***Imagen que contiene Logotipo

Descripción generada automáticamente***

15/5/2021

Trabajo 5

Distribución Parrot

Seguridad Informática

Alejandro Ortega Martínez

Grado en Ingeniería Informática

Contenido

[1. Introducción 3](#_Toc72862536)

[1.1. Parrot: ¿Qué es? 4](#_Toc72862537)

[1.2. Parrot: diferencias en cuanto al resto de distribuciones 4](#_Toc72862538)

[1.3. Parrot: Diferentes ediciones 5](#_Toc72862539)

[2. Privacidad 6](#_Toc72862540)

[2.1. AnonSurf 6](#_Toc72862541)

[2.2. Criptografía 7](#_Toc72862542)

[2.2.1. EncryptPad 7](#_Toc72862543)

[2.2.2. GPA 8](#_Toc72862544)

[2.2.3. SiriKali 9](#_Toc72862545)

[2.2.4. ZuluCrypt 10](#_Toc72862546)

[2.2.5. ZuluMount 10](#_Toc72862547)

[2.3. MetaData Cleaner 11](#_Toc72862548)

[2.4. Secure File Deleter (Shred) 13](#_Toc72862549)

[3. Pentesting 14](#_Toc72862550)

[3.1. Information Gathering 14](#_Toc72862551)

[3.1.1. DNS Analysis (DNSmap) 14](#_Toc72862552)

[3.1.2. Live Host Identification 15](#_Toc72862553)

[3.1.3. Análisis OSINT: Maltego 15](#_Toc72862554)

[3.1.5. Escaneo de puertos: nmap 17](#_Toc72862555)

[3.1.6 netdiscover 18](#_Toc72862556)

[3.2. Vulnerability Analysis 18](#_Toc72862557)

[3.2.1. Unix privesc check 18](#_Toc72862558)

[3.2.2. Fuzzing Tools 19](#_Toc72862559)

[3.2.3. OpenVAS 20](#_Toc72862560)

[3.2.4. Auditoria del sistema: Lynis 23](#_Toc72862561)

[3.3. Análisis de aplicaciones web 26](#_Toc72862562)

[3.3.1. Ataques a páginas web (JSQL Injection) 26](#_Toc72862563)

[3.4 Exploitation Tools (Metasploit) 27](#_Toc72862564)

[3.5. Wireless Testing (Airgeddon) 31](#_Toc72862565)

[3.5. Automitive Tools (CAN) 33](#_Toc72862566)

[4. Conclusión y opinión 39](#_Toc72862567)

[5. bibliografía 40](#_Toc72862568)

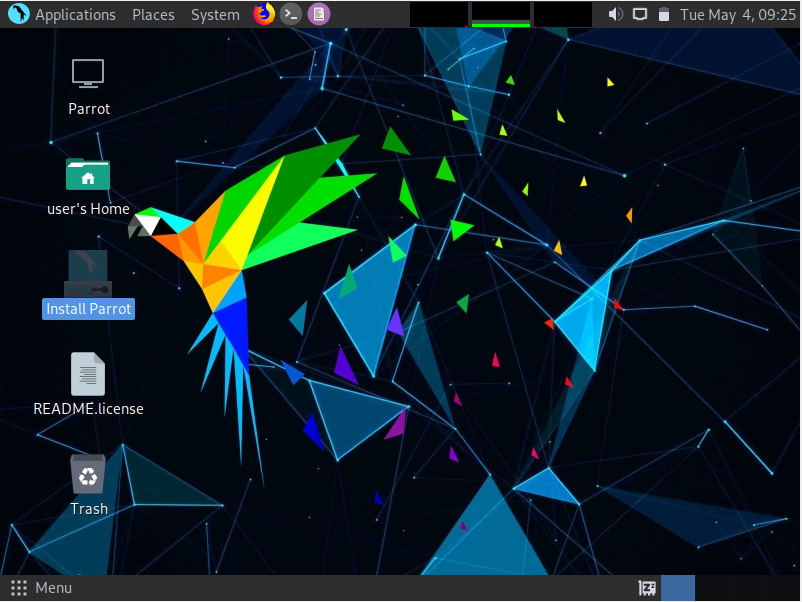
[Bibliografía 40](#_Toc72862569)

# Introducción

En este trabajo indagaremos en una distribución orientada a la seguridad informática, como es Parrot.

Se ha elegido Parrot porque, a pesar de que hay otras distribuciones más conocidas, como pueden ser Kali Linux o BlackArch, he considerado que podía ser más interesante enfocar el trabajo a una distribución más nueva y menos conocida como es Parrot, para ver que puede ofrecer, y si está a la altura de las más conocidas.

En primer lugar hablaremos un poco sobre la distribución en general, y luego iremos navegando por cada una de las herramientas que nos ofrece.

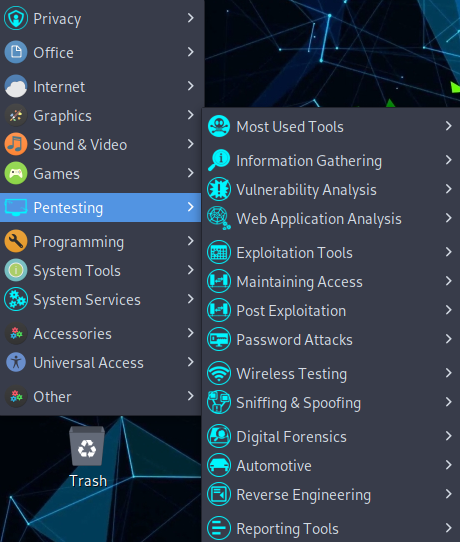


## 1.1. Parrot: ¿Qué es?

Parrot es (Velasco, 2020) una **distribución de hacking ético basado en Debian**, orientada a la seguridad informática. Ha sido una distribución construida desde cero, sin base en otras distros orientadas a la seguridad, y que se centra en las pruebas de penetración, evaluación y análisis¸ así como para análisis forense de sistemas, preservación del anonimato, y criptografía.

## 1.2. Parrot: diferencias en cuanto al resto de distribuciones

Una ventaja por la que se lo conoce a Parrot es la variedad de herramientas que tiene, en comparación con sus competidores. Se han seleccionado las mejores de cada campo, teniendo así una **gran abanico de tipos de herramientas, pero solo las mejores de cada uno**. Mas adelante entraremos en ellos, pero para tener un vistazo global, tenemos las siguientes categorías (Aunque dentro de cada una hay diferentes subcategorías):



Otro punto a favor es que viene de serie preparado para que se puedan usar todas las funcionalidades de serie, sin tener que configurar nada a parte.

Por último, podemos observar que desciende de la rama “testing” de Debian, por lo que siempre estarán disponibles las nuevas características.

## 1.3. Parrot: Diferentes ediciones

Dentro de la página web de Parrot (Parrot, s.f.) tenemos diferentes tipos de distribuciones, en función de cuál sea nuestro objetivo:

* **Home edition**: Es un sistema operativo de propósito general, pero con el Look&Feel de Parrot. Es una edición diseñada para usarla en el día a día, teniendo en cuenta la privacidad, y para el desarrollo software. A parte, se pueden ir instalando manualmente las diferentes herramientas de Parrot, para así llegar a construir un entorno de pentesting personalizado.
* **Security Edition**: Este sistema operativo ya es de propósito especifico, orientado a test de penetración y ejercicios de Red Team. Contiene una gran variedad de herramientas de pentesting listas para usar. Esta es la distribución que usaremos en la práctica.
* **Ediciones portables:** Parrot también ofrece ediciones portables para sistemas empotrados, entornos en la nube, maquinas IoT y máquinas virtuales:
  + OVA: Una máquina virtual ya preparada para instalar, con todo preconfigurado, para VirtualBox y VMWare. Tenemos tanto las ediciones Home como Security.
  + Docker: También tenemos la opción de montar el sistema operativo en un contenedor, y correrlo desde la nube.
  + Pwnbox: Parrot, en colaboración con la empresa HackTheBox, una herramienta para poder usar todas las características de Parrot desde el navegador (HackTheBox, s.f.).

# 2. Privacidad

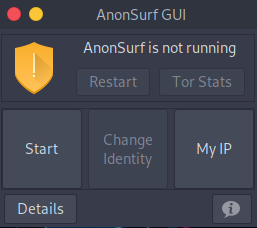
La primera opción que nos encontramos en el menú de Parrot es la de privacidad, que nos brinda herramientas relacionadas con el tema, como pueden ser la navegación segura por internet, o la eliminación de metadatos.



## 2.1. AnonSurf

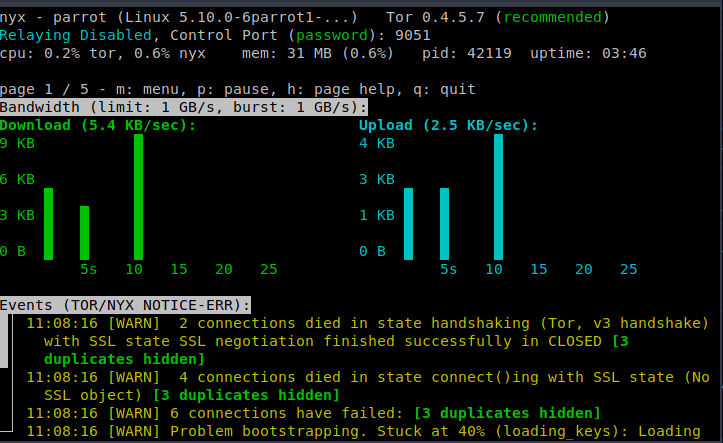


AnonSurf es una aplicación que nos permite conectarnos a la red Tor, y a redes i2p. Tiene tanto una interfaz gráfica, como interfaz de línea de comandos. (Parrot Documentation, s.f.)



