizadas para esa **Zona** al **Servidor DNS** que tiene la **AUTORIDAD** para responder (*el que tiene el Fichero de Zona*).

Podemos utilizar ‘***nslookup’*** de dos formas distintas: **Normal** o **Interactivo**.

* **Modo Normal** (*No Interactivo*):

En la **Línea de Comandos** se introduce el **nombre del comando**, seguido de las **opciones** que se quieren ejecutar:

***nslookup*** *‘nombre\_dominio\_objetivo’ [opciones]*

Otro tipo de petición que podemos solicitar en el **Modo Normal**, es una **Resolución Inversa**, es decir, **conocer a que Nombre de Dominio o Servidor apunta una determinada Dirección IP**:

***nslookup*** *´dirección\_ip’*

* **Modo Interactivo:**

Para **consultas más específicas**, es recomendable el **Modo Interactivo**, se ejecuta el comando y después realizamos las consultas.

El uso básico que se suele realizar de ‘***nslookup’*** es para conocer cual es la **Dirección IP** de un determinado **Dominio** (*Registro A del Fichero de Zona*), pero podemos utilizarlo también para recopilar otro tipo de información.

Podemos realizar **peticiones**/**consultas** de otros **Tipos de Registros** del **Fichero de Zona** además del **Registro A** que ya hemos visto:

1. **set type = A**, para buscar los **Registros A**.
2. **set type = PTR,** para buscar **Registros Reversos**.
3. **set type = MX**, para buscar los **Registros Mail Exchange** de correo.
4. **set type = TXT**, para buscar **Registros de Texto** (*SPF o DKIM*).
5. **set type = CANAME**, para buscar **Alias del Dominio**.
6. **set type = NS**, para buscar Registros de **Servidores de Nombres de Dominio**.

Con el parámetro ‘***set type***’ establecemos que **Tipo de Registro** del **Fichero de Zona** queremos consultar, luego basta con teclear el **Nombre del Dominio** que queramos consultar.

*>* ***set type=MX***

*> ‘nombre\_dominio\_objetivo’*

Podemos cambiar el **Servidor DNS** sobre el cual queremos realizar las consultas (necesario para intentos de **Transferencias de Zona**):

***server*** *‘direccion\_ip\_servidor\_dns’*

Podemos solicitar **toda la información** que tenga el **Servidor DNS** sobre un **Dominio**:

*>* ***set debug***

*> ‘nombre\_dominio\_objetivo’*

**ANEXO**:

**La Transferencia de Zona DNS:**

Una **Transferencia de Zona** consiste básicamente, en **conseguir una copia del Fichero/Ficheros de Zonas de un determinado Servidor DNS**.

Podemos intentar realizar una **Transferencia de Zona** con ‘***nslookup’*** con la siguiente secuencia de instrucciones:

*>* ***nslookup***

*>* ***set type******=******ns***

*> ‘nombre\_dominio\_objetivo’*

*>* ***server*** *‘algun\_servidor\_dns\_del\_objetivo’*

*>* ***ls*** *‘nombre\_dominio\_objetivo’*

**IMPORTANTE**:

La herramienta ‘***nslookup***’ que incluye **Linux**, **no incorpora la orden** ‘***ls***’, por lo tanto, para ese **último paso** (*para los anteriores podemos utilizar ‘****nslookup****’*), se debe utilizar en su lugar la herramienta **DIG** (*Domain Information Groper*) para realizar el intento de **Transferencia de Zona**:

***dig axfr******@****’algun\_servidor\_dns\_del\_objetivo’ ‘nombre\_dominio\_objetivo’*

1. **DNSEnum**:

El propósito de ‘***dnsenum***’ es conseguir tanta información como sea posible sobre un **Dominio** a través de la información almacenada en los **Servidores DNS**. Las operaciones que lleva a cabo son:

1. Obtener la **Dirección IP** del **Host**.
2. Obtener los **Servidores DNS** del **Dominio**.
3. Obtener los **Servidores de Email** del **Dominio**.
4. Realizar una **Transferencia de Zona**.
5. Obtener **Subdominios** a través de consultas a **Google**.
6. Obtener **Subdominios** a través de un ataque de **Diccionario**.

**NOTA**:

Por defecto, ‘***dnsenum’*** realizará sus consultas al **Servidor DNS** que tengamos configurado para nuestra conexión. Para especificar el Servidor DNS sobre el que se quiere ejecutar la herramienta, utilizaremos la opción ‘***--dnsserver***' y la **Dirección IP** del **Servidor DNS** que se quiera utilizar.

El uso más **común** de esta herramienta sería:

***dnsenum******--dnsserver*** *‘ip\_servidor\_dns\_a\_consultar’ ‘nombre\_dominio\_objetivo’*

Si queremos realizar un ‘descubrimiento’ de Subdominios del Dominio Principal haciendo un ‘***Scraping’*** (*extraer información de un Sitio Web*) **en Google**, debemos utilizar las opciones ‘***-p***’ y ‘***-s***’, para especificar el numero de páginas de Google en las que va a buscar la herramienta y el máximo numero de Subdominio que queremos ‘extraer’, respectivamente:

***dnsenum******--dnsserver*** *‘ip\_servidor\_dns’ ‘nombre\_dominio\_objetivo’* ***-p*** *‘num\_paginas’* ***-s*** *‘num\_subdominios’*

**NOTA**:

El ‘***Google Scraping***’ puede que no pueda llevarse a cabo con éxito por que en ocasiones **Google** considera que esta técnica **viola sus términos de uso** y puede **bloquear** el acceso si intentamos realizar muchas consultas.

Si queremos realizar un ‘**descubrimiento’** de **Subdominios** del **Dominio** **Principal** a través de un ataque de **Fuerza Bruta**, debemos utilizar la opción ‘***-f***’ y especificar un ‘**Diccionario’** (*Fichero de Nombres*):

***dnsenum******--dnsserver*** *‘ip\_servidor\_dns’* ***-f*** *‘ruta\_diccionario\_nombres’ ‘nombre\_dominio\_objetivo’*

1. **WhatWeb**:

**WhatWeb es una herramienta** que identifica **Tecnologías Web** que han sido utilizadas para desarrollar un determinado **Sitio Web**. La ejecución básica de **WhatWeb** podría ser:

***whatweb -v*** *‘url\_objetivo’*

1. **WafW00f**:

La función principal de esta herramienta es identificar la **existencia de un Firewall en un Sitio o Aplicación Web** (***WAF****: Web Application Firewall*). Básicamente lanza una serie de **peticiones** y **métodos** **HTTP** (*request*) contra el objetivo y analiza las respuestas (*response*) de este para determinar la existencia o no de un ***WAF***. Su ejecución es muy simple:

***wafw00f******-v*** *‘url\_objetivo’*

Conocer la existencia y el tipo de un **WAF** en una **Web** facilita en posteriores fases del proceso de **Pentesting** el identificar posibles **vulnerabilidades** que este **WAF** tenga.

1. **The Harvester**:

Aunque la información que esta herramienta es capaz de ofrecernos va más allá de una **Lista de Direcciones de Email asociadas a un determinado Dominio**, es este, el objetivo final que se busca cuando la utilizamos:

***theharvester*** *-****d*** *‘dominio\_objetivo’* ***-l*** *‘limite\_resultados’* ***-b*** *‘fuente\_de\_busqueda’* ***-f*** *‘fichero\_destino’*

**NOTA**: Si no aparece ningún resultado, es aconsejable volver a ejecutar la misma sentencia pasados unos minutos, es probable que entonces si aparezcan algún resultado. Si después de haberla ejecutado un número razonable de veces, no aparece nada, es que no hay nada que mostrar.

1. **DIRB** (*Directory Buster*):

**Dirb** no es más que un **Escáner de Contenido Web**. Busca **Objetos Web** (Directorios, Ficheros, etc.) visibles (**u ocultos**) dentro de una determinada **URL**. Su funcionamiento es muy simple, realiza un **Ataque de Diccionario** contra el **Servidor Web** objetivo y analiza sus respuestas (*Ataque de Directorio Transversal/Directory Traversal*).

**NOTA**: **Directory Traversal**:

Consiste en explotar una vulnerabilidad que ocurre cuando no existe suficiente seguridad en cuanto a la validación de un Usuario, permitiéndole acceder a cualquier tipo de **Directorio Superior** sin ningún control y, por lo tanto, **tener acceso a ficheros a los que no debería acceder**.

**Dirb** incluye ***WordList*** (Diccionario) **propios**, pero puede utilizarse **cualquier otro** que tengamos, evidentemente, **cuanto más completo, mejor será el resultado**.

El funcionamiento de ‘**dirb**’ por defecto, utiliza el fichero ‘***common.txt***’ (/usr/share/wordlists/dirb) como ***WordList*** y es tan sencillo como:

***dirb*** *‘url\_objetivo’*

Además de este tipo de ejecución por defecto, podemos afinar un poco más el escáner para ajustarlo a nuestro objetivo, por ejemplo:

***dirb*** *‘url\_objetivo’* ***-X*** *‘.extension\_fichero’* ***-o*** *‘fichero\_salida.txt’* ***-N*** *‘codigo\_estado’*

En este caso, definimos la **extensión** de los **ficheros** que queremos localizar (*-X*), indicamos que el resultado del escaneo lo **guarde** en un **fichero** (*-o*) y que **ignore** las respuestas del **Servidor Web** que tengan un determinado **Código de Estado HTTP**(*-N*).

**Todo este tipo de herramientas tienen una gran cantidad de parámetros que vamos a poder utilizar para ajustar o mejorar nuestras búsquedas, por lo que es OBLIGATORIO, antes de ponerse a utilizarlas, echarle un vistazo a la AYUDA que todas suelen incluir y que nos va a facilitar su uso.**

1. **WFuzz**:

¿***Fuzzing***?: El ***Fuzzing*** es una técnica para las **pruebas de Software**, a menudo **automatizada** o **semi-automatizada**, implica proporcionar datos **inválidos**, **inesperados** o **aleatorios** a las entradas de un programa. Después se monitorizan los resultados que se van obteniendo.

**WFuzz** es una herramienta similar a **Dirb**, es decir, un **Escáner de Contenido Web** para la **búsqueda** de **Ficheros** y **Directorios**, ya sean **visibles** o **invisibles**.

Su funcionamiento está basado en un concepto muy sencillo: remplaza cualquier referencia a la ‘***keyword’*** **FUZZ** por el valor de un **Payload** dado. Un **Payload** en **WFuzz** no es más que una **fuente de datos de entrada** (*equivalente al* ***WordList*** *en* ***Dirb***). Y como pasa con **Dirb**, el éxito de un escaneo depende mucho del **Payload** utilizado.

Para realizar un escaneo ‘por defecto’ de un determinado objetivo con **WFuzz** basta con ejecutar:

***wfuzz******-c******-z******file,****’ruta\_del\_payload/wordlist’* ***--hc 404*** *‘url\_objetivo’****/FUZZ***

**b. Sitios Web Corporativos:**

Aunque pueda parecer una afirmación muy obvia, uno de los **primeros pasos** que debemos llevar a cabo a la hora de un **proceso de Recolección de Información** es pasar tiempo navegando por el **Sitio Web** del objetivo y ver si es posible obtener algún tipo de **información directa** **sólo navegando por sus contenidos**.

