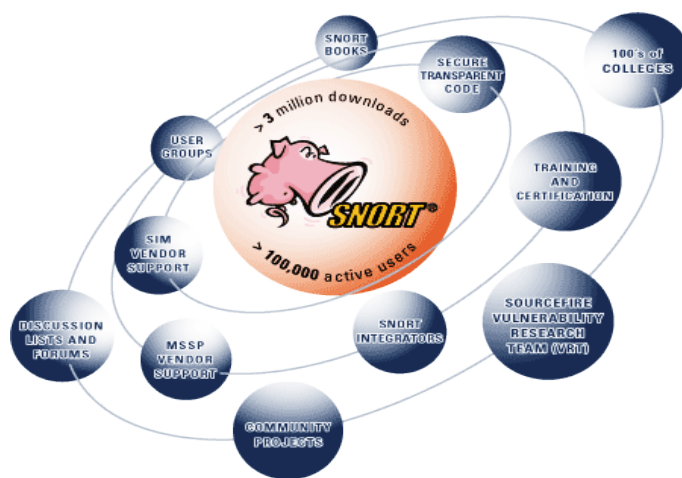


# Snort:

## Análisis de la herramienta

### Visión práctica



Alumnos:

Miguel Ángel Rodríguez García

Miguel Ángel Orenes Fernández

Profesores:

Gabriel López Millán

Gregorio Martínez Pérez

# Índice

Introducción.....	3
1. ¿Qué es Snort?.....	4
2. Instalación y configuración de Snort.....	5
2.1. Manejo de las reglas.....	6
3. Instalación y configuración de MySQL.....	12
4. Instalación y configuración de ACID.....	14
5. Puesta en marcha de la herramienta.....	22
6. Interpretación y manejo de los resultados, ejemplo práctico.....	23
7. Ataques Remotos.....	27
7.1. Escaneo de puertos.....	27
7.2. Spoofing.....	29
7.3. Negociación de servicios.....	31
7.4. Interceptación.....	33
7.5. Ataques de aplicaciones.....	34
7.5.1. Correo Electrónico.....	34
7.5.2. Ataques vía web.....	35
7.6 Ejemplo de ataque remotos (Nessus).....	36
8. Conclusión.....	42

# Introducción

Para la realización de este trabajo hemos instalado la herramienta Snort sobre el sistema operativo Windows.

El trabajo está partido en tres partes bien definidas. La primera es el capítulo 1, en el que hemos querido dar una breve descripción de lo que es Snort, así como mostrar sus principales utilidades. La segunda, consta de los capítulos del 2 al 6, y en ella se muestra la instalación de Snort, MySQL y ACID para un entorno Windows. El hecho de decantarnos por realizar el trabajo sobre Windows, es debido a que en la web se puede encontrar mucha literatura sobre los pormayores y pormenores de la instalación de Snort para Linux, así como cursos de gestión de redes UNIX, mientras que de Windows no se encuentra gran cosa. Así pues hemos creído conveniente crear una tutorial en el que se expliquen con detalle los cambios necesarios en ficheros de configuración, así como los comandos a introducir en la consola.

Por último la tercera parte, consta del capítulo 7. En este capítulo mostramos los más famosos ataques remotos que se pueden realizar a una máquina, metodologías que siguen, las vulnerabilidades que aprovechan, y distintos métodos que podemos utilizar para evitarlos.

Todas las herramientas que se han utilizado son Software Libre, y se pueden descargar de sus páginas web. Iremos dando las instrucciones de descarga, así como los sitios desde los que se pueden descargar según vayan haciendo falta.

Creemos que toda la ayuda para comenzar a utilizar Snort queda bien plasmada en el documento, no obstante, la web <http://www.snort.org> tiene una gran fuente de ayudas, foros, y documentación que podrá servir de ayuda para los interesados en los IDS.