Como podemos observar, no tiene una interfaz muy simple e intuitiva, desde el botón start iniciamos la vpn, y una vez iniciada, podemos reiniciar o cambiar la identidad de los respectivos botones.

En la sección “My IP” podemos ver nuestra IP, y en los “Tor Stats” podremos ver una gran cantidad de estadísticas, en las que no entraremos en profundidad:



## 2.2. Criptografía

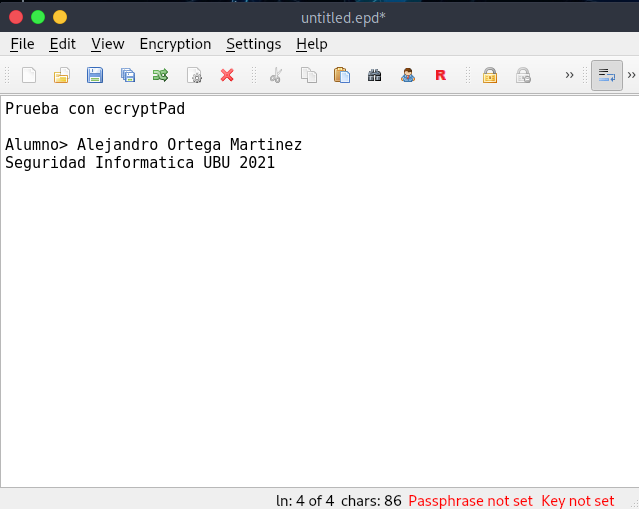


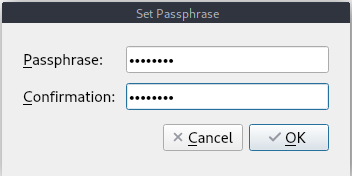
Dentro de este apartado tenemos numerosas herramientas para jugar con criptografía. A continuación se van a describir un poco a rasgos generales cada una, pero no se va a entrar a probar diferentes algoritmos criptográficos, ya que eso es temario de la parte de prácticas de esta materia, y ahí se ha indagado con mucha profundidad en el tema.

### 2.2.1. EncryptPad



EncryptPad es una aplicación para ver y editar texto encriptado simétricamente, y en esta distribución viene instalada la versión gráfica, aunque también se puede usar mediante línea de comandos (EcryptPad Documentation, s.f.).

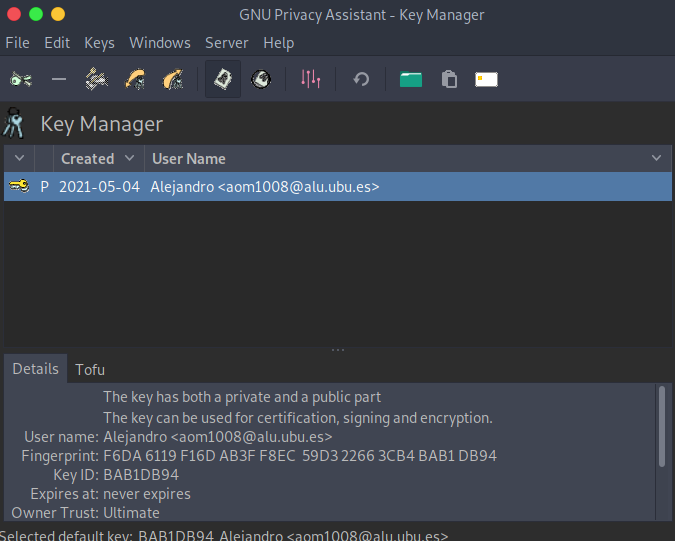
Al abrirlo, es un editor de texto normal y corriente. En el podemos escribir lo que queramos, y si le damos a la opción de encriptar en el menú superior, nos dejara introducir la contraseña, la key, y encriptar o desencriptar.



### 2.2.2. GPA



GPA son las siglas de Gnu Privacy Assistant. GPA es una aplicación con interfaz grafica para GPG (GNU Privacy Guard).

Debido a que ya se realizó una práctica completa sobre GPG en prácticas, no se va a profundizar más en el tema, más allá de ver un poco como funciona la interfaz gráfica.

Se ha creado una llave como ejemplo, con mis datos, como se puede ver en la siguiente captura de pantalla. Desde el menú de keys se podrían crear más, o importar otras para añadirlas al llavero.

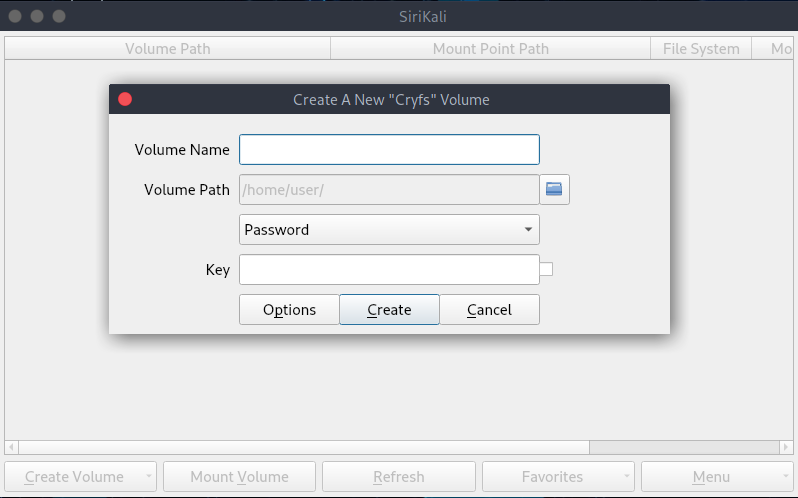
### 2.2.3. SiriKali



SiriKali es una herramienta que permite gestionar ficheros encriptados en encryptfs, cryfs, encfs, gocryptfs, fscrypt y securefs.

Esta aplicación permite crear volúmenes cifrados y montar directorios ya cifrados (Atareao, s.f.).

Una vez la abrimos, nos muestra los volúmenes que tenemos en el equipo, y con el botón “Create Volume” nos permitirá crear un nuevo volumen con el cifrado que deseemos.



Aquí podemos ver un ejemplo de cómo se crea un volumen encriptado en “Cryfs”.

### 2.2.4. ZuluCrypt



Similar a SiriKali, su objetivo es crear sistemas de archivos encriptados, por lo que no se entrará en detalle.

### 2.2.5. ZuluMount



Es la versión de línea de comandos de ZuluCrypt

## 2.3. MetaData Cleaner

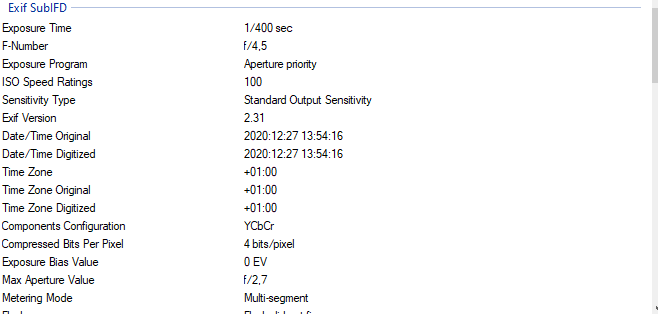


Esta herramienta nos permite limpiar los metadatos almacenados en un archivo.

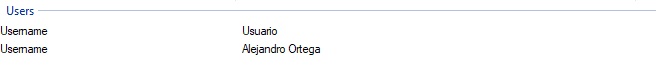
Haciendo referencia a mi trabajo de la herramienta FOCA, podemos resumir los metadatos de la siguiente manera:

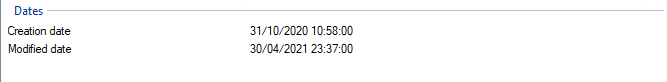
Los archivos que generamos en el ordenador con diferentes programas tienen **datos concretos**, que son **los que introducimos**, pero también hay otra serie de datos que describen otras características sobre el contexto del archivo, unos **metadatos** de los que por defecto **no se tiene control directo**. Por ejemplo, si creamos un archivo de Windows con Word u otra herramienta podemos escribir en él todo lo que queramos, pero luego la aplicación generará una serie de metadatos para ese archivo. En ellos describirá aspectos como la fecha en la que fue creado, la aplicación exacta que utilizaste o la fecha en la que hiciste una última modificación del contenido.