* **NOTA**: Ficheros **robots.txt** y **sitemap.xml**:

Cuando accedemos al **Sitio Web** objetivo, una buena práctica es intentar acceder a estos dos ficheros, que por lo general suelen existir, y **nos van a permitir conocer parte o toda la estructura de la Web, la visible y la ‘invisible’.**

Definido de forma simple, estos dos ficheros, **dan información de una Web a los Buscadores sobre que deben y que no deben ‘rastrear’** (*indexar*):

- ***sitemap.xml***: Lista de páginas que forman un **Sitio Web** (*Estructura*).

- ***robots.txt***: Indica a los Buscadores que partes de un **Sitio Web** se deben rastrear y **cuales no**.

**c. Google Hacking:**

**IMPORTANTE**:

**Para conocer el funcionamiento de GOOGLE HACKING se recomienda el libro ‘Hacking con Buscadores’ de la editorial 0xWORD.**

**Google** es uno de los **motores de búsqueda** más poderosos y efectivos que existen. Pero debido a ese poder, **puede causar la exposición de información sensible** en un **Sitio Web**. Gracias a las **Búsquedas Avanzadas**, podemos realizar búsquedas que nos revelen gran cantidad de información que posteriormente nos faciliten el trabajo en las siguientes fases del proceso de Pentesting.

Para realizar este tipo de **búsquedas avanzadas**, hacemos uso de los **Operadores de Google** que combinados de la forma adecuada pueden generar consultas **extremadamente útiles** (***Google Dorks***):

**OPERADORES de BUSQUEDA GOOGLE**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operador** | **Sintaxis** | | **Descripción** |
| ***or*** | | palabra1 ***or*** palabra2 | Muestra los resultados que contengan **cualquiera** de los términos especificados. |
| ***and*** | | palabra1 ***and*** palabra2 | Busca páginas que incluyan **todos** los términos especificados. |
| ***“ “*** | | ***“***cadena\_de\_busqueda***“*** | Muestra los resultados donde aparece **exactamente** la cadena de búsqueda especificada. |
| ***-*** | | ***-***palabra | Muestra los resultados donde se **excluye** el termino puesto detrás del guion. |
| ***\**** | | palabra1\*palabra2 | Un **comodín** que puede **sustituir** a cualquier termino. |
| ***#..#*** | | numero1..numero2 | Muestra resultados que se encuentran **entre** un **rango** de **cifras** específicas. |
| ***( )*** | | ***(***palabra1 or palabra2***)*** -palabra3 | Permite **combinar** varios **operadores** en una expresión. |
| ***around*** | | palabra1 ***around****(****num****)* palabra2 | Muestra resultados donde aparecen los términos especificados **combinados** con el número de **términos** **definidos** (*num*) **entre** **ellos**. |
| ***site*** | | ***site:***nombre\_dominio | Muestra resultados **existentes** **dentro** del **Sitio** especificado. |
| ***link*** | | ***link:***nombre\_dominio | Muestra resultados donde **existan** **enlaces** al **Sitio** especificado. |
| ***cache*** | | ***cache:***nombre\_dominio | Muestra la **copia del Sitio** especificado, almacenado en la **cache** de **Google**. |
| ***filetype*** | | ***filetype:***extension\_fichero | Muestra resultados que **contengan** **archivos** en el **formato** especificado. |
| ***allintext / intext*** | | ***allintext:***”palabra1 palabra2” | Muestra resultados que **incluyan** en sus **textos**, **alguno** o **todos** los términos especificados. |
| ***allintitle / intitle*** | | ***alintitle:***palabra1 palabra2 | Muestra resultados que **incluyan** en su **titulo**, **alguno** o **todos** los términos especificados. |
| ***allinurl / inurl*** | | ***inurl:***”palabra1/palabra2.ext” | Muestra resultados que **incluyan** en su **URL**, **alguno** o **todos** los términos especificados. |
| ***@*** | | **@**palabra | Busca **etiquetas** sociales asociadas a **Twitter**. |
| ***#*** | | **#**palabra | Busca los términos especificados, **dentro** de **hashtags** publicados en **Redes** **Sociales**. |
| **IMPORTANTE**: Para que los **operadores** con **dos** **puntos** (:) funcionen, es necesario **no dejar ningún espacio entre los dos puntos y el término que le sigue**. | | | |
| **Todos los operadores pueden combinarse entre si para construir búsquedas más especificas.** | | | |

**d. Plugins/Complementos del Navegador:**

Otra de las herramientas que podemos utilizar en la fase de **Recolección de Información** son los **Plugins**/**Complementos** que existen para los **Navegadores** y que permiten **extraer** **información** de los **Sitios Web** que están mostrando.

Uno de los **Plugins** que permiten esta recolección de información es **PassiveRecon**. Este complemento, al activarlo, nos abre una serie de pestañas que nos muestran información de todo tipo: WhoIs, Direcciones IP, Servidores DNS, Google Dorks para descubrir documentos, todo ello de forma automatizada.

**IMPORTANTE**:

**PassiveRecon** ya no esta disponible para las nuevas versiones de **Firefox**. Para poder utilizarlo, lo mejor es instalar **OWASP Mantra** dentro de nuestra distribución de **Kali Linux**.

**OWASP Mantra** es un **Framework** de seguridad implementado sobre un **Navegador** **Web** y diseñado específicamente para testear **Aplicaciones** **Web**, incluye muchas **herramientas** que permiten realizar **análisis básicos de seguridad**.

Para instalar **OWASP Mantra** en nuestra distribución de **Kali**, basta con ejecutar el siguiente comando:

***apt-get install owasp-mantra-ff***

Una vez instalado, cuando ejecutamos **Mantra**, se nos abrirá una ventana del **Navegador** que ya incorpora el **Plugin** de **PassiveRecon**, lo único que debemos hacer es situarnos en algún punto de la página que queramos analizar, pulsar el **botón derecho del ratón** y seleccionar ‘***PasiveRecon’*** dentro del menú desplegable que aparece.

Otros **Plugins**/**Complementos** que pueden resultar útiles para esta fase son:

|  |  |  |
| --- | --- | --- |
|  | **Nombre** | **Descripción** |
|  | FireBug | Permite **analizar**, **editar**, **monitorizar** y **depurar** el código fuente, **CSS**, **HTML** y **JS** de una página web de manera instantánea. |
|  | Live HTTP Headers | Permite ver la **información** contenida en los **Headers** (**Cabeceras**) de un **Sitio** **Web**. |
|  | Wappalyzer | Permite conocer las distintas **tecnologías** utilizadas en un **Sitio** **Web**, detecta, ***CMS’s***, ***Web* *Frameforks***, ***Server Software***, etc. |
|  | IP Address and Domain | Permite ver la información detallada sobre cada **Dirección IP**, **Nombre de Dominio** y **Proveedor** de un **Sitio** **Web.** |

**e. Metadatos:**

Los **MetaDatos** pueden definirse como “**aquellos datos que describen a otros datos**”, por ejemplo, las propiedades de un documento de **Office**, donde podemos encontrar información acerca de ese fichero.

Algunos ejemplos de **MetaDatos** pueden ser: quien ha creado el documento, cuando lo ha creado, donde se ha creado, etc. Una de las herramientas más utilizadas para la extracción y el análisis de **MetaDatos** es **FOCA**.

**IMPORTANTE**:

**Para conocer el funcionamiento de FOCA se recomienda el libro ‘Pentesting con FOCA’ de la editorial 0xWORD.**

Existen muchas herramientas para el análisis de MetaDatos a parte de FOCA, por ejemplo, **Metashield** **Analyzer**, que permite realizar una tarea similar a **FOCA,** pero **On-Line**.

**f. Ingeniería Social:**

La **Ingeniería Social** se define como el proceso mediante el cual trataremos de **engañar** a una persona o grupo de personas, **para que nos suministren información**.

**Técnicas para la Recolección de Información en una RED CORPORATIVA:**

Hasta el momento, todas las técnicas y herramientas que se han mostrado en este documento, iban orientadas a la **Recolección de Información** de un **Sitio** o **Página Web**, a partir de este punto, vamos a ver como podríamos realizar esta **Recolección de Información** en el caso de que nuestro objetivo no fuera un Sitio Web, sino un **Red Corporativa**.

**NOTA**:

Alguna de las herramientas que se han visto en puntos anteriores, también pueden utilizarse para la recolección de información de una Red Corporativa.

En el caso de una **Red**, cuando hablamos de un objetivo, estamos hablando de una **Dirección IP**, no de un **Servicio de Internet**, por lo que no podemos utilizar muchas de las herramientas que hemos visto hasta ahora. **Para recoger información de una Red Corporativa, dos de las herramientas más utilizadas son NMAP y WhireShark**.

La mejor forma de **comenzar** la fase de **Information Gathering** en una **Red**, es utilizar un **Escáner de Protocolos y Servicios**, que nos va a permitir saber que **equipos** ‘**están vivos**’ dentro de una **Red**. Uno de los más utilizadas es **NMAP**.

**NOTA**:

En este punto, sólo vamos a utilizar **NMAP** para poder saber que equipos de la red analizada están disponibles. Más adelante veremos más en detalle que es y como funciona **NMAP** y para que nos puede resultar útil (***Fase de Escaneo y Enumeración***)

Para conocer que **Hosts** de una **Red** se encuentran **activos**, podemos ejecutar la siguiente instrucción en **NMAP**:

***nmap*** *‘objetivo\_analisis’* ***-v***

**NMAP** nos permite **definir el objetivo** del análisis de diferentes **formas**:

a. **Dirección IP** (*Privada o Pública*): 192.168.1.1

b. **Rango de Direcciones IP**: 192.168.1.1-254

c. **Dominio** (*Local o Internet*): www.nombre\_dominio.net

d. **Subredes**: 192.18.1.0/24

Para conocer más en profundidad las formas de hacer ‘**Host Discovery**’ con **NMAP**, podemos ver la sección con el mismo nombre que existe en el [**Sitio Web**](https://nmap.org/book/man-host-discovery.html) de la herramienta.

Una vez hemos reconocido que **Hosts** se encuentran activos, podemos proceder con la **recolección de información**, para esta recolección, una de las herramientas más utilizadas es **WhireShark**.

**WHIRESHARK:**

**WhireShark** es un **Analizador de Paquetes de Red** (*Sniffer*) que permite **analizar el tráfico que circula por una Red** **en tiempo real**, la herramienta **intercepta el tráfico** y lo **convierte en un formato legible para las personas**.

**NOTA**:

Adjunto a este documento, se incluye la **Guía de Referencia Oficial** de WhireShark.

Dado el gran volumen de tráfico que atraviesa una red, una de las mejores **características** que ofrece es el **Filtrado de Paquetes**. **WireShark** ofrece dos tipos de **Filtros**, los **Filtros de Captura** y los **Filtros de Visualización**.

**Filtros de Captura de WhireShark** (*Capture Filters*):

Son los que se establecen para mostrar **solo los paquetes que cumplan los requisitos indicados por el filtro mientras se está realizando la captura**. Si **no se establece** ninguno, **WireShark** captura **todo el trafico** y lo presenta en pantalla.