# 1. ¿Qué es Snort?

Snort es un sniffer de paquetes y un detector de intrusos basado en red (se monitoriza todo un dominio de colisión). Es un software muy flexible que ofrece capacidades de almacenamiento de sus bitácoras tanto en archivos de texto como en bases de datos abiertas como lo es MySQL.

Implementa un motor de detección de ataques y barrido de puertos que permite registrar, alertar y responder ante cualquier anomalía previamente definida. Así mismo existen herramientas de terceros para mostrar informes en tiempo real (ACID) o para convertirlo en un Sistema Detector y Preventor de Intrusos.

Este IDS implementa un lenguaje de creación de reglas flexible, potente y sencillo. Durante su instalación ya nos provee de cientos de filtros o reglas para backdoor, DDoS, finger, FTP, ataques web, CGI, Nmap...

Puede funcionar como sniffer (podemos ver en consola y en tiempo real qué ocurre en nuestra red, todo nuestro tráfico), registro de paquetes (permite guardar en un archivo los logs para su posterior análisis, un análisis offline) o como un IDS normal (en este caso NIDS). Cuando un paquete coincide con algún patrón establecido en las reglas de configuración, se logea. Así se sabe cuando, de donde y cómo se produjo el ataque.

Snort está disponible bajo licencia GPL, gratuito y funciona bajo plataformas Windows y UNIX/Linux. Dispone de una gran cantidad de filtros o patrones ya predefinidos, así como actualizaciones constantes ante casos de ataques, barridos o vulnerabilidades que vayan siendo detectadas a través de los distintos boletines de seguridad.

La característica más apreciada de Snort, además de su funcionalidad, es su subsistema flexible de firmas de ataques. Snort tiene una base de datos de ataques que se está actualizando constantemente y a la cual se puede añadir o actualizar a través de la Internet. Los usuarios pueden crear 'firmas' basadas en las características de los nuevos ataques de red y enviarlas a la lista de correo de firmas de Snort, para que así todos los usuarios de Snort se puedan beneficiar. Esta ética de comunidad y compartir ha convertido a Snort en uno de los IDSes basados en red más populares, actualizados y robustos.

Podemos señalar que Snort es una herramienta que a pleno funcionamiento consume muchos recursos. Por este motivo, y debido al echo de que estamos monitorizando una sola máquina, hemos decidido para este trabajo utilizar la base de datos de reglas que no necesita registro por internet, si no que se crea en el mismo momento de la creación de la versión que utilizaremos de Snort, con el fin de no cargar las máquinas de los compañeros que quieran instalar Snort para probarlo en sus PCs. No obstante, se darán las directrices a seguir por si se desea instalar el NIDS en un equipo para que monitorice con las mayores prestaciones su red, descargando las reglas registrándose en <http://www.snort.org>.

## 2. Instalación y configuración de Snort

Para comenzar , necesitamos la herramienta WinCap3.1 descargable desde <http://www.wincap.org/install/default.htm>. Esta herramienta es básica para el funcionamiento de Snort, debido a que es la librería de bajo nivel que utilizará Snort para el acceso a redes. Es para Windows como la libCap utilizada en Linux.

Una vez instalado WinCap, debemos acceder a la web de Snort, <http://www.snort.com> y descargarnos la versión de Snort para Windows desde <http://www.snort.org/dl/binaries/win32/> y pasamos a instalarlo. Lo único que hay que subrayar en el proceso de instalación, es que en un momento dado nos preguntará el wizard que seleccionemos las opciones de configuración, seleccionamos **“I do not plan to log to a database, or I am planning to log to one of the databases listed above”** y presionamos next para continuar. A continuación nos pedirá que seleccionemos los componentes de la instalación, los seleccionamos todos y hacemos click en next. Cuando nos pida la ruta de instalación, dejamos la que sle por defecto **C:\Snort** y presionamos next para finalizar la instalación.

Una vez que está instalado Snort, tenemos que proseguir con su configuración. Lo primero que tenemos que hacer es acceder a la carpeta **C:\Snort\etc** en este directorio encontraremos un fichero llamado **snort.conf** que es el fichero de configuración que utilizaremos para configurar Snort. Accedemos al fichero con un editor de texto que no corrompa el formato original del archivo (notepad o wordpad).