Para hacer la prueba analizare una foto que tengo en mi equipo, realizada por mí mismo en una excursión, con la herramienta FOCA, para ver los metadatos que contiene. Como podemos observar en la siguiente captura, se ha mostrado una pequeña parte de los datos EXIF de la foto:

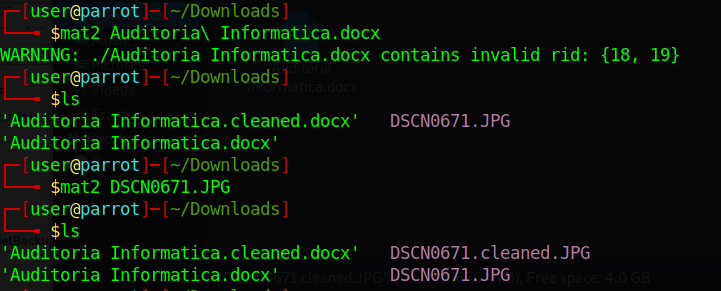


También haremos lo mismo con un documento Word, para ser exactos, con uno de los trabajos de esta asignatura:

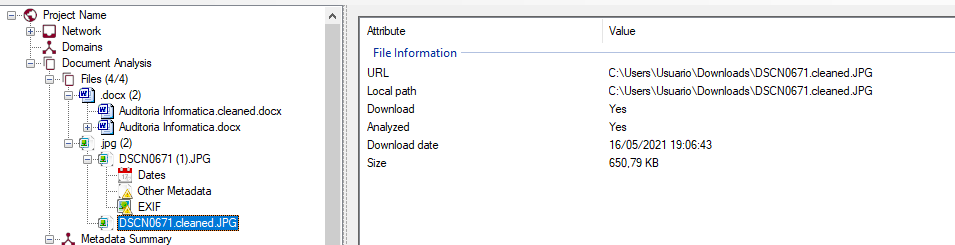




A continuación, pasaremos la imagen por la herramienta de limpiado de metadatos:



Ahora pasaremos las dos versiones “cleaned” por FOCA, para ver el resultado:



Como podemos observar, mientras que las versiones normales de los archivos nos permiten desplegarlas, para ver los metadatos que tienen, de estas dos nuevas no se nos permite, lo cual quiere decir que no tienen ningún tipo de metadato.

## 2.4. Secure File Deleter (Shred)



Shred es una herramienta que nos permite eliminar un archivo permanentemente de nuestro sistema de almacenamiento.

Como sabemos, cuando un sistema operativo borra un fichero, lo que hace es marcar que las posiciones de memoria donde estaba dicho archivo ahora se encuentran libres para poder sobre escribirlo. Sin embargo, hay muchos programas que nos permiten recuperar archivos que estaban “borrados”, o, mejor dicho, marcados como borrados.

La finalidad de esta herramienta es no permitir eso, y para ello, aparte de marcar el archivo como “borrado”, lo sobre escribe varias veces con 1s y 0s, para borrar los datos permanentemente (Haas, 2020).

Veremos un ejemplo con los archivos utilizados para el MetaData Cleaner:



El parámetro **-u** es para borrar los archivos una vez sobre escritos. Podemos usar otros argumentos, como **-f** para cambiar permisos para permitir escribir si los necesitara, **-n=x**, para indicar cuantas veces (x) queremos que sobrescriba el archivo (por defecto son tres), o **-z**, que añade una sobre escritura final de todo ceros, para ocultar que se ha usado el shred.

# 3. Pentesting



En este apartado, parrot incluye una inmensa cantidad de herramientas, divididas en varios grupos según su objetivo.

Si tratáramos todas las herramientas, se haría un trabajo demasiado extenso, por lo que solo vamos a ver algunas de las más interesantes de cada subgrupo.

## 3.1. Information Gathering



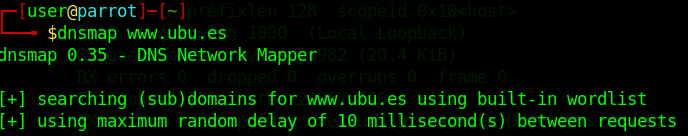
Dentro de este bloque, podemos observar diferentes tipos de herramientas. No vamos a profundizar en todos, pero si en los más importantes:

### 3.1.1. DNS Analysis (DNSmap)



Dentro de este bloque vamos a hablar de **DNS Map**, una herramienta que escanea subdominios usando técnicas de fuerza bruta. Le podemos dar nosotros una wordlist, o usar él la que tiene precargada, que contiene más de 1000 palabras en Inglés y Español.

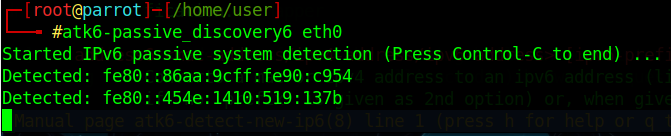
La sintaxis es la siguiente:



### 3.1.2. Live Host Identification

En esta rama entran todas las herramientas que tienen que ver con identificar los hosts que están conectados a nuestra red.

En nuestro caso vamos a usar la herramienta de **passive\_discovery6**, que entra dentro del paquete de herramientas “**The Hacker’s Choice**”**.** Esta herramienta hace un sniffing pasivo de la red, detectando todos los clientes conectados, y volcando su IPv6.

****

Como podemos observar, pasándole la interfaz por la que queremos que escuche, llamamos al programa, que poco a poco va recopilando todos los clientes conectados a la red (En este momento solo mi ordenador y mi teléfono móvil).

### 3.1.3. Análisis OSINT: Maltego

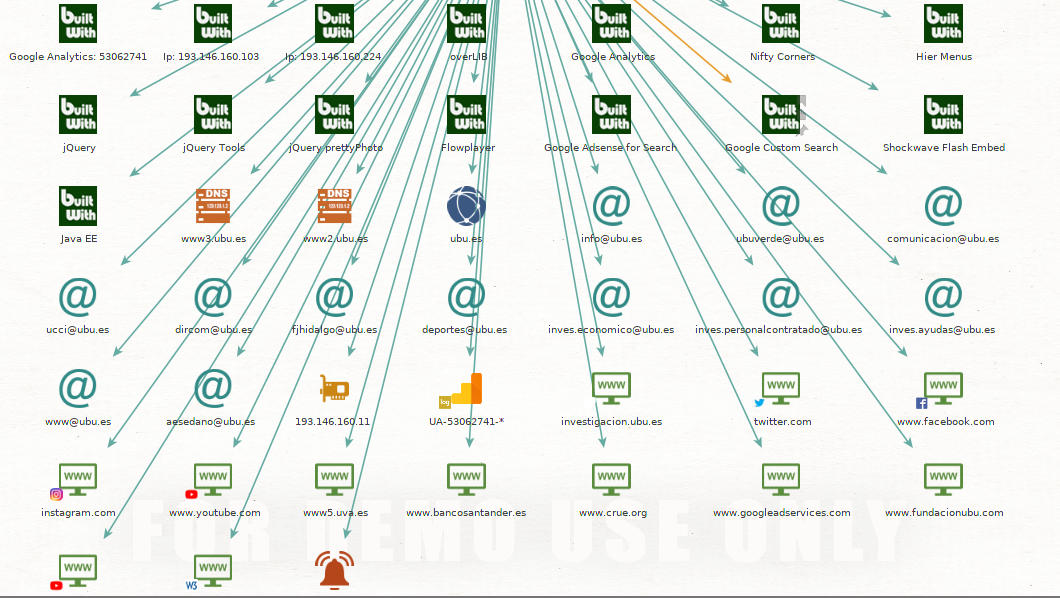
Maltego es una herramienta muy potente, que nos permite encontrar información de cualquier persona o empresa por internet cruzando datos y generando grafos, como redes sociales, correos electrónicos, etc.

Para este ejemplo Instalaremos las extensiones CaseFile Entities, el have i been pawned, y el social link.

Para este ejemplo, intentaremos sacar toda la información posible de la ubu, a ver que conseguimos:



Tras ejecutar un análisis sencillo, obtenemos los siguientes resultados:



Caben destacar que se ha encontrado todas las redes sociales de la Ubu, así como su canal de YouTube.

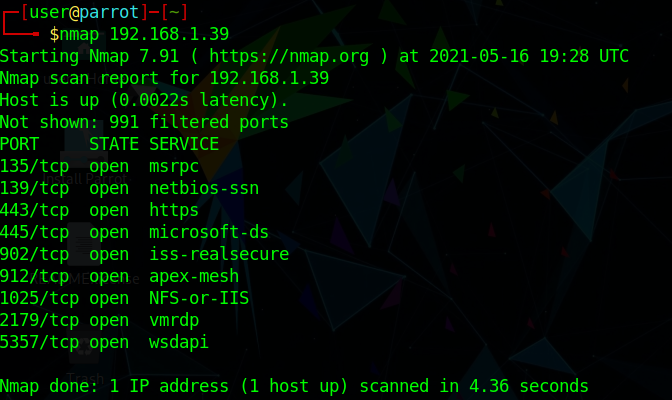
También se han descubierto unos cuantos correos y paginas web, como la de la fundación ubu.

A partir de aquí, se podrían hacer escaneos a cada uno de los elementos conseguidos, creando una red como nos vaya interesando.

### 3.1.5. Escaneo de puertos: nmap

Nmap es una aplicación sencilla a la par que útil. Dado un host, nos permite ver que puertos tiene abiertos.

La sintaxis es muy sencilla, solamente tienes que proporcionar la dirección IP de la víctima, y nmap se encarga de escanear puertos.



Aquí podemos observar algunos de los puertos abiertos de mi PC. Hay otros 991 puertos que no ha mostrado.