Para aplicar un filtro, seleccionamos: ***Capture*** > ***Options***, una vez allí, en el **campo** ‘***Capture Filter***’, **introducimos** el **filtro** o **pulsamos** sobre el **botón** ‘***Capture Filter***’ para **seleccionar** alguno de los **filtros** **predefinidos**.

**SINTAXIS** y **EJEMPLOS** de **Filtros de Captura**:

Podemos **combinar** varios **filtros** utilizando los **operadores** de:

- **Negación**: *!* o ***not***.

- **Unión** o **Concatenación**: *&&* o ***and***.

- **Alternancia**: *||* o ***or***

|  |  |
| --- | --- |
| **Filtros Basados en Host** | |
| ***Sintaxis*** | ***Descripción*** |
| **host** *host* | Filtrar por **Host**. |
| **src host** *host* | Capturar por **Host** de **Origen**. |
| **dst host** *host* | Capturar por **Host** de **Destino**. |
| **Ejemplos** | |
| **host** *192.168.1.20* | Captura todos los paquetes con **Origen** y **Destino** ***192.126.1.20*** |
| **src host** *192.168.1.1* | Captura todos los paquetes con **Origen** en ***Host*** ***192.168.1.1*** |
| **dst host** *192.168.1.1* | Captura todos los paquetes con **Destino** en ***Host*** ***192.168.1.1*** |
| **dst host** *SERVER-1* | Captura todos los paquetes con **Destino** en ***Host*** ***SERVER-1*** |
| **host** *http://www.google.es* | Captura todos los paquetes con **Origen** y **Destino** ***http://www.google.es*** |
| **Filtros Basados en Puertos** | |
| ***Sintaxis*** | ***Descripción*** |
| **port** *puerto* | Captura los paquetes con puerto de Origen y Destino ***puerto***. |
| **src port** *puerto* | Captura todos los paquetes con puerto de Origen ***puerto***. |
| **dst port** *puerto* | Captura todos los paquetes con puerto de Destino ***puerto***. |
| **not port** *puerto* | Captura todos los paquetes excepto Origen y Destino puerto ***puerto***. |
| **not port** *puerto* **and not port** *puerto1* | Captura todos los paquetes excepto Origen y Destino puertos ***puerto*** y ***puerto1***. |
| **Ejemplos** | |
| **port** *21* | Captura todos los paquetes con puerto **Origen** y **Destino** ***21***. |
| **src port** *21* | Captura todos los paquetes con puerto **Origen *21***. |
| **not port** *21* **and not port** *80* | Captura todos los paquetes excepto **Origen** y **Destino** puertos **21** y **80**. |
| **portrange** *1-1024* | Captura todos los paquetes con puerto **Origen** y **Destino** en el **rango** de puertos **1** a **1024** |
| **dts portrange** *1-1024* | Captura todos los paquetes con puerto **Destino** en el **rango** de puertos **1** a **1024** |
| **Filtros Basados en Protocolos Ethernet / IP** | |
| ***Sintaxis*** | ***Descripción*** |
| **ip** | Captura todo el tráfico IP. |
| **ip proto** *\tcp* | Captura todos los segmentos TCP. |
| **ether proto** *\ip* | Captura todo el tráfico IP. |
| **ip proto** *\arp* | Captura todo el tráfico ARP. |
| **Filtros Basados en Red** | |
| ***Sintaxis*** | ***Descripción*** |
| **net** *red* | Captura todo el tráfico con Origen y Destino red ***red***. |
| **src net** *red* | Captura todo el tráfico con Origen red ***red***. |
| **dst net** *red* | Captura todo el tráfico con Destino red ***red***. |
| **Ejemplos** | |
| **net** *192.168.1.0* | Captura todo el tráfico con **Origen** y **Destino** la **subred** **192.168.1.0** |
| **net** *192.168.1.0/24* | Captura todo el tráfico para la **subred** **192.168.1.0** y **mascara 255.0** |
| **dst net** *192.168.2.0* | Captura todo el tráfico con **Destino** la **subred** **192.168.2.0** |
| **net** *192.168.2.0* **and port** *21* | Captura todo el tráfico **Origen** y **Destino** **puerto** **21** en la **subred** **192.168.2.0** |
| **brodcast** | Captura solo el tráfico Broadcast. |
| **not broadcast and not multicast** | Captura todo el tráfico excepto el **broadcast** y el **multicast**. |

**NOTA**: **Brodcast** y **Multicast**:

- Trafico **Broadcast**:

Es el tráfico que se genera cuando un dispositivo de la red (*Host*), envía paquetes **a TODOS los dispositivos de la red** (*Hosts*).

- Trafico **Multicast**:

Es similar al tráfico **Broadcast** sólo que en **Multicast** se envían paquetes **a un NUMERO ESPECIFICO de dispositivos de la red** *(Hosts)*.

**Filtros de Visualización de WhireShark** (*Display Filters*):

Establecen un criterio **sobre los paquetes capturados y que estamos visualizando** en la pantalla principal de **WireShark**. Estos filtros son **más Flexibles y Potentes**. También se utilizan para filtrar el contenido de una captura a través de un **fichero** *pcap*.

Para aplicar un filtro de visualización, basta con escribirlo en el **campo** ‘***Filter’*** que aparece en la parte superior de la pantalla principal de **WireShark**, encima de la pantalla que muestra los paquetes capturados. Si queremos aplicar otro filtro, pulsamos el **botón** ‘***Clear****’*, introducimos el nuevo filtro y pulsamos el **botón** ‘***Apply****’.*

**SINTAXIS** y **EJEMPLOS** de **Filtros de Visualización**:

Es posible utilizar un conjunto de **operadores de comparación** a la hora de definir un filtro:

- **Igual** a: ***eq*** *o ==.*

- **No igual**: ***ne*** *o ¡=*.

- **Mayor que**: ***gt*** *o >*.

- **Menor que**: ***lt*** *o <.*

- **Mayor o igual**: ***ge*** *o >=*.

- **Menor o igual**: ***le*** *o <=*.

También podemos utilizar los **operadores** vistos en los **Filtros de Captura** para combinar **Filtros de Visualización** (*Negación, Concatenación y Alternancia*).

Existe **otro operador** que podemos utilizar con este tipo de filtros, el operador ‘***Contains****’*, que realiza una **búsqueda por la cadena que pongamos a continuación** de este operador.

|  |  |
| --- | --- |
| **Ejemplos Filtros de Visualización** | |
| ***Sintaxis*** | ***Descripción*** |
| **ip.addr** == *192.168.1.40* | Visualizar el tráfico por ***Host*** ***192.168.1.40*** |
| **ip.addr** ¡= *192.168.1.25* | Visualizar todo el tráfico menos por ***Host*** ***192.168.1.25*** |
| **ip.dst** == *192.168.1.30* | Visualizar por ***Host Destino*** ***192.168.1.30*** |
| **ip.src** == *192.168.1.30* | Visualizar por ***Host Origen*** ***192.168.1.30*** |
| **ip** | Visualiza por todo el ***trafico IP***. |
| **tcp.port** == *143* | Visualiza todo el trafico ***Origen*** y ***Destino******puerto******143*.** |
| **ip.addr** == *192.168.1.30* **and** **tcp.port** == *143* | Visualiza todo el tráfico ***Origen*** y ***Destino*** ***puerto*** ***143*** relativo al ***Host 192.168.1.30*** |
| **http contains “***http://www.google.es***”** | Visualiza el tráfico ***Origen*** y ***Destino*** ***Google***. Visualiza los paquetes que ***contienen*** ‘*http://google.es*’ ***en el contenido en protocolo http***. |
| **frame contains “**@micorreo.es**”** | Visualiza todos los correos con ***Origen*** y ***Destino*** al **dominio** ‘*micorreo.es*’. |
| **icmp[0:1] == 08** | Visualizamos todo el tráfico ***icmp*** ***de tipo echo request***. |
| **ip.ttl** == *1* | Visualiza todos los ***paquetes IP*** cuyo ***campo TTL*** sea ***igual a 1***. |
| **tcp.windows\_size** ¡= *0* | Visualiza todos los paquetes cuyo ***campo*** ‘***Tamaño de Ventana***’ del ***segmento TCP*** sea ***distinto de 0***. |
| **ip.tos** == *x* | Visualiza todos los ***paquetes IP*** cuyo ***campo TOS*** sea ***igual a x***. |
| **ip.flags.df** == *x* | Visualiza todos los ***paquetes IP*** cuyo ***campo DF*** sea ***igual a x***. |
| **udp.port** == *43* | Visualiza todo el ***tráfico UDP*** por el ***puerto 53***. |
| **tcp contains “***google.es***”** | Visualiza ***segmentos TCP*** que contengan la ***cadena*** ‘***google.es***’. |

**NOTA: ENCAPSULACIÓN:**

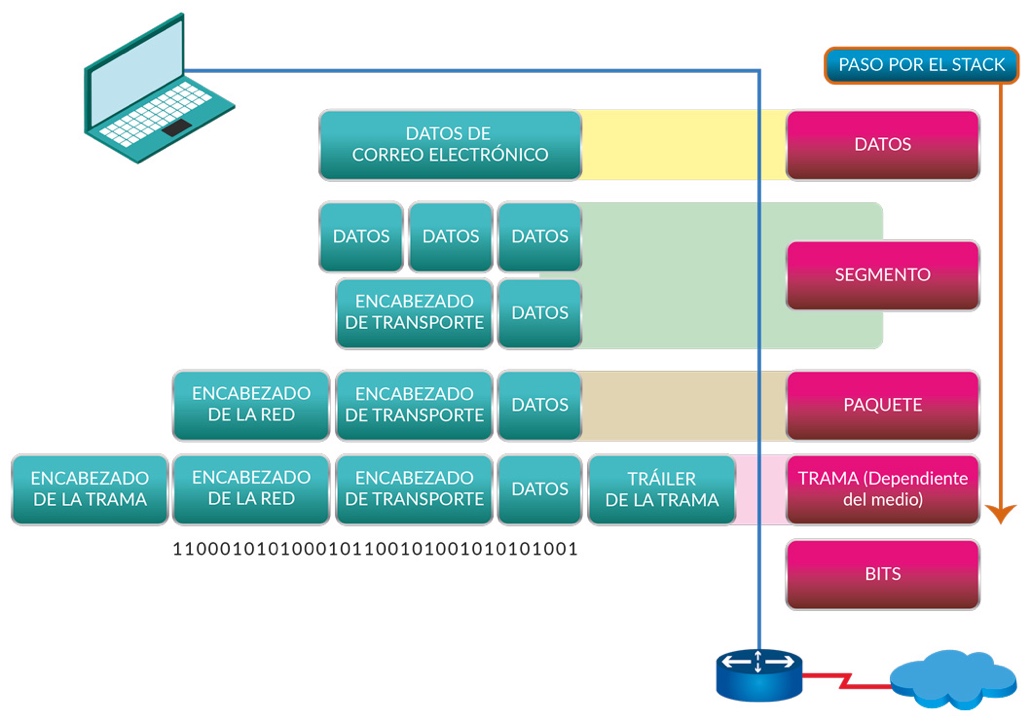
La **Encapsulación** es el proceso por el cual los datos son **convertidos** para que puedan **viajar** por la red y ser **interpretados** por los dispositivos de la red. Este proceso está relacionado con las diferentes **Capas del Modelo OSI** y consiste en **dar formato a los datos** y **agregar** la **información necesaria** a medida que pasan por **cada una de las capas** dependiendo del **Protocolo** que intervenga en ese momento.

La información que se agrega en cada una de las **Capas** se conoce como **Encabezado** (*Header*). El **Proceso de Encapsulación** consta de los siguientes pasos:

1. Los **Datos** son generados por el **Usuario** en la **Capa de Aplicación**, pasan a las **Capas de Presentación y Sesión** donde se les da el **formato especifico** y se les **agrega el Encabezado** correspondiente para ser enviados a la **Capa de Transporte**.
2. Esta **Capa de Transporte**, **divide** los **datos** recibidos en porciones más pequeñas para facilitar la transmisión. A **cada porción** se le añade el **Encabezado** correspondiente. **Cada porción se convierte en un Segmento** y es enviado a la **Capa de Red**.
3. La **Capa de Red** recibe cada **Segmento** y le **agrega** el **Encabezado** correspondiente. **Cada Segmento se convierte en un Paquete** que será enviado a la **Capa de Enlace de Datos**.
4. La **Capa de Enlace de Datos** recibe los **Paquetes**, les **agrega** su **Encabezado** y **convierte los Paquetes en Tramas** que se envían a la **Capa Física**.
5. La **Capa Física** recibe las **Tramas** y las **convierte en Bits**, que son enviados a través de la red hacia su destino.

***Datos*** > ***Segmentos*** > ***Paquetes*** > ***Tramas*** > ***Bits***

Una vez los **Bits** son recibidos por el **Host** receptor, comienza el **proceso de convertir esos Bits en Datos**, para ello, se comienza a **eliminar** los **distintos Encabezados** que corresponden a cada **Capa del Modelo OSI**. Este proceso se conoce como **Desencapsulación**.



**ETTERCAP:**

**NOTA IMPORTANTE:**

**Ettercap** es una **herramienta muy agresiva** por lo que **no se recomienda su utilización** o no ser que los responsables de los sistemas que vamos a auditar, nos den permiso para utilizarla. Debido a la gran cantidad de tráfico que genera la herramienta para realizar su trabajo, es posible que, al ejecutarla, realicemos sin querer un **Ataque de Denegación de Servicio** (*DoS/DDoS*) e inutilicemos el sistema auditado.

Otra herramienta que se puede utilizar para el **análisis del tráfico en una red** es **Ettercap**, la principal diferencia que existe entra **Ettercap** y **Wireshark**, es que mientras **Wireshark** es un analizador de paquetes ‘***pasivo’***, o **no intrusivo**, **Ettercap** puede ‘***cambiar****’* el **tráfico** de los paquetes de red y **dirigirlo** (*enrutarlo*) **a través de él** (***MitM****: Man in the Middle*), puede configurarse para **descartar**, **modificar** o **insertar** **paquetes** dentro de una comunicación en la red.

La **utilización básica** de **Ettercap** para ‘***capturar****’* el tráfico en una red sería el siguiente:

1. En el ***Menú Principal*** seleccionamos: ***Sniff*** > ***Unified Sniffing***

donde seleccionaremos la **Interfaz de Red** que queremos utilizar.

1. Después: ***Host*** > ***Scan for Hosts***

para que **Ettercap** realice un ‘***descubrimiento’*** de los ***Hosts*** disponibles en la red.

1. A continuación: ***Host*** > ***Hosts list***

para poder ver la lista de ***Hosts*** conectados. En este punto podemos realizar la **captura** de **todo** el **tráfico** de la red o seleccionar dos objetivos (***Add to Target 1*** y ***Add to Target 2***) y ‘***controlar’*** los paquetes que los dos dispositivos intercambiarán, es decir, un ***Ataque Man in the Middle***.

1. El siguiente paso: ***Mitm*** > ***ARP Poisoning***

donde marcaremos la opción ‘***Sniff Remote Connections***’ y aceptaremos.

1. Finalmente, en el mismo ***Menú Principal***: ***Start*** > ***Start Sniffing***

A partir de este momento estaremos capturando el tráfico de la red. Si seleccionamos la opción ***View*** > ***Connections***, del ***Menú Principal***, podremos ver todas las conexiones y pulsando sobre ellas, podremos ver los datos que contienen.

**Recolectar MAS INFORMACION de una RED CORPORATIVA: ESCANEAR UNA RED con NMAP:**

Antes de profundizar en **NMAP**, vamos a ver **algunos conceptos** que nos ayudarán a entender esta herramienta, fundamental en el **Proceso de Pentesting**.

**- Mapeo de una Red** (*Network Mapping*)**:**

Básicamente consiste en obtener una representación ‘grafica’ o en formato texto, de todos los ***Hosts*** y **dispositivos** en una red.

**- Escaneo de Puertos** *(Port Scanning)***:**

Consiste en el envío de **paquetes de red** (*sondas*) a un **equipo** o ***Host***, con el objetivo de **identificar el estado de sus puertos TCP y/o UDP**.

Los **estados de un puerto** en un **equipo** o **Hosts** pueden ser **seis** (*según* ***nmap***): *Abierto*, *Cerrado*, *Filtrado*, *No Filtrado*, *Abierto|Filtrado* y *Cerrado|Filtrado*. Aunque podríamos **resumirlos** **en tres**:

* **Abierto**:

Una **Aplicación** **acepta** **conexiones** **TCP**/**UDP** en este puerto. **Permiten identificar que Servicios están disponibles en un equipo o Host**, aunque un puerto abierto también es un ***Vector de Ataque*** por lo que es recomendable **cerrarlos** o **protegerlos**.

* **Cerrado**:

Un puerto cerrado **es accesible** (*recibe y responde a las* ***sondas***) pero **NO tiene una Aplicación** ‘***escuchando’*** **en el**.

* **Filtrado**:

**No se puede determinar** si el puerto se encuentra **Abierto** o **Cerrado** por que un **filtrado** **de** **paquetes** impide que las ‘***sondas’*** alcancen el puerto. El **filtrado** puede venir de un **Firewall**, las **Reglas del Router**, **Antivirus**, etc.

Los **objetivos** principales de un **Escáner de Puertos** son:

- Detectar **Sistemas** ‘**Vivos**’.

- Descubrir **Puertos Abiertos** con **Programas**/**Servicios** en **ejecución**.

- Descubrir **S.O. de los Host** (*OS Fingerprinting*)

- Descubrir **Direcciones IP** de la Red.

- Identificar **Banners**.

Existen **varias técnicas** para **Escanear Puertos**, que varían según las necesidades y habilidades del auditor, y que se veremos con **NMAP**.

**- Intercambio de Tres Vías del Protocolo TCP:**

Antes de conocer cada una de las técnicas para escanear puertos, es necesario comprender **cómo se establece una conexión entre dos Hosts en la red**.

El **Protocolo TCP**, es un protocolo **orientado a conexión**, y **requiere que se establezca una conexión** **antes** de que comience la **transferencia de datos** y que los dos **Host** **se sincronicen**. Esta **sincronización** se realiza mediante el **Intercambio de Tres Vías**:

1. *Host* ***Origen*** inicializa conexión enviando un **Paquete SYN** (*Synchronize*) hacia *Host* ***Destino***.
2. *Host* ***Destino*** responde enviando un **Paquete SYN/ACK**.
3. *Host* ***Origen*** responde a *Host* ***Destino*** con un **Paquete ACK** (*Acknowledge*), completando la conexión.

- **Bits de Control o *Flags* de un Paquete TCP**:

Un **paquete TCP** (*Datagrama*) contiene ***6 Flags*** o ***Bits de Control*** que permiten **controlar la transmisión de los datos a través de una conexión**: *SYN*, *ACK*, *FIN*, *RST*, *PSH*, *URG,* donde*:*

1. **SYN**, **ACK**, **FIN** y **RST**: se emplean para el **establecimiento**, **mantenimiento** y **finalización** de la conexión:
   1. ***SYN*:** Inicia la conexión entre *Hosts*.
   2. ***ACK***: Reconoce la recepción de un paquete.
   3. ***FIN***: No habrá mas transmisiones.
   4. ***RST***: Resetea y aborta la conexión.
2. **PSH** y **URG**: dan **instrucciones** al **sistema**:
   1. ***PSH*:** Indica al *Host Receptor* que debe pasar la información a la **Capa de Aplicación** lo más rápido posible.
   2. ***URG***: Indica al *Host Receptor* la existencia de información urgente en el flujo de datos.

**ESCANEAR UNA RED CON NMAP**:

Una vez vistos los conceptos anteriores pasaremos a ver los distintos tipos de **Escáner de Puertos** y como pueden ejecutarse con **NAMP**:

a. *TCP* ***Connect*** *Scan*:

*nmap* ***-sT*** *‘especificar\_objetivo’ -v*

b. *TCP* ***SYN*** *Scan (suele ser el más utilizado por el poco ‘****ruido’*** *q genera)*:

*nmap* ***-sS*** *‘especificar\_objetivo’ -v*

c. TCP ***FIN*** *Scan*:

*nmap* ***-sF*** *‘especificar\_objetivo’ -v*

d. *TCP* ***Xmas Tree*** *Scan*:

*nmap* ***-sX*** *‘especificar\_objetivo’ -v*

e. *TCP* ***Null*** *Scan*:

*nmap* ***-sN*** *‘especificar\_objetivo’ -v*

f. ***UDP*** *Scan*:

*nmap* ***-sU*** *‘especificar\_objetivo’ -v*

g. ***Ping*** *Scan*:

*nmap* ***-sP*** *‘especificar\_objetivo’ -v*

A todos estos **Tipos de Escáner**, es posible añadirles más **parámetros** para **mejorar** **su** **eficiencia**, por ejemplo, partiendo del ***TCP SYN Scan*** (*-sS*), que será el más utilizado, podemos añadir otros parámetros como:

|  |  |
| --- | --- |
| ***Parámetro / Opción*** | ***Descripción*** |
| ***- Pn*** |  |
| ***- f***  ***- - mtu <****numero\_paquetes****>*** | Para **Fragmentar** los paquetes (*sondas*) enviados y ***Evitar*** la posible existencia de un **Firewall**/**IDS** |
| ***- sV*** | Para determinar ***Información*** sobre el ***Servicio****/****Versión*** que este corriendo en el puerto. |
| ***- sC*** | Incluye en el escaneo la ejecución de un conjunto de ***Scripts*** para potenciar la obtención de información. |
| ***- -traceroute*** | Descubre la ***Ruta*** seguida por los paquetes hasta el ***Host***. |
| ***- O*** | Para determinar el ***S.O.*** del ***Host***. |
| ***- A*** | Para realizar un ***Escaneo*** ***Agresivo***, agrupa ***-O***, ***-sV***, ***-sC*** y ***--traceroute*** |
| ***- T <****num\_plantilla****>*** | Permite definir como de ***Agresivos*** queremos ser (*de 0 a 5*). |
| ***- p <****rango\_de\_puertos****>*** | Escanea los ***Puertos*** ***Especificados*** (recomendable especificar **TODOS** ellos para evitar pasar algo por alto, ***1-65535***). |
| ***-oX <****fichero****>*** | **Guarda** el **Resultado del Escáner** en el fichero ***XML*** especificado. |

A modo de resumen de lo anterior, a continuación, se presentan **dos ejemplos** de ejecuciones de **NMAP** que pueden darnos buenos resultados:

***nmap******-sS*** *-****sV******-O*** *‘especificar\_objetivo’* ***-v******-p1-65535***

Esta segunda opción es bastante **mas agresiva** que la anterior por lo **solo debe utilizarse con el consentimiento de la persona/empresa auditada**. Además, a parte del **Escáner de Puertos**, incluye la ejecución de un conjunto de ***Scripts*** que potencia el análisis.