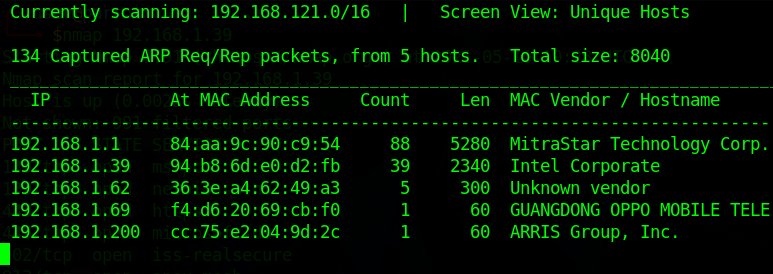
Esta es la funcionalidad básica de nmap, pero también tiene alguna más avanzada, como la de detectar el sistema operativo de la víctima. Esto es porque cada SO trata los paquetes de una manera diferente, y esto lo podemos usar en nuestro favor. Esto se hace con el parámetro -A:



También tiene muchas más características, como permitirnos trocear los paquetes para evitar firewalls, o falsear la dirección ip de origen y de destino.

### 3.1.6 netdiscover

Esta es una aplicación que nos permite ver quienes están conectados a nuestra red. Es muy simple, únicamente la ejecutamos con netdiscover, y nos saldrá información como esta:



Lo que hace es ir escaneando todas las direcciones ip de la red para ver si existen, y de ser así, nos las muestra, junto con su dirección MAC.

## 3.2. Vulnerability Analysis

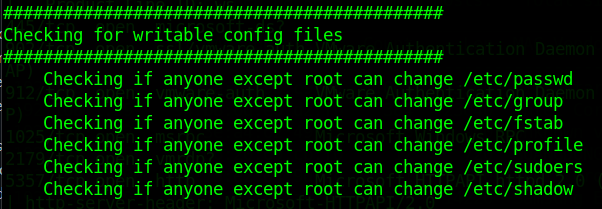


En este apartado veremos alguna herramienta de análisis de vulnerabilidades que trae consigo Parrot.

### 3.2.1. Unix privesc check

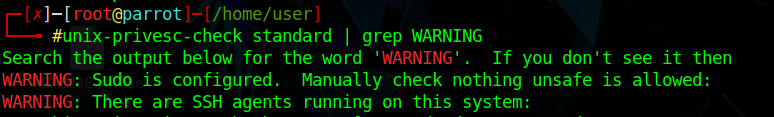


Esta herramienta nos permite comprobar si hay algún fallo de seguridad en la configuración de la maquina con sistema operativo basado en Unix que puedan permitir a usuarios sin privilegios escalar privilegios a otros usuarios, o a aplicaciones locales (como bases de datos) (pentestmonkey, s.f.)



Este es una parte de la salida del script, en el que observamos que comprueba si hay usuarios que puedan cambiar las contraseñas a parte del root.

Para saber si algo está mal, este script nos avisa con la palabra WARNING, por lo que haciendo un grep, podemos comprobar si todo está bien:



Podemos observar que nos avisa de que el sudo está configurado, y que hay SSH corriendo en la máquina.

### 3.2.2. Fuzzing Tools

Este tipo de herramientas, de lo que se encargan en de intentar “romper” las entradas de algún tipo de software, introduciendo parámetros inválidos o inesperados, para ver como los trata.

Unos ejemplos que trae de estas herramientas Parrot son tanto Afl como bed, que nos sirven para forzar estas entradas.

Con la herramienta de bed podemos hacer fuzzing a páginas web, servidores de correo, etc; mientras que con afl, el fuzzing está orientado a programas y aplicaciones.

### 3.2.3. OpenVAS

OpenVas es un Famework de pentesting que se compone de una colección de herramientas para escanear sistemas (Otieno, s.f.).

OpenVAS se compone de:

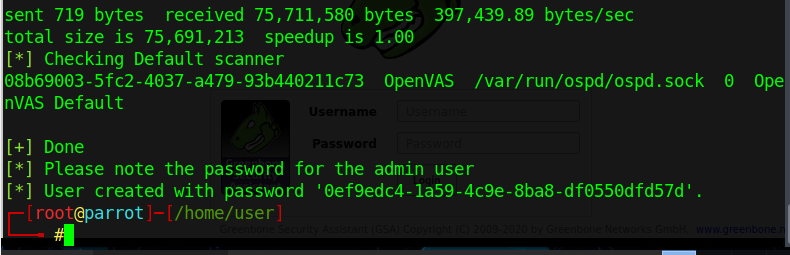
* Una base de datos de resultados y configuraciones.
* Un escáner que es capaz de realizar varios test de vulnerabilidad de redes
* Una colección de vulnerabilidades de red.
* Una interfaz web que nos permite ejecutarlo.



Para empezar, tendremos que instalar todos los paquetes con el comando:

Gvm-setup.

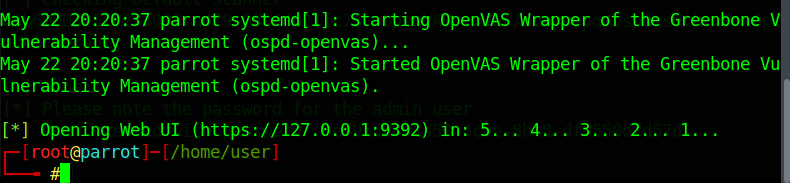
Una vez instalados, nos saldrá algo como esto:



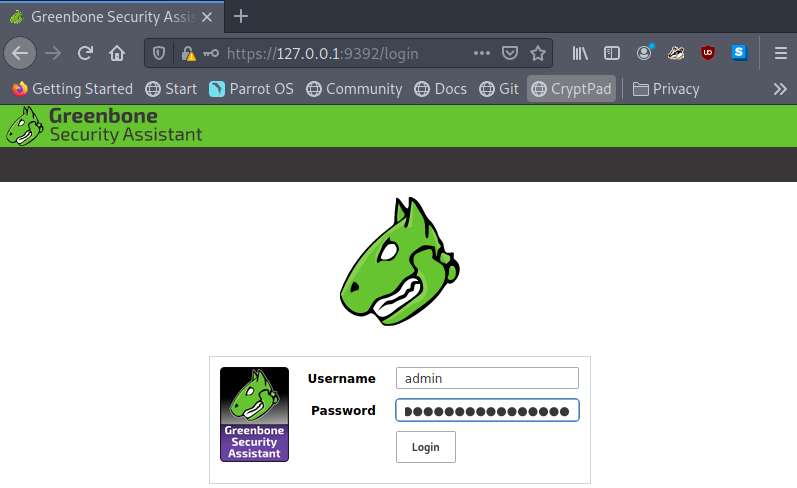
Ahora tenemos que crear la conexión, por lo que ejecutaremos el comando:

Gvm-start

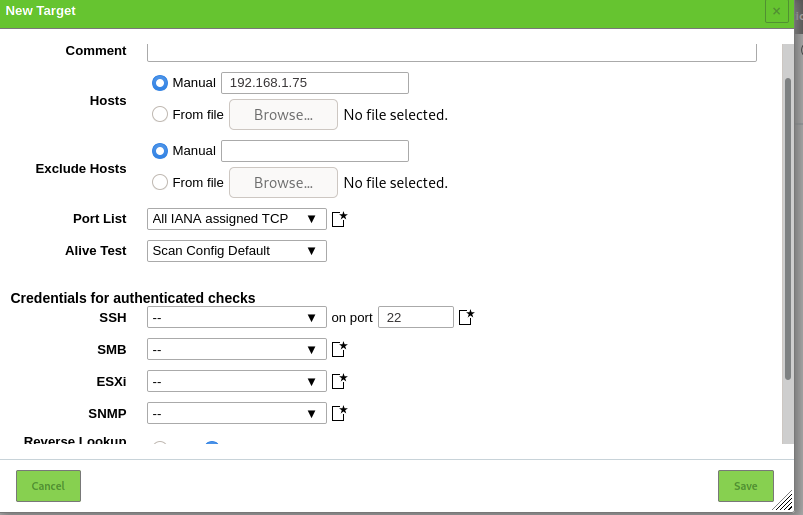
Hasta que veamos que ha cargado la interfaz gráfica:



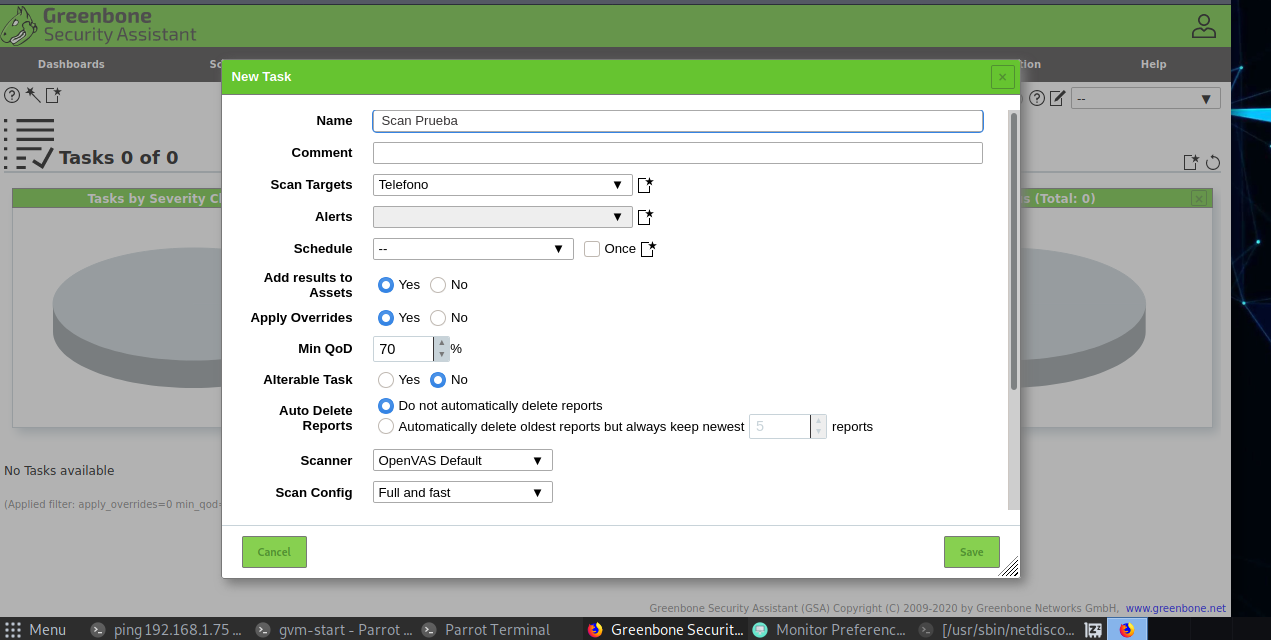
Una vez inicializada, entramos e introducimos el nombre de usuario (admin) y la contraseña que se nos ha proporcionado en el setup:



A continuación, introduciremos un host a atacar. Ya hemos visto anteriormente como ver que direcciones están conectadas a nuestra red, por lo que no entraremos en detalle. Esto lo haremos en el apartado new Target:



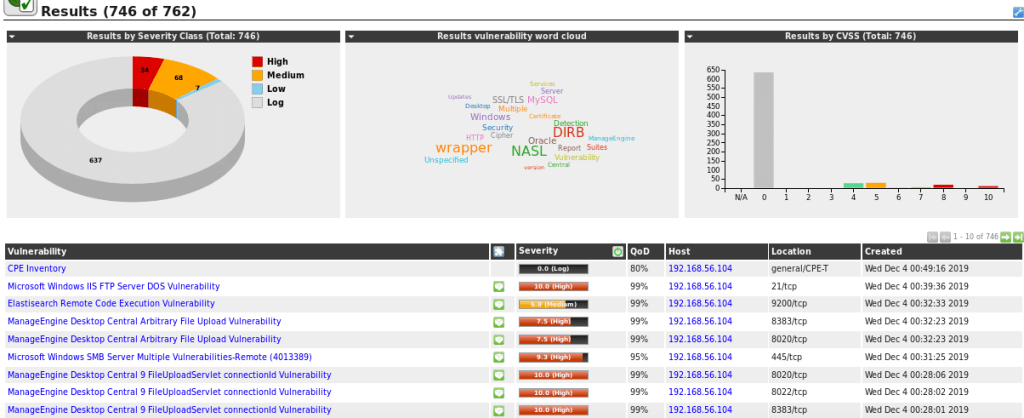
Después de esto, vamos a escanearlo. Iremos a Scan>Task>New Task, e introduciremos los datos:



Le daremos a guardar, y al botón de iniciar:



Después del escaneo, nos saldrá una ficha de resultados como la siguiente (Tomada de (TEHG, s.f.), ya que la mía no tenía nada):



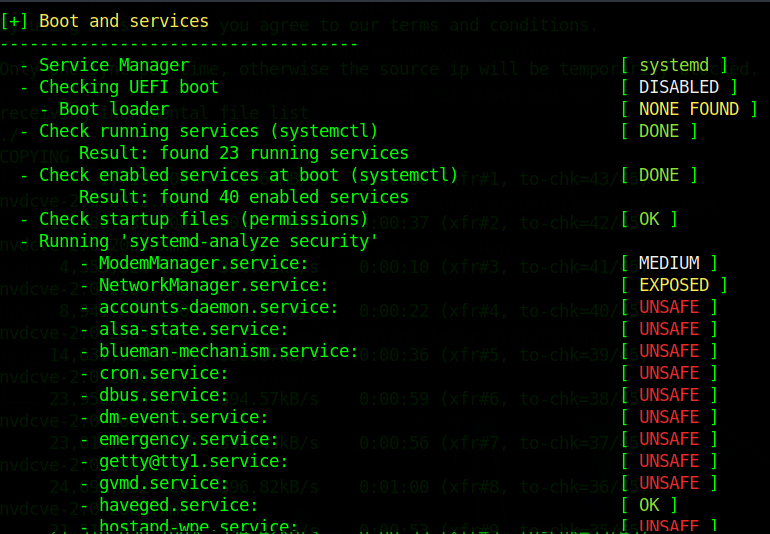
Podemos observar que nos lista todas las vulnerabilidades que ha encontrado, así como las categoriza según lo peligrosas que sean.

### 3.2.4. Auditoria del sistema: Lynis

Lynis es una herramienta de seguridad que se encarga de realizar un escaneo de seguridad del sistema (Cisofy, s.f.). Nos permite realizar escaneos tanto de forma local como remota. A continuación, realizaremos un escaneo de nuestro equipo, y veremos que nos diagnostica:

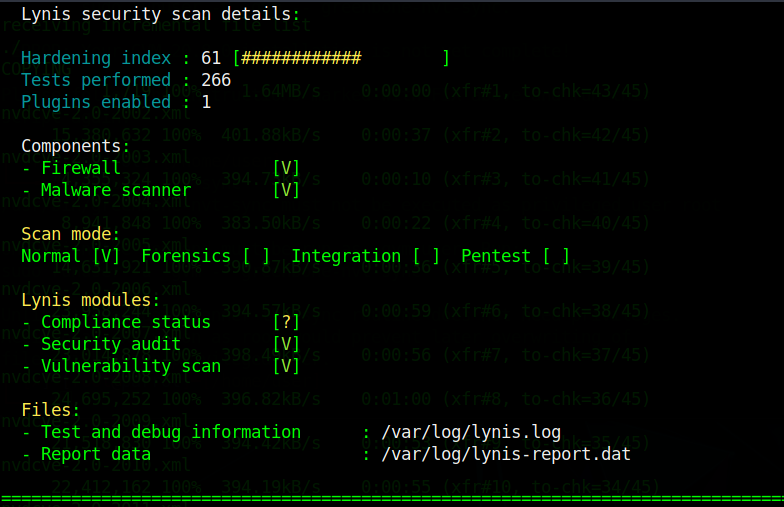


En primer lugar, nos hace un escaneo del sistema operativo, para ver cual se está corriendo en la máquina.

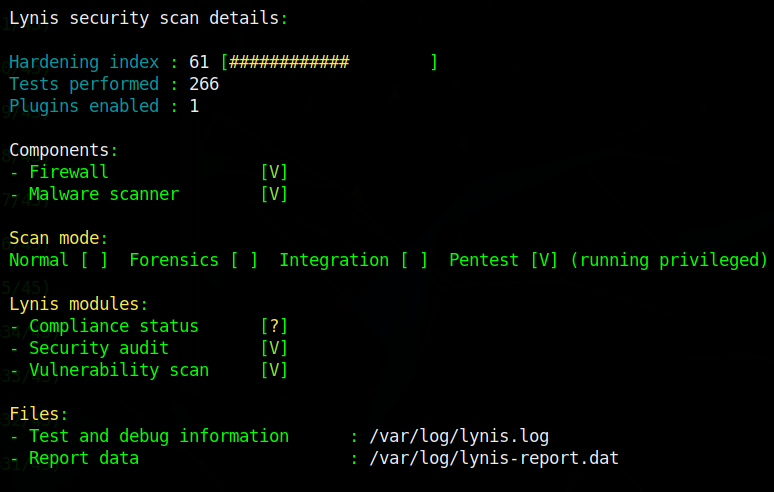


En el apartado de los servicios, nos enseña los servicios que son vulnerables, por lo que los podríamos tener en cuenta para un posible ataque.

Al final del escáner, nos muestra unas características generales, como podemos ver a continuación:



Podemos ver que nuestro equipo ha conseguido una puntuación de seguridad de 61/100. También podemos ver que tenemos el firewall activado, así como el escáner de malware. El escaneo que hemos hecho es el de seguridad, pero, hay otros tres. El resultado de un escaneo de Pentest sería algo como lo siguiente:



## 3.3. Análisis de aplicaciones web



En esta sección vamos a probar distintas aplicaciones que nos permiten analizar aplicaciones web, tanto su seguridad, como el cómo explotarlas.

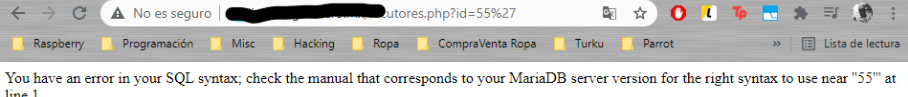
### 3.3.1. Ataques a páginas web (JSQL Injection)

Vamos a probar a hacer un ataque de SQL injection con la herramienta JSQL que viene instalada por defecto.

Para empezar, resumiremos rápidamente lo que es un ataque de inyección SQL. Un SQLInjection es un ataque en el que nos servimos de como están preparadas las consultas que realiza una aplicación web a una base de datos, para inyectar el código SQL que queramos, y hacernos con la información.

Por privacidad, se censurarán las URLs y datos privados.