***nmap -A -T4*** *‘especificar\_objetivo’* ***-v -p1-65535***

**NOTA**:

En el siguiente [***enlace***](https://nmap.org/book/man-briefoptions.html)puede encontrarse un ***Listado******de******Parámetros****/****Opciones*** que soporta **NMAP**. También es posible acceder a este ***Listado de Opciones*** **ejecutando** la herramienta **NMAP** **sin argumentos**.

**NMAP CON SCRIPTS** (***NSE***: *NMAP Scripting Engine*):

**NMAP** incorpora un **Motor de Scripts** que ha permitido **aumentar** **su** **funcionalidad** en todos los aspectos, permite a los usuarios utilizar **scripts** para automatizar una amplia variedad de tareas.

Los usuarios podemos utilizar alguno de los muchos **scripts que NMAP incluye por defecto** o **crear nuestros propios scripts personalizados** *(Programados con LUA)*.

Los ***Scripts*** incluidos en **NSE** están divididos en **15 categorías**:

* ***auth***: contiene scripts para la **Autenticación de Usuarios**.
* ***broadcast***: contiene scripts para el **Descubrimiento de Hosts** *(con peticiones broadcast)*
* ***brute***: contiene scripts para **Ataques** **de** **Fuerza Bruta**.
* ***default***: contiene un **conjunto de scripts** **que se ejecutan por defecto** *(\*)*.
* ***discovery***: contiene scripts para el **Descubrimiento** **de** **Hosts** y **Servicios**.
* ***dos***: contiene scripts para **Denegación de Servicios** *(DoS).*
* ***exploit***: contiene scripts para la **Explotación de Vulnerabilidades**.
* ***external***: contiene scripts que **se combinan con** **Servicios**/**Aplicaciones** **Externas** para su ejecución.
* ***fuzzer***: contiene scripts para hacer **Fuzzer**.
* ***intrusive***: contiene scripts **Intrusivos**. **Alto riesgo de** “*tumbar*” **el objetivo** durante su ejecución.
* ***malware***: contiene scripts para la **Detección de Malware** en el objetivo.
* ***safe***: contiene scripts **Seguros.**
* ***version***: contiene scripts para la **Detección de Versiones Avanzada**.
* ***vuln***: contiene scripts para la **Detección de Vulnerabilidades**.

Existen distintos métodos de **ejecutar** el **NSE** de **NMAP** a través de la **Línea de Comandos**:

*nmap* ***-sC*** *‘especificar\_objetivo’*

El parámetro ‘***-sC***’ pone en ejecución la **categoría** ***default*** *(\*)*, es equivalente a ***--script default***

*nmap* ***--script <fichero>|<categoría>|<directorio>|<expresión>[,…]*** *‘especificar\_objetivo’*

Ejecuta un escaneo utilizando los **scripts incluidos** dentro de un ***fichero***|***categoría***|***directorio*** **individual** o una **lista** **de** ***ficheros***|***categorías***|***directorios***, separados por comas (,).

**Ejemplos**:

*nmap --script* ***default****,****safe*** > Ejecuta los scripts incluidos en las categorías ***default*** y ***safe***.

*nmap --script* ***smb-os-discovery*** > Ejecuta **solo** el script **smb-os-discovery**.

*nmap --script* ***default,banner,/user/misscripts*** > Ejecuta scripts de categoría ***default***, el script ***banner*** y del ***directorio***.

*nmap --script* ***“http-\*”*** > Ejecuta **todos** los scripts cuyo **nombre comience con ‘http-’**.

**Otros parámetros** útiles que podemos utilizar cuando utilizamos scripts con **NMAP** son:

|  |  |
| --- | --- |
| ***Parámetro*** | ***Descripción*** |
| ***--script-args*** *<argumentos>* | Permite definir **Argumentos de Entrada** para el script. |
| ***--script-args-file*** *<fichero>* | Permite definir **Argumentos de Entrada** para el script **mediante** fichero. |
| ***--script-help*** | Muestra la **Ayuda** relacionada con el script especificado. |
| ***--script-updatedb*** | Permite **Actualizar la BD** de Scripts. |

**IMPORTANTE: CONSEJOS de USO de MNAP:**

a. **Nunca** utilizar los **DNS** de nuestro **ISP** (*Internet Services Provider*), utilizar los **DNS** de **Google**:

*nmap* ***--dns-servers 8.8.8.8,8.8.4.4*** *‘especificar\_objetivo’*

b. **Probar** con diferentes **Configuraciones de Tiempo** (*-T*):

*nmap* ***-T5*** *‘especificar\_objetivo’*

c. Ante cualquier cosa **Sospechosa**:

*nmap* ***-d*** *‘especificar\_objetivo’.*

d. Cuando se utilicen **Scripts HTTP**, **siempre** cambiar nuestro **User-Agent**:

*nmap -p80* ***--script http-enum*** *‘especificar\_objetivo’*

**NOTA**:

Las ejecuciones de **NMAP con Scripts** anteriores, pueden incluir otras de **Opciones**/**Parámetros** para afinar su funcionamiento (*aunque en los ejemplos no aparezca ninguna*)

**NOTA**:

Para conocer más en profundidad el **Nmap Scripting Engine** y como funciona en detalle podemos ir al **Sitio Web Oficial de** [**NMAP**](https://nmap.org/book/nse.html).

# **4.- Anonimato:**

Un concepto que siempre debemos de tener en cuenta a la hora de realizar esta **Primera Fase** del **Proceso de Pentesting**, la **Recolección de Información**, es el **ANONIMATO,** vamos a utilizarlo principalmente como una **medida de seguridad** **para** **nosotros**.

Cuando se habla de **anonimato en un entorno de red**, es obligatorio hacer referencia a dos conceptos:

1. La **Red** **Tor.**
2. Las **Redes** **VPN**.

**CONCEPTOS** necesarios para entender la ***RED TOR***:

* ***Clearnet*** / ***Surface Net***: El Internet que Conoces.

El **Internet** tal cual lo conocen la mayoría de los usuarios, ese pedazo de la ***World Wide Web*** a la que cualquiera puede acceder fácilmente desde cualquier navegador.

Se trata de una red en la que somos **fácilmente rastreables a través de nuestra Dirección IP**.

La componen principalmente la **Páginas Indexadas por los Buscadores Convencionales** (*Google, Bing o Yahoo*), pero **también** todas esas **otras** **Webs** a las que puedes acceder **de forma pública sin estar Indexadas** (*Facebook, Twitter, etc.*), así como **cualquier otro Sitio Web** o **Blog**.

* **Deep Web**: Las Profundidades de la World Wide Web.

Si de forma general, la **Clearnet** es esa porción de **Internet** a la que puedes acceder fácilmente con un navegador, podemos decir que la **Deep Web viene a ser justo lo contrario.** También conocida como **Invisible Web** o **Hidden Web**, **agrupa toda esa información que está online, pero a la que no se puede acceder de forma pública**.

Por un lado, puede tratarse de **páginas convencionales protegidas con un paywall**, pero **también** **archivos** guardados en **Dropbox** o **emails** guardados en los **Servidores** de nuestro proveedor.

La **Deep Web** también la componen los **Sitios Web** con un ‘***Disallow****’*en el archivo ‘***robots.txt***’ o **páginas dinámicas** que se generan al consultar una base de datos (*consultas bancarias, de viajes, etc.*), son **temporales** y **no las Indexa ningún buscador**.

* **Dark Web**: El Internet de las Profundidades.

Muchas veces **confundida con la Deep Web**, aunque **forma parte de ella**, la **Dark Web** es ese **fragmento de Internet al que sólo se puede acceder mediante aplicaciones específicas**.

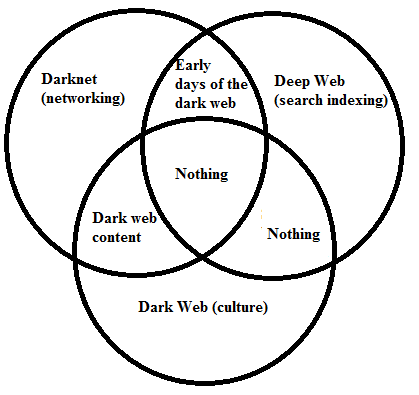
Puede definirse como “**la porción de internet que esta intencionadamente oculta a los Motores de Búsqueda, utiliza Direcciones IP enmascaradas y es accesible únicamente con un Navegador Web especial**”.

Está formada por **Sitios Web** **a los que no puedes acceder a no ser que dispongas del software necesario para navegar por las Darknets en las que se alojan**.

* **Darknets**: Las Redes Independientes que Componen la Dark Web.

Mientras la **Dark Web** es todo ese **contenido deliberadamente oculto** que nos encontramos en **Internet**, **las Darknets son esas redes especificas** (*TOR, I2P*) **que alojan esos Sitios Web**.

Aunque existen **Darknets** conocidas como: ***Freenet***, ***I2P*** o ***ZeroNet***, **la más popular de ellas es *TOR***.



**LA RED TOR** *(The Onion Router)*:

Es posiblemente la ***Darknet*** más conocida de **Internet**, el objetivo de este proyecto es crear una red de comunicaciones **distribuida** y **superpuesta** al **Internet** convencional. Los **Sitios Webs** que se pueden encontrar en la ***Darknet*** **de** **TOR** se diferencian por tener el dominio ‘.***onion****’.*

**¿Cómo funciona la Red TOR?:**

Implementa una técnica denominada ***Onion Routing*** que consiste en **enviar la información por un camino no directo utilizando diferentes Nodos** *(Relays)* **seleccionados de forma aleatoria por el *Software TOR* entre todos los disponibles en la Red TOR**.

Podemos dividir el proceso en **3 partes**:

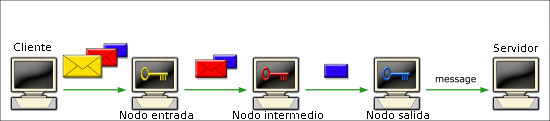
1. **Establecer** la **Ruta** a seguir para enviar/recibir una información:

Nodode **Entrada** (*Guard Relay*),Nodos **Intermedios** (*Middle Relays*),Nodode **Salida** (*Exit Relay*)

1. **Cifrar** toda la información que se quiere enviar/recibir.
2. **Descifrar** de forma **Progresiva** toda la información que queremos enviar/recibir.

**Método de Cifrado de la Red TOR**:

Una vez establecida la **ruta**, se utiliza un **Cifrado Asimétrico** (*o* *de Clave Publica*) para cifrar toda la información. Antes de entrar en el Nodo de Entrada, la información se Cifra Por Capas del siguiente modo:



1. Se utiliza la **Clave Pública** del **Nodo de Salida** para cifrar la información, de esta forma se añade una **Capa de Cifrado** y aseguramos que **sólo el último nodo podrá descifrar el mensaje**.
2. Al mismo tiempo, se usa la **Clave Pública** del **Nodo Intermedio** y de este modo se aplica una **segunda Capa de Cifrado** al mensaje.
3. Finalmente se utiliza la **Clave Pública** del **Nodo de Entrada** para añadir una **tercera Capa de Cifrado** al mensaje.