Primero, tenemos que saber si una página web es vulnerable al SQL injection. Para esto, una opción es con ir a ella, seleccionar algún producto de algún catálogo, y en la URLs poner una ’ como se muestra a continuación:



Como podemos observar, nos dice que hay un error de SQL, por lo que esto quiere decir que sus consultas a las bases de datos no están preparadas contra la inyección SQL.

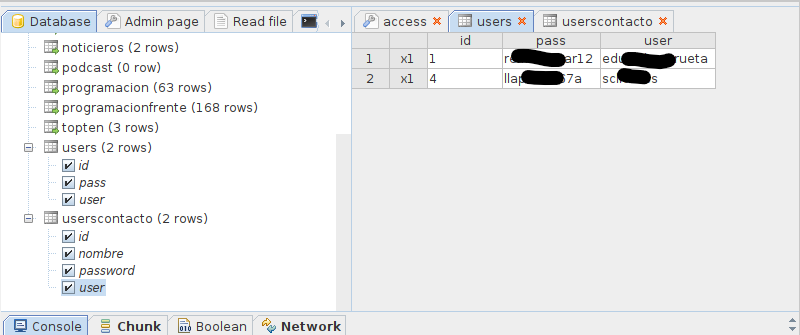
Para ejecutar el ataque, vamos a JSQL Inyection, e introducimos en el campo superior la URL:



Al darle a ejecutar, después de hacer las correspondientes inyecciones SQL, obtendrá las siguientes bases de datos:



Vamos a acceder, por ejemplo, a la segunda base de datos. Iremos a las tablas users y userscontacto. Seleccionaremos las filas de cada tabla que queremos que nos muestren, y le damos click derecho sobre el nombre de la tabla, y “load”. Así, nos mostrará los siguientes resultados con usuarios y contraseñas:



Y en usercontacto, la tabla es la siguiente:



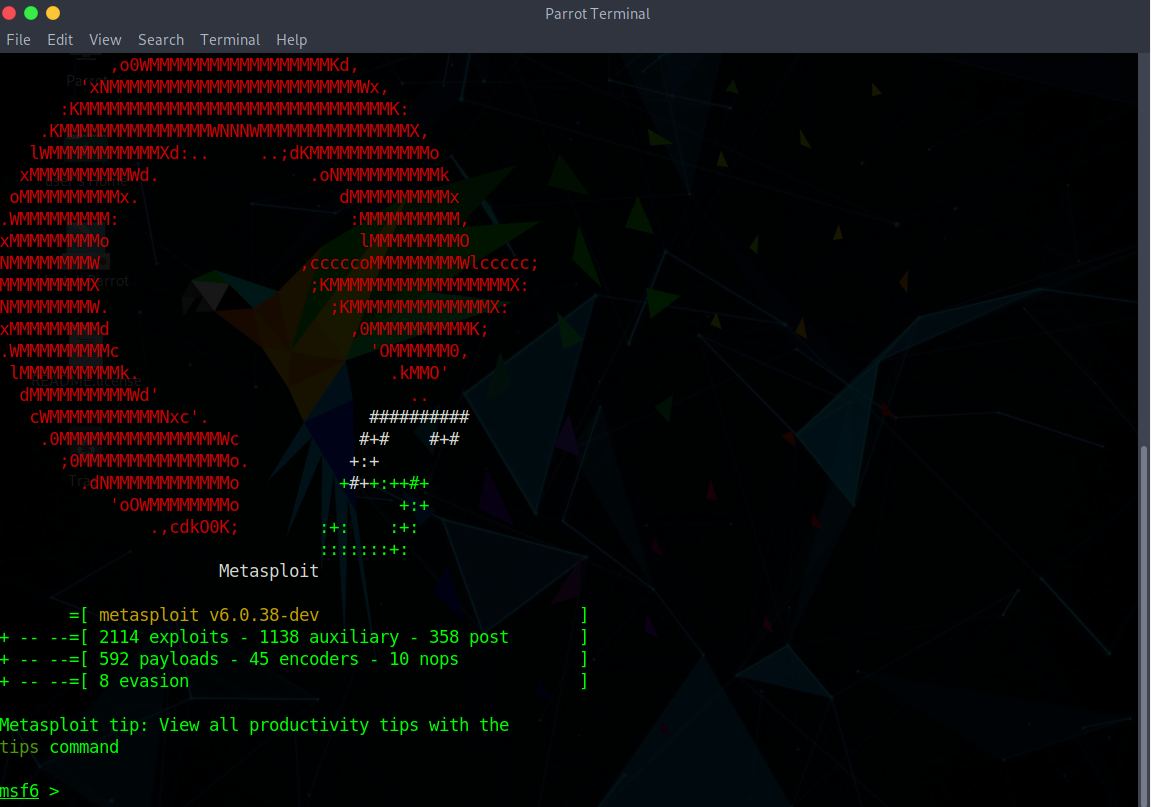
## 3.4 Exploitation Tools (Metasploit)



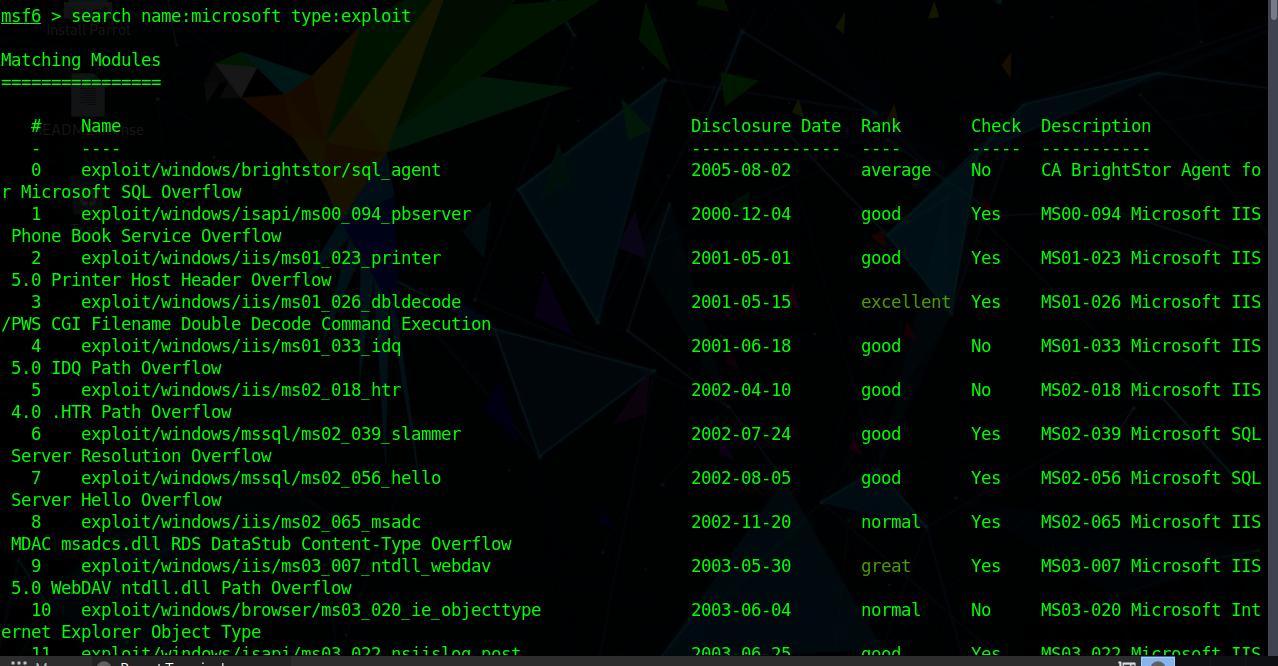
En este apartado vamos a ver diversas aplicaciones que nos pueden servir para crear exploits. En concreto vamos a usar metasploit, y vamos a usar las siguientes referencias (tutorialesPoint, s.f.) (Obbayi, s.f.)

Metasploit es, más que una herramienta, un Famework con múltiples utilidades. En nuestro caso lo vamos a usar para crear un payload y enviarlo a un equipo con Windows 10, para conseguir introducirnos en él.

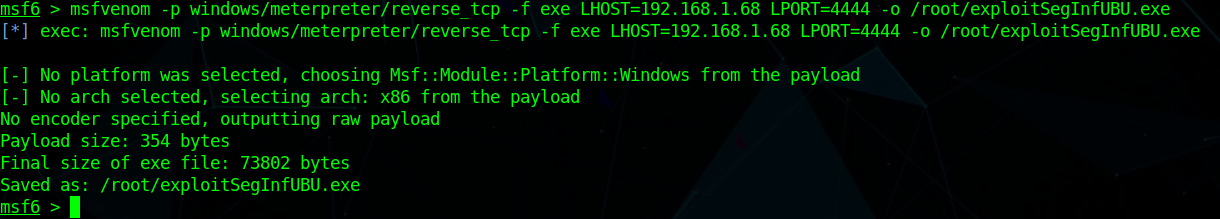
Lo primero que vamos a hacer es abrir una consola de metasploit, que tiene la siguiente apariencia:



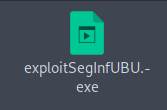
Como en nuestro caso lo que queremos es crear un exploit para Windows, usaremos la herramienta Search para ver que exploits hay disponibles con el comando Search name:Microsoft type:exploit



Como vemos, nos muestra una larga lista de exploits relacionados con Microsoft. En nuestro caso utilizaremos el llamado **windows/meterpreter/reverse\_tcp**

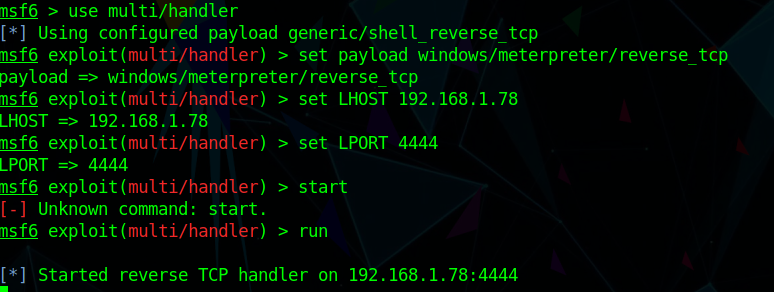
Que no es más que un exploit para hacer una conexión inversa desde la maquina victima a la nuestra, vía tcp.