Una vez finalizado el cifrado, el mensaje se envía al **Nodo de Entrada** y empieza la **transmisión** y **Descifrado Progresivo** de la información.

**Proceso Progresivo de Descifrado:**

1. Una vez en el **Nodo de Entrada**, usando su **Clave Privada**, se elimina la **primera de las Capas de Cifrado**, el mensaje aun tendrá **dos Capas de Cifrado** por lo que **no será posible que el Nodo de Entrada** pueda ver su contenido.
2. Después la información se dirige al **Nodo Intermedio** donde, haciendo uso de su **Clave Privada**, se elimina la **segunda Capa de Cifrado**.
3. Finalmente, en el **Nodo de Salida**, se utiliza su **Clave Privada** para eliminar la **última Capa de Cifrado** y dejar la **información sin cifrar**. Después la información es entrega al **Servidor.**

**NOTA**:

Cifrado **SIMETRICO**:

La información se cifra con **UNA CLAVE COMPARTIDA UNICA** para todo el mundo.

Cifrado **ASIMETRICO**:

Se utilizan **DOS CLAVES DIFERENTES**, La **Clave Publica**, que está a disposición de todo el mundo y la **Clave Privada** que es personal.

**¿Cómo usar TOR para navegar?:**

La **Red TOR** dispone de un **Navegador Propio** (*Tor Browser*) que nos permite conectarnos. El **Navegador Tor** incluye el **Buscador DuckDuckGo**.

**NOTA**:

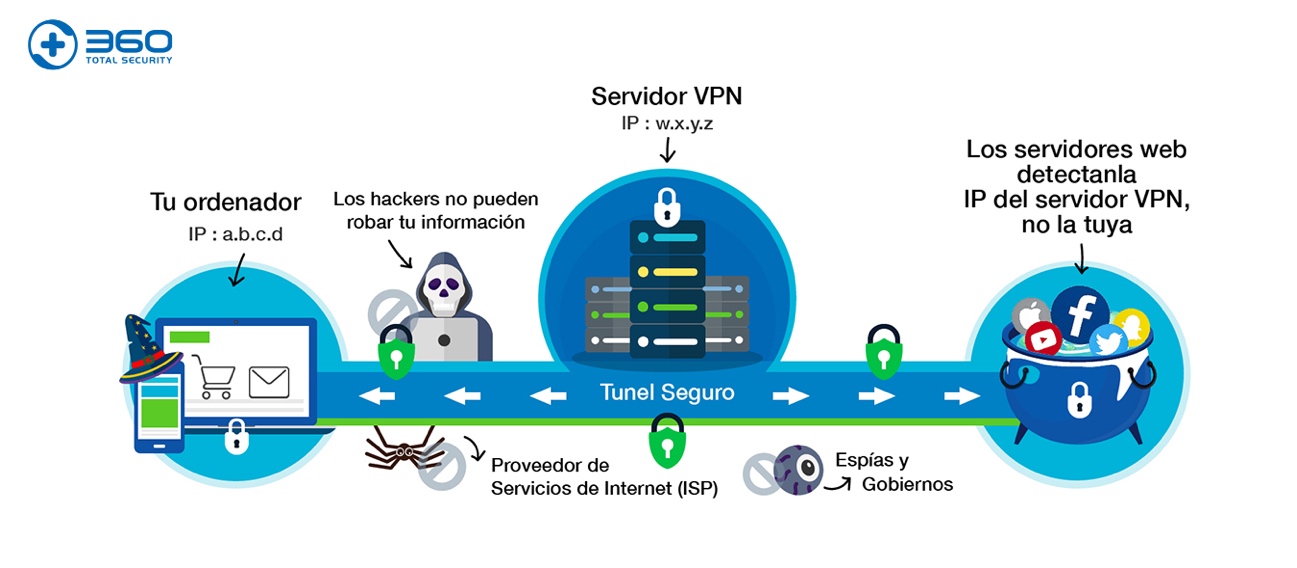
Es normal que los **Sitios Web** de la **Red TOR** **carguen extremadamente lentas**, algo que se debe al **cifrado de la red**.

**LAS REDES VPN** *(Virtual Private Network):*

# **Tecnología de red** que va a permitir **extender nuestra Red Local** *(LAN)* sobre una **Red Pública** (Internet). **Permite** que nuestro equipo **envía** y **reciba** **datos** a través de **Redes Compartidas** o **Públicas** como si fueran una **Red Privada**.

# Esto se realiza estableciendo una **Conexión Virtual Punto a Punto** que nos va a permitir trabajar con seguridad. De forma sencilla, podemos decir que se crea un **TUNEL VIRTUAL PRIVADO** ‘**encima’** **de** **Internet** **que va a conectarnos con otro Host o con Internet**.

Básicamente, **una Red VPN coge nuestra conexión, la encripta y la envía a otro Servidor (*Servidor VPN*) y será este Servidor VPN el que realice las conexiones por nosotros.**



# **5.- Análisis de Vulnerabilidades:**

Es la ***Segunda***/***Tercera*** **Fase del Proceso de Pentesting** y tiene como **objetivo**, **identificar si el sistema auditado/analizado es débil o susceptible de ser afectado/atacado** de alguna forma, es decir, es **VULNERABLE**.

Un **Análisis de Vulnerabilidades** puede dividirse en el análisis de los siguientes **puntos**:

1. Identificación y Análisis de vulnerabilidades en ***Aplicaciones*** (*según sus* ***versiones***) y ***S.O***.
2. Identificación y Análisis de los ***Parches de Seguridad*** (*Patch Management*).
3. Identificación y Análisis de vulnerabilidades ***Tecnológicas*** y ***Humanas*.**
4. Identificación y Análisis de ***Configuraciones por Defecto***.
5. Identificación y Análisis de vulnerabilidades ***Técnicas*** *y* ***Funcionales***.

Podemos diferenciar **3 Clases Generales** de **Vulnerabilidades en el Software**:

1. De **Uso**, como **Configuraciones** de **Usuario** o **Por** **Defecto** de los fabricantes *(IoT).*
2. De **Implementación** a causa de **Errores** en la **Codificación** *(Buffer Overflow, Inyecciones SQL, XSS, etc.)*
3. De **Diseño** a la hora de establecer la **Arquitectura** o un **Protocolo** del **Software**.

Las Vulnerabilidades pueden encontrarse en **todo tipo de *Dispositivos Físicos***, ***Software*** y en los ***Seres Humanos***.

Algunos aspectos importantes a la hora de **Iniciar un Análisis de Vulnerabilidades** serían:

1. Como las herramientas de Análisis suelen estar basadas en ***Plugins***, es **fundamental mantenerlos actualizados**.
2. **Configurar** de forma adecuada estas **herramientas** (*Perfil del Análisis*) **de acuerdo a la Información Recolectada** en fases anteriores.
3. La **experiencia del Auditor.**
4. Un **Análisis de Vulnerabilidades** forma pate del **Proceso Secuencial del Pentesting** (*después de la* ***Recolección de Información*** *y el* ***Escaneo y Enumeración***) por lo tanto, la **adecuada información** recogida en las dos **etapas anteriores**, será de **gran ayuda para lograr el éxito** y **no generar falsas alarmas** en el Cliente.

Existen muchas clasificaciones distintas de **Vulnerabilidades** (*antes hemos visto una* *de ellas*), pero si nos fijamos en la **Gravedad** que esta implica para el objetivo, podemos clasificarlas como: ***Bajas***, ***Medias***, ***Altas*** y ***Críticas***:

1. ***Bajas***:

Las vulnerabilidades **más débiles**, son las que **menos** **afectan** a un sistema o aplicación y **menos** **impacto** tendrán. Se les puede hacer frente **fácilmente**. **Se suele ignorar la ‘*lectura’* de este tipo de vulnerabilidades y esto es un error grave, en ocasiones, dan información muy relevante**.

1. ***Medias/Moderadas***:

También es **fácil** **de atajar**, aunque tendrán un **mayor impacto** que las anteriores ya que podrían **ser la base** **de otras vulnerabilidades más críticas**. Las consecuencias negativas de estas debilidades se pueden **reducir** de **forma** **sencilla**.

1. ***Altas***:

Más **peligrosas**, se aprovechan para **atacar rápidamente** un sistema o aplicación. El mayor impacto se encuentra en la **perdida de confidencialidad** de los **datos** o **recursos** y la **integridad** de estos.

1. ***Críticas***:

Las **peores vulnerabilidades que existen** y las que pueden traer m**ayores consecuencias negativas** al sistema. Requieren de la **evaluación** y **corrección** de forma **inmediata**.

**NOTAS: TOP VULNERABILIDADES WEB OWASP:**

Podemos ver un listado de las Vulnerabilidades más importantes que se producen en un Sitio o Aplicación Web en [***OWASP Top Ten Project***](https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project)***.***

**HERRAMIENTAS para el ANALISIS DE VULNERABILIDADES:**

Alguna de las **herramientas** que se pueden utilizar para el análisis de vulnerabilidades son:

1. ***Acunetix*** ([*www.acunetix.com*](http://www.acunetix.com/)):
   * Escáner de vulnerabilidades **orientado** a ***Aplicaciones Web y API’s***.
   * **Alto Rating de detección**, en torno a **4500** **tipos** de vulnerabilidades (*XSS, SQL Injection, etc.*).
   * Escáner de puertos.
   * Programación de análisis.
   * Informes y Reportes.
   * Multiplataforma.

**NOTA IMPORTANTE:**

***Acunetix*** es uno de los programas más potentes para el **análisis de vulnerabilidades,** pero existe una ***Suite Especial***, [***Acunetix* *Manual Pen Testing Tools***](https://www.acunetix.com/vulnerability-scanner/free-manual-pen-testing-tools/)**,** gratuita, que incluye un **conjunto de herramientas** que puede ser de mucha utilidad.

1. ***Nessus*** ([*www.tenable.com*](https://es-la.tenable.com/products/nessus)):
   * Escáner de **vulnerabilidades** y de **puertos** (*Host y Red*).
   * Interfaz Web.
   * Actualización diaria.
   * Informes y Reportes.
   * Multiplataforma.
   * Versión gratuita.

Aunque ***Acunetix*** y ***Nessus*** son las dos herramientas más utilizadas para este tipo de análisis, **existen muchas otras** que también pueden resultar muy útiles:

* ***Zap*** (*Kali Linux*), ***BurpSuite*** (*Kali Linux*), ***Nikto*** (*Kali Linux*), ***CMSMap*** *(WordPress, Joomla y Drupal)*

**OWASP ZAP**:

***OWASP Zap*** es un **Escáner de Vulnerabilidades** **gratuito** que puede servirnos en el caso de que no podamos acceder a una licencia de **Acunetix** o **Nessus**. Evidentemente, la potencia de ***Zap*** no es comparable con la que tienen las herramientas de pago, pero puede sernos de utilidad.

Debido a su **Arquitectura de *Proxy***, Zap puede **capturar** **peticiones** y **respuestas** **HTTP**/**HTTPS** (*request y response*), tanto ***GET*** como ***POST***. Para ello debemos **configurar el Navegador** que vayamos a utilizar para que mande todas esas peticiones a Zap, este las reenvíe al **Servidor Web** y pueda recoger las respuestas de ese **Servidor Web**, para enviarlas al **Cliente** (*es decir, utilice un Proxy*).

Ahora Zap esta preparado para interceptar todas las peticiones del **Navegador** y todas las respuestas del **Servidor Web**.

Podemos ver un **listado de peticiones** y **respuestas** capturadas por **Zap** a medida que se van produciendo en la **Pestaña** ‘*Historia’*. Para cada **Petición Web** capturada podremos ver:

|  |  |
| --- | --- |
| ***Titulo*** | ***Descripción*** |
| **Timestamp** | **Fecha** y **Hora** de la captura. |
| **Método** | Método empleado, **GET** o **POST**. |
| **URL** | URL donde se ha mandado la Petición. |
| **Code** | **Código** **de** **Respuesta** devuelto por el **Servidor** tras procesar la Petición. |
| **RTT** | ***Tiempo que tarda un paquete de datos enviado desde el emisor en volver al mismo emisor habiendo pasado por el receptor de destino***. |
| **Size Response Body** | **Tamaño** del **Cuerpo** de la **Respuesta** del Servidor. |
| **Alert** | Tipo de Alerta detectada: **Alta**, **Media**, **Baja** o **Informativa**. |
| **Tags** | Etiquetas relacionadas con la posible vulnerabilidad detectada. |

**Función Escáner Pasivo**:

Una de las funcionalidades de Zap es la **Monitorización** e **Inspección** **Pasiva** **del** **Trafico** **HTTP**/**HTTPS** (*Escáner Pasivo*), es decir, Zap **no envía nuevas peticiones a la Aplicación Web** (*lo que sería un Escáner Activo*). **Intercepta y almacena las peticiones y respuestas HTTP/HTTPS durante la navegación**, para su análisis en busca de vulnerabilidades.

Zap incluye un ***Conjunto de Reglas*** para su **Escáner Pasivo** que podemos ver en: *‘****Herramientas*** *>* ***Opciones*** *> Passive Scan Rules’.*

Ya se ha comentado que Zap es un **Proxy de Intercepción de Peticiones y Respuestas Web**. Esto implica que, **además de capturar las Peticiones GET y POST** que se mandan al **Servidor Web**, **es posible reenviarlas e incluso modificarlas ‘***al vuelo***’ cambiando el valor de sus parámetros antes de mandarla de nuevo al Servidor Web**.

**NOTA**:

**Mensajes HTTP/HTTPS**: *Peticiones y Respuestas*

Los **mensajes HTTP** permiten el **intercambio de datos** entre los **Servidores** y los **Clientes**, existen **2 tipos** de **mensajes HTTP**: Las **Peticiones** (*Request*), enviados **desde el Cliente al Servidor** para solicitar el **inicio de una acción**, y las **Respuestas** (*Response*), que son la **respuesta del Servidor**.

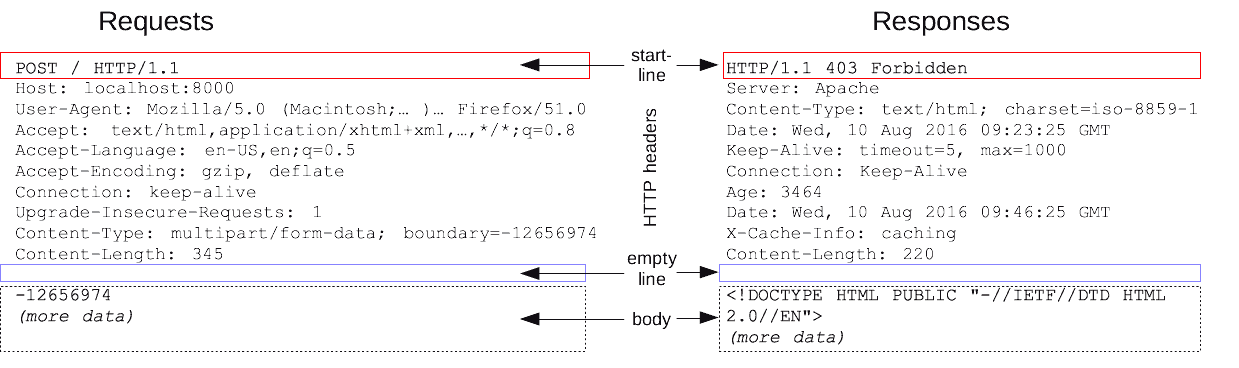
Las **Peticiones** y **Respuestas** **HTTP**, comparten una **estructura** **similar**, compuesta de:

1. **Línea Inicial**: Describe, o la petición que se quiere implementar (*Cliente* ***>*** *Servidor*) o la respuesta con su estado (*Servidor* ***>*** *Cliente*), sea de **Éxito** o **Fracaso**. Es siempre una sólo línea.

2. **Cabeceras HTTP** (*HTTP Headers*): Opcionales, contienen Información Adicional acerca de la Petición o Respuesta.

3. **Línea Vacía**.

4. **Cuerpo del Mensaje**: Opcional, incluye **Datos** asociados a la **Petición** (*Contenido de un Formulario HTML, por ejemplo*), o los archivos o documentos asociados a la Respuesta (*una Página HTML, un Archivo de Audio, Un Documento PDF, etc.*).



**Las Cabeceras HTTP** (*HTTP Headers*):

Su **estructura** (*formato*) consiste en una **Cadena de Caracteres** (*que no diferencia entre mayúsculas y minúsculas*), **seguida por Dos Puntos** (*:*), y un **Valor** cuya estructura dependerá de la Cabecera. La Cabecera completa, incluido su valor, debe ser de **una sola línea** y puede ser bastante larga.

Existen bastantes **Cabeceras HTTP** y podemos clasificarlas en varios grupos:

a. **Generales**:

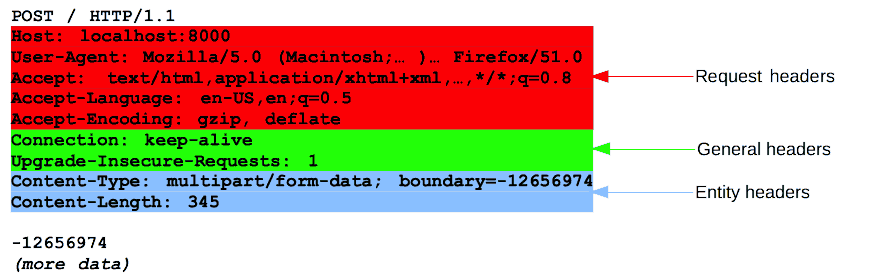
Se aplican tanto a las Peticiones como a las respuestas.

b. **De** **Petición** *(en las Peticiones)* **/Respuesta** *(en las Respuestas)*:

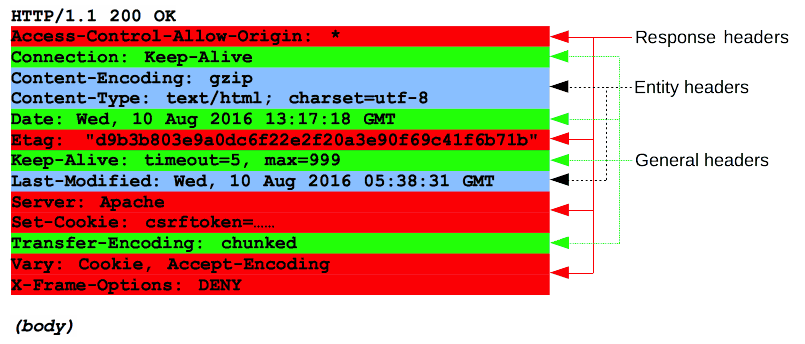
Contienen información extra sobre el contenido que va a obtenerse o sobre el cliente, en el caso de las ***Cabeceras de Petición***. Contienen información extra sobre el contenido, en el caso de las ***Cabeceras de Respuesta***.

c. **De** **Entidad**:

Contienen información extra sobre el cuerpo de la **Entidad**.



Cabecera HTTP Request



Cabecera HTTP Response

**NOTA**:

Adjunto a este documento (*Directorio ‘****Recursos’***), existe un documento con un listado de todas las **Cabeceras HTTP** aceptadas hasta la fecha (*Cabeceras-HTTP.pdf*).

**Función Spider**:

Otra de las funcionalidades con las que cuenta Zap es una **Araña** o **Spider** para la **detección y descubrimiento de nuevos recursos o URL’s en el Sitio Web a auditar**.

Para lanzar el **Spider** (*hay mas formas de hacerlo*), **seleccionamos** el **Sitio** que queremos analizar desde la **Pestaña** ‘*Sitios’*, botón derecho: ‘***Atacar*** *> Spider*’. También es posible **configurar** el **Spider** accediendo a: ‘***Herramientas*** *>* ***Opciones*** *> Spider*’.

**NOTA**:

Es recomendable hacer el ***Spidering*** **completo de todo el *Dominio***, por lo que será necesario ‘***marcar’*** el **ámbito para todo el *Dominio***.

**Función AJAX Spider**:

Las Web basadas en ***AJAX*** presentan **mucha dificultad para que los Spider rastreen su contenido** ya que este, **se genera de forma automática en el Navegador**, y por lo tanto **no es visible** para los rastreadores (*Spiders*). **Zap incluye un Spider ‘*especial’* para este tipo de Sitios Web**.

Para ejecutar un **AJAX Spider**, **seleccionamos** el **Sitio** que queremos analizar desde la **Pestaña** ‘*Sitios’*, botón derecho: ‘***Atacar*** *> AJAX Spider Sitio*’.

Igual que ocurre con el **Spider** ‘*tradicional*’, Zap permite realizar una **configuración del AJAX Spider** para ajustarlo mejor a las necesidades del **Sitio** que se va a auditar.

**Función Forced Browser**:

La **Navegación Forzada** (*Forced Browser*) es una técnica de **Descubrimiento de Recursos** cuyo propósito es **descubrir Ficheros o Directorios no referenciados desde la Aplicación Web** que se está auditando.

Aunque esto se puede hacer de forma manual, pueden utilizarse **técnicas de Fuerza Bruta** **basadas en Diccionario** para buscar este contenido no enlazado en el directorio del **Dominio**. Esta técnica es la que utiliza Zap para realizar Forced Browser, Se van **concatenando palabras** (*admin, private, documents, etc.*) del diccionario, a la **URL Base**, para mandar estas nuevas peticiones al **Servidor**. Así Zap puede detectar, **ficheros** y **directorios** del **Servidor** en función de cual sea el **Código HTTP** **de** **respuesta** (*200, 301, etc.*).

Para ejecutar un proceso de Forced Browser, **seleccionamos** el **Sitio** que queremos analizar desde la **Pestaña** ‘*Sitios’*, botón derecho: ‘***Atacar*** *>Forced Browser Directory (and Children)’*.