Hemos creado un payload llamado exploitSegInfUBU, con el exploit mencionado anteriormente, y con localHolst y local port los indicados en la captura.



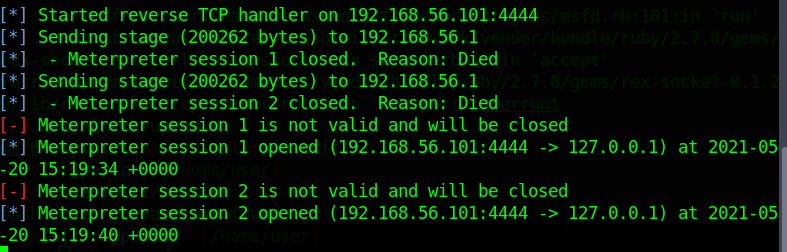
El siguiente paso es crear un “Listener” en el puerto seleccionado. Esto lo haremos también con metasploit.

Lo haremos con la siguiente sucesión de comandos:

****

A continuación, para más simplicidad de la práctica, desactivare los antivirus en mi equipo (tanto Sophos como Windows defender), y ejecutaremos el programa creado.

NOTA: Para este punto se ha tenido que crear una red virtual en la que conectar el sistema operativo anfitrión con la máquina virtual parrot, pero no es material de la asignatura, por lo que no se profundizará en el tema.



Como podemos observar, se conecta al objetivo, pero se muere el proceso, posiblemente por culpa del antivirus.

Por falta de tiempo no se va a intentar más, pero se hará el desarrollo teórico de como continuar una vez tengamos la consola abierta.

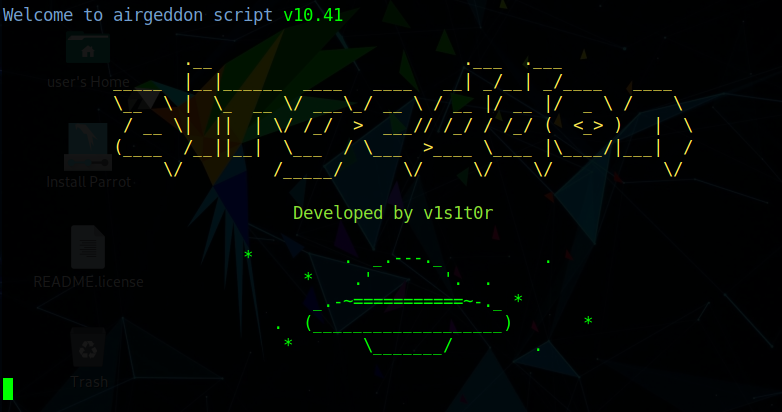
Como es obvio, se ejecutaría sin permisos de administrador, por lo que, para escalar permisos, haríamos lo siguiente:

En una nueva sesión del multi/handler, ejecutaríamos el exploit **Windows/local/bypassuac\_comhijack.**

## 3.5. Wireless Testing (Airgeddon)

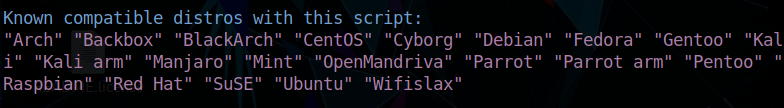


En este apartado, vamos a indagar en las auditorias de conexiones inalámbricas. Para ello, utilizaremos la aplicación Airgeddon.

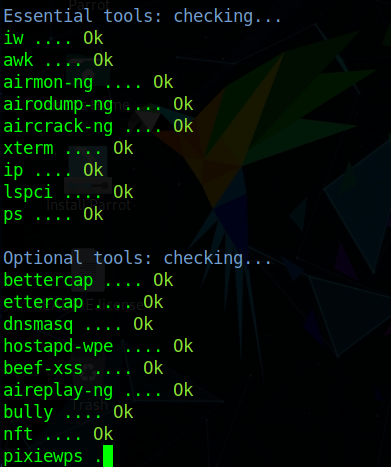


Airgeddon consiste en un Famework que reúne varias herramientas, como pueden ser airmon-ng, airodump-ng, aireplay-ng, etc.

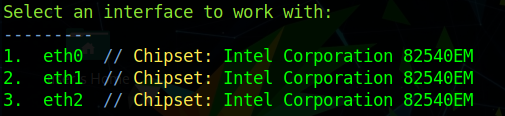
Lo primero que comprueba esta aplicación es que estemos en una distribución compatible:



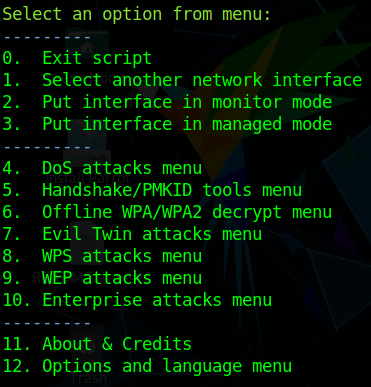
Y que tengamos las herramientas que usa instaladas:



Una vez que hayamos hecho todos los preparativos, nos pedirá seleccionar con que interfaz wifi queremos funcionar:



A continuación, nos dará una lista de opciones que podemos hacer:



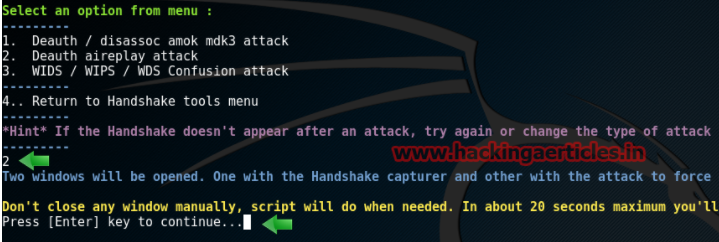
Debido a que **no dispongo de un adaptador wifi externo**, **no se podrá poner en modo monitor**, por lo que la mayoría de las practicas no se podrán realizar.

Debido a esto, a partir de ahora, se realizará una exposición teórica de cómo se usa la aplicación, en nuestro ejemplo, para **conseguir una contraseña wifi**.

Nota, a partir de ahora, se seguirán las instrucciones de (Chandel, s.f.), con sus respectivas imágenes para que quede mejor documentado el proceso.

Una vez puesta nuestra interfaz wifi en modo monitor, seleccionaremos la opción 5 (Herramientas de handsake). De aquí, nos llevara a una nueva ventana donde elegiremos nuestra red objetivo.

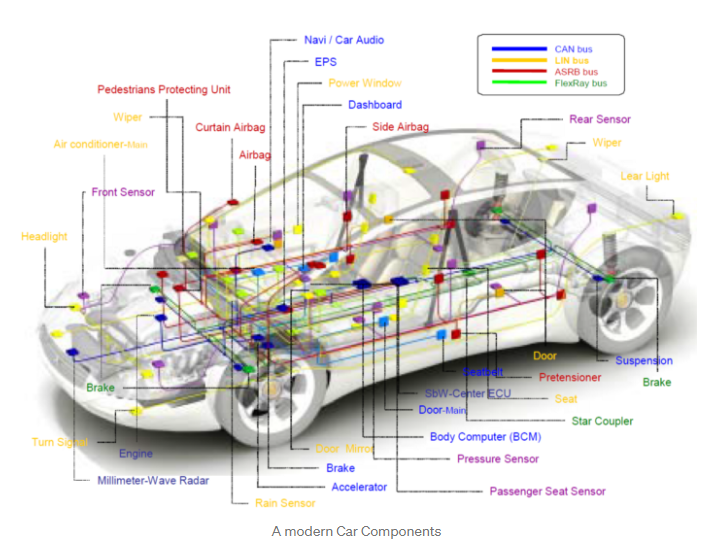
Tras seleccionar la red objetivo, realizaremos un ataque de desauntentificacion, para que todos los dispositivos conectados se tengan que volver a conectar. Esto lo haremos seleccionando la opción 2 (Deauth aireplay attack):



## 3.5. Automitive Tools (CAN)

En este apartado voy a hacer mención especial a un conjunto de herramientas para capturar tráfico de CAN, y poder inyectar paquetes en él.

Para empezar, podemos resumir lo que es CAN, como el bus Controled Area Network (CAN) que conecta y comunica todos los periféricos y utilidades de un coche, algo así como su espina dorsal, el bus donde está todo centralizado (Ojha, s.f.):



Para acceder a este bus, se accede por el conector ODB, que con frecuencia está situado o debajo del volante, o en la parte del copiloto, debajo de la zona de la guantera.