Es **fundamental** seleccionar el **Diccionario** (*pestaña* ***Navegación Predefinida***) que se va a utilizar para que el proceso comience. Aunque Zap cuenta con un **Diccionario** por defecto, **sería importante poder utilizar alguno más, cuanto más completo sea el Diccionario, más posibilidades de descubrir más recursos**. Para seleccionar otro Diccionario, vamos a ‘***Herramientas*** *>* ***Opciones*** *> Navegación Predefinida*’.

**Función Fuzzer**:

**Fuzzing**: Técnica de **pruebas** que consiste en el **envío de datos semi-aleatorios mal formados** a un **Software** o **Aplicación Web** para ver **como se comporta** frente a estos **Datos de Entrada** y poder **detectar** si el **Software** o la **Aplicación Web** tiene **fallos** o **comportamientos** **no** **deseados**.

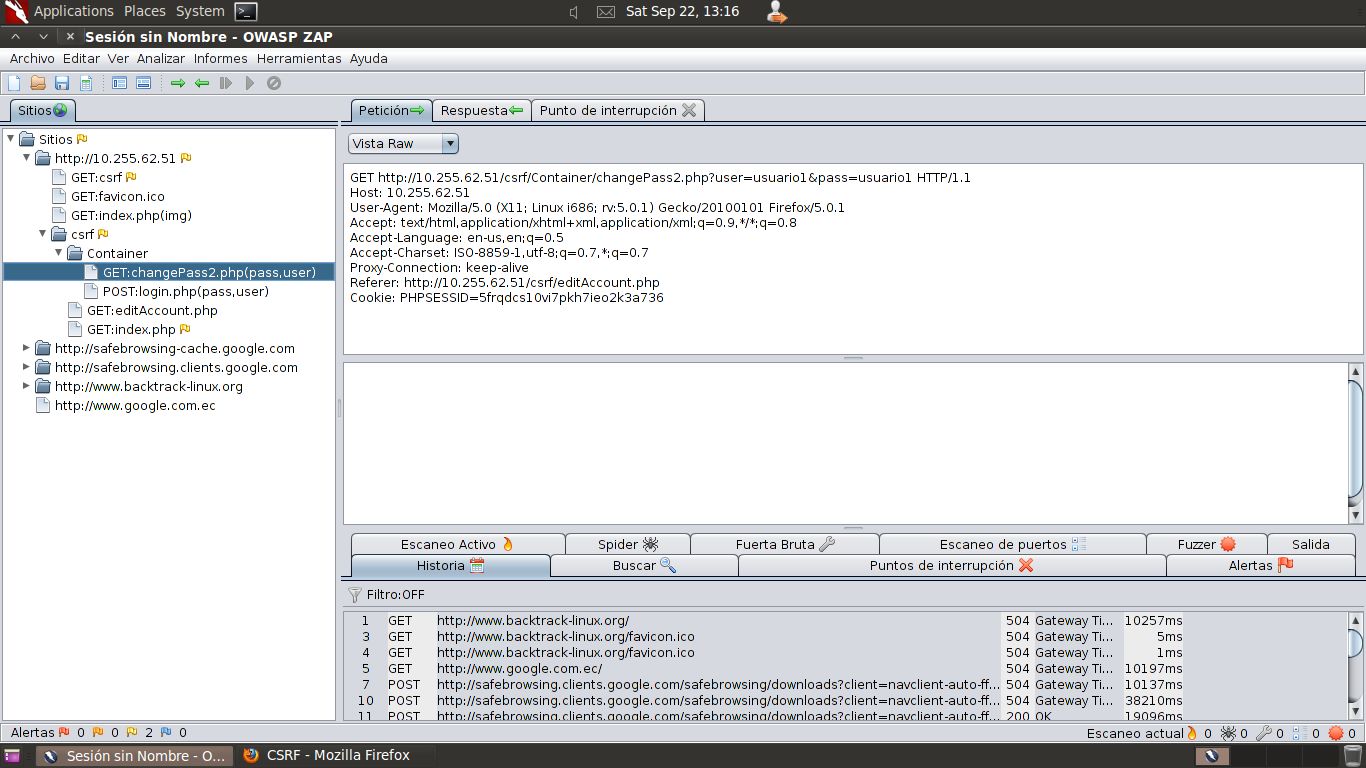
En un proceso de **Pentesting**, se intenta extraer de estos fallos o comportamientos la **máxima información** posible o incluso **explotarlos** como una **vulnerabilidad**.

Para automatizar este proceso se utilizan los ***Fuzzers***. Zap permite hacer **Fuzzing** a través de **peticiones** **GET** y **POST** enviadas desde el cliente. Como para el resto de las funciones vistas hasta ahora, podemos configurar esta a través de, ‘***Herramientas*** *>* ***Opciones*** *> Fuzzer*’.

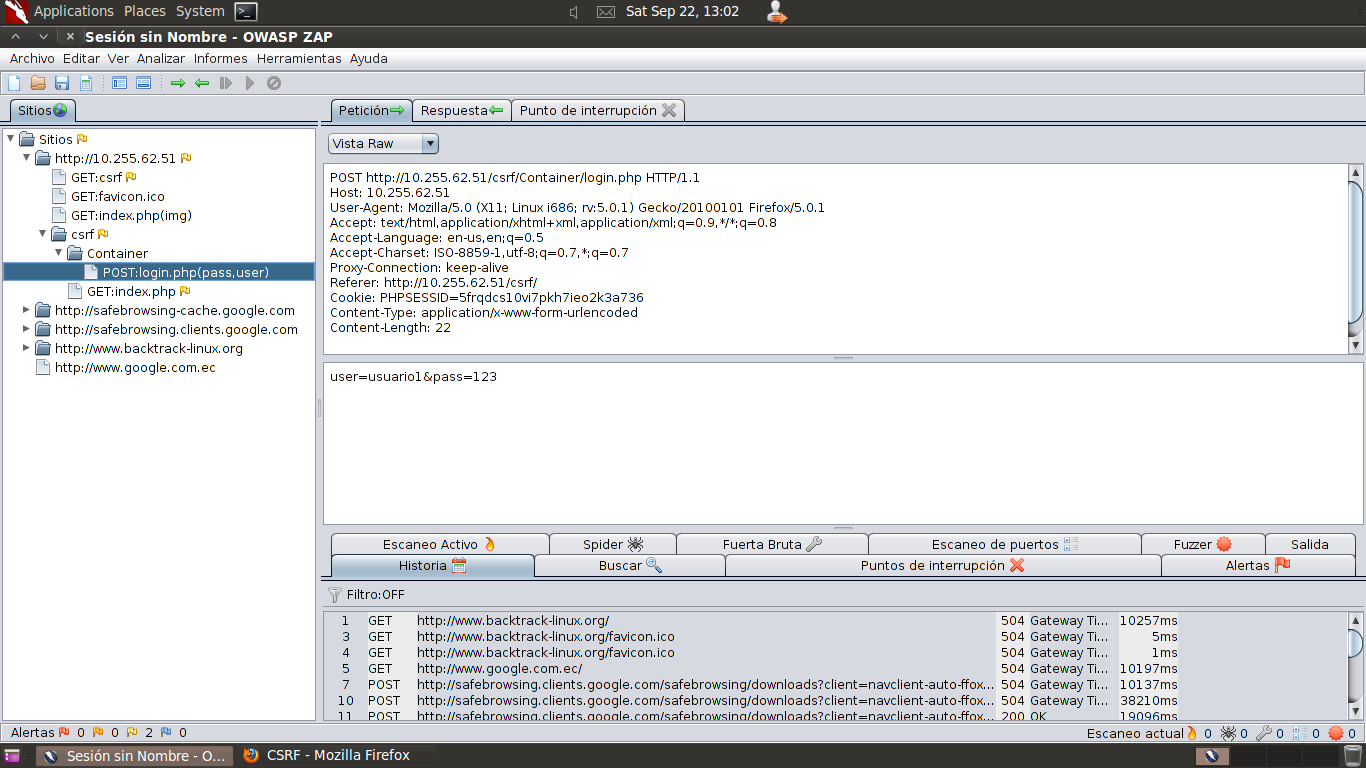
Como se ha comentado en el párrafo anterior, Zap puede ejecutar un **proceso de Fuzzer** a través de **Peticiones GET** (*donde los* ***parámetros*** *de la* ***petición*** *van* ***incluidos*** *dentro del* ***recurso*** *o* ***URL*** *solicitados y* ***son visibles***) y a través de **Peticiones POST** (*donde los* ***parámetros*** *de la* ***petición*** *van* ***incluidos*** *dentro del* ***cuerpo*** *de la* ***petición*** *y* ***no son visibles***).

Para ejecutar un **Fuzzer** sobre una **petición GET/POST** de un determinado **Sitio**, basta con:

1. **Seleccionar** la **petición** que nos interesa para que Zap nos **muestre** los **parámetros** que están implicados:
   1. En el caso de una **Petición GET**, los **parámetros** se mostrarán en el **Header** de la petición.



* 1. En el caso de una **Petición POST**, los **parámetros** se mostrarán en el **Body** de la petición.



1. Seleccionamos el **valor del parámetro** sobre el que queremos hacer el **Fuzzing**.
2. Botón derecho del ratón sobre ese parámetro y **seleccionamos** ‘*Fuzz.*’
3. Elegimos el ***Tipo de Payload*** (*Type*) que se quiere utilizar, en función de nuestro propósito.
4. Pulsamos ‘*Start Fuzzer*’.

**Después, deberemos analizar todas las respuestas que el Servidor nos ha ido devolviendo para cada una de las posibilidades del parámetro que hemos ido lanzando.**

**NOTA IMPORTANTE:**

Es importante tener en cuenta que **todas las peticiones** que va a realizar el **Fuzzer** durante el proceso, **pueden quedar registradas en los Ficheros de Log del Servidor Web**.

**Función Escáner Activo**:

Zap proporciona un **Escáner Activo** para buscar vulnerabilidades de manera rápida.

Para ponerlo en ejecución, **seleccionamos** el **Sitio** que queremos analizar desde la **Pestaña** ‘*Sitios’*, botón derecho: ‘***Atacar*** *>Active Scan’*.

Este **Escaneo Activo** funciona de la siguiente manera:

1. Se hace un recorrido de la **URL** con el **Spider**.
2. Se realiza un **Escaneo Activo** de todas las **URL’s** obtenidas por el **Spider**.
3. Se analiza el contenido de cada **URL** y se muestran las **Alertas** en función de lo crítica que sea la **vulnerabilidad** encontrada.

**NOTA**: **Reducir el Tiempo de un Escaneo Activo**.

Para **minimizar el tiempo** que utiliza Zap para un **Escaneo Activo**, lo ideal es **conocer** cual es el **Contexto** bajo el que se está ejecutando el **Objetivo** (***Tecnologías*** *utilizadas por el* ***Servidor Web*** *que lo aloja*), el **Sistema Gestor de Base de Datos** (*SGBD*), **Tipo de Servidor Web**, **Sistema Operativo**, etc.

De esta forma podemos **centrar el escaneo en ese Contexto** y no probar con todas las posibilidades que ofrece Zap. Para ello, una vez conocemos el **Contexto**, basta con **seleccionar** la **Pestaña** ‘*Tecnologías’* dentro de la **configuración** del **Escaneo Activo** y **marcar** sólo las **opciones** que se **ajusten al Contexto** que hemos descubierto (*utilizando nmap, por ejemplo*).