Para esta práctica, ya que no dispongo de un conector USB-ODB, ni tiempo para montar un simulador de comportamiento del bus CAN, seguiré el siguiente estudio realizado por la universidad de North Georgia (Payne, s.f.)

**Instalación y preparación de herramientas**

Primero, necesitamos unas herramientas que nos permitan actuar con el bus CAN. Este conjunto de herramientas se llaman **CAN Utilities**, y vienen preinstaladas por defecto en nuestra distribución Parrot.

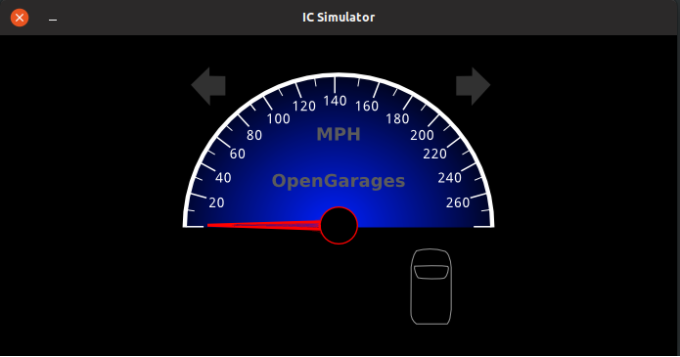
Por otra parte, se va a narrar también como montar la simulación de conexión a un canal CAN, y esto es con la herramienta **ICSim**, creada por el hacker Craigh Smith.

Esta herramienta se puede clonar de su repositorio github:

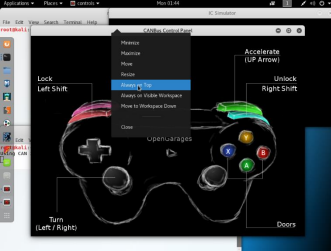
git clone <https://github.com/zombieCraig/ICSim.git>

Una vez instalada ICSim, correremos el comando setup\_vcan.sh, lo que hará que se nos conecte la maquina a una nueva red, llamada vcan0. Esto lo podemos comprobar con el comando ifconfig.

Cuando ya tengamos conectada la nueva red, podemos correr la aplicación. Abriremos tres terminales. En el primero ejecutaremos el comando ~/ICSim/icsim vcan0, Para que nos aparezca el **cuadro de mandos** de nuestro coche simulado:



En el segundo terminal, ejecutaremos el comando ~/ICSim/controls vcan0 para **bindear los controles a nuestro coche virtual**. Por defecto, se usa un mando de PS3 para emular el acelerar, frenar, etc.



Por último, en el tercer terminal, **abriremos el sniffer de paquetes** de nuestras canutilities, con el comando cansniffer -c vcan0.

Este tendrá una interfaz similar a wireshark.

**Capturando y reproduciendo los paquetes CAN**

Primero de todo, tenemos que pulsar ctrl-c para que el sniffer pare su trabajo.

A continuación, utilizaremos la herramienta **candump** para hacer un log de todos los paquetes:

candump -l vcan0

Con este comando, crearemos un log de todos los paquetes que hemos conseguido capturar con el sniffer. Este log se guardará en el directorio en el que estamos con un nombre del estilo:

candump-2021-05-24\_083845.log

Para reproducir los paquetes capturados, simplemente tenemos que hacer uso de la herramienta **canplayer**. Para ello, correremos el siguiente comando:

canplayer -I candump-2021-05-24\_083845.log

Esto inyectara los paquetes que habíamos capturado anteriormente en el bus CAN, ejecutando todo lo que se había capturado con anterioridad.

**Ingeniería inversa para encender los intermitentes**

Ahora vamos a ver como hacer, con todo lo que hemos visto previamente, para conseguir inyectar paquetes que enciendan y apaguen los retrovisores. El ejemplo lo tomaremos de la misma fuente (Payne, s.f.).

Primero tenemos que **descubrir como son los paquetes que encienden las luces**. Para ello, iniciaremos un sinfín, encenderemos un intermitente, e inmediatamente lo pararemos, para no tener mucho paquetes “basura”. Esto lo guardaremos en un log con el candump.

Para asegurarnos de que vamos bien hasta aquí, reproduciremos el log con canplayer, para ver si en verdad hemos capturado el encendido del intermitente.

Si lo hemos conseguido, el siguiente paso es saber **cual de todos los paquetes que hemos capturado es el de encender la luz**.

En una captura normal, capturaremos una media de 10.000 paquetes, por lo que es inviable probarlos uno por uno. Para ello, haremos uso de la herramienta Split de linux.

Cogeremos el log, y lo dividiremos en dos de la siguiente manera:

split -l 5000 candump-2019-03-07\_102721.log x1

Con esto, conseguiremos dos logs diferentes, uno con la primera mitad de los paquetes, y otro con la segunda mitad, que tendrán el nombre x1aa y x1ab.

A continuación, los reproduciremos para saber cuan de los dos contiene el encendido de los intermitentes:

canplayer -I x1aa

canplayer -I x1ab

NOTA: El encender la luz de intermitentes es algo muy simple (podemos considerarlo un booleano: encendido/apagado). Si lo que estamos intentando sacar es, por ejemplo, la aceleración, puede que sea mas sutil, por lo que se recomienda meter el canplayer en un bucle para que se ejecute durante un tiempo:

for i in {1..10}; do canplayer -I x1aa ; done

Por último, **resetearemos el bus can para que no haya interferencias**:

canplayer -I baseline

A partir de aquí, seguiremos con esta búsqueda binaria hasta **encontrar la línea que activa el intermitente**.

En el ejemplo, el paquete activador es el siguiente:

(1551972443.099502) vcan0 188#02000000

El siguiente paso es comprobar a ver si es cierto que enciende las luces, haciendo:

cansend vcan0 188#02000000

Y vemos que enciende la luz derecha.

Si utilizamos un poco de lógica, podremos intuir que si esa secuencia enciende el intermitente derecho, quizás la secuencia 188#01000000 encienda la izquierda, o 188#00000000 apaque ambos intermitentes.

**Probarlo en un coche real**

A partir de aquí, solo queda probar este procedimiento en un coche real. Para ello necesitaremos un cable ODB-USB:



Instalar sus drivers, y proceder con la misma técnica que hemos ido explicando.

# 4. Conclusión y opinión

Sinceramente, este es el trabajo que mas me he divertido haciendo a lo largo de toda la carrera, y me ha parecido muy interesante. Por otra parte, opino que hemos tenido muy poco tiempo (y poca preparación) para usar todas estas herramientas, pero aun así ha sido muy entretenido aprender a usar algunas de ellas, aunque sea solo por encima.

Me ha ayudado a darme cuenta de lo mucho que me gusta el mundo de la seguridad informática, y, sin duda, muchas de las herramientas que no he tenido tiempo de explotar en profundidad, como pueden ser el metasploit y las CAN Utilities, me dedicare este verano a jugar con ellas.

También hay que he tener cuenta que no he podido probar tantas herramientas como me gustaría. Por ejemplo, wireshark, que aunque me parecía importante, ya tuvimos una asignatura entera sobre ella el año pasado, y me parecía invertir mal el tiempo.

# 5. bibliografía

# Bibliografía

Atareao. (s.f.). *atareao.es*. Obtenido de https://atareao.es/software/seguridad/sirikali/

Chandel, R. (s.f.). *HackingArticles*. Obtenido de https://www.hackingarticles.in/hack-wireless-network-using-airgeddon/

Cisofy. (s.f.). *cisofy.com*. Obtenido de https://cisofy.com/lynis/

EcryptPad Documentation. (s.f.). *evpo.net*. Obtenido de https://evpo.net/encryptpad/

Haas, J. (19 de Febrero de 2020). *Lifewire.com*. Obtenido de https://www.lifewire.com/shred-linux-command-4094148

HackTheBox. (s.f.). *HackTheBox*. Obtenido de https://help.hackthebox.eu/machines-challenges/v2-pwnbox

Obbayi, L. (s.f.). Obtenido de https://resources.infosecinstitute.com/topic/how-to-attack-windows-10-machine-with-metasploit-on-kali-linux/

Ojha, Y. (s.f.). *medium*. Obtenido de https://medium.com/@yogeshojha/car-hacking-101-practical-guide-to-exploiting-can-bus-using-instrument-cluster-simulator-part-i-cd88d3eb4a53

Otieno, J. (s.f.). *Linuxhint*. Obtenido de https://linuxhint.com/install-openvas-kali-linux/

Parrot Documentation. (s.f.). *Parrot*. Obtenido de https://docs.parrotlinux.org/library/anonsurf/

Parrot. (s.f.). *Parrotsec*. Obtenido de https://parrotsec.org/download/

Payne, B. R. (s.f.). *University of Georgia*. Obtenido de https://digitalcommons.kennesaw.edu/cgi/viewcontent.cgi?article=1045&context=jcerp

pentestmonkey. (s.f.). Obtenido de https://github.com/pentestmonkey/unix-privesc-check

TEHG. (s.f.). *EthicalHackingGuru*. Obtenido de https://ethicalhackingguru.com/openvas-tutorial-how-to-use-openvas/

tutorialesPoint. (s.f.). Obtenido de https://www.tutorialspoint.com/metasploit/metasploit\_basic\_commands.htm

Velasco, R. (24 de Marzo de 2020). *Softzone*. Obtenido de https://www.softzone.es/programas/linux/parrot-os/