Entonces comenzamos la configuración:

### 1. Comenzamos con la configuración de la red

Como podemos ver, en snort.conf, la red que aparece por defecto

```
var HOME_NET any
```

Lo que tenemos que hacer es modificar esta variable. La podemos modificar de tres modos dependiendo de lo que queramos:

1. Una red C:           var HOME\_NET 192.168.1.0/24
2. Host específico :   var HOME\_NET 192.168.1.3/32
3. Varios Host:       var HOME\_NET 192.168.1.2/32,192.168.1.3/32,192.168.1.4/32

En nuestro caso lo que nos interesa es monitorizar un solo host, con lo que utilizamos el segundo modo, en nuestro caso será:

```
var HOME_NET IpdelHost/32
```

La Ip del Host se puede obtener ejecutando en comando **ipconfig** en la consola.

### 2. Seguimos con la configuración de las reglas

Para este punto lo único que tenemos que hacer es descargarnos las reglas de la web de Snort e incluirlas en el directorio **c:\Snort\rules**.

*Como se observará en la web de Snort, hay tres conjuntos de reglas para descargar, **Subscribers** , **Registered** y **Unregistered**. Para este tutorial recomendamos descargar las **Unregistered** ya que cargarán menos el PC. Una vez que lo finalicemos, si se desea dejar instalado Snort y utilizarlo como servicio, es aconsejable registrarse para que podamos recibir las actualizaciones que se van realizando de las reglas.*

En snort.conf, casi al final aparecen una serie de sentencias con el siguiente formato:

```
include $RULE_PATH/name.rules
```

Se trata de las sentencias que incluyen las diferentes librerías de reglas. Las que tienen una '#' delante son las que están comentadas, si queremos que se incluyan, solamente tenemos que quitarle la almohadilla de delante.

Por ahora le quitamos la almohadilla a todas.

### 3. Finalizamos con unos retoques de acceso

Solamente nos queda por modificar lo siguiente:

Cambiar la línea donde aparece :

```
dynamicpreprocessor directory /usr/local/lib/snort_dynamicpreprocessor/
```

por:

```
dynamicpreprocessor directory c:\snort\lib\snort_dynamicpreprocessor
```

Y la línea que pone:

```
dynamicengine /usr/local/lib/snort_dynamicengine/libsf_engine.so
```

Por:

```
dynamicengine c:\snort\lib\snort_dynamicengine\sf_engine.dll
```

Una vez realizados estos cambios podemos probar Snort desde la línea de comandos. Accedemos a la carpeta c:\Snort\bin y este directorio escribimos :

```
C:\Snort\bin> snort -dev -c c:\Snort\etc\snort.conf -l C:\Snort\log -i2
```

Nos dará un error diciendo que no se encuentra el archivo **spyware-put.rules**. Comentamos esa línea en el fichero de configuración y volvemos a ejecutar. Comenzaremos a ver el tráfico que pasa por nuestro equipo. Si observamos el directorio C:\Snort\log, podemos ver como en el fichero **alert.ids** se han ido almacenando las alertas que los ficheros de reglas tienen reconocidas como tráfico del que deben de alertar.

Las opciones que le hemos pasado en la línea de comandos a Snort son:

-d : visualizar los campos de datos que pasan por la interface de red.

-e: snort nos mostrará información más detallada.

-v: Iniciamos snort en modo sniffer visualizando en pantalla las cabeceras de los paquetes TCP/IP.

-c: archivo que utilizará snort como fichero de configuración.

-l: directorio donde guardar las alertas y logs.

-i: interfaz que monitorizaremos.

## 2.1. Manejo de las reglas

*Las opciones que aquí se comentan para la creación de reglas, son solamente para el comienzo de la utilización de Snort, puede encontrar toda la documentación necesaria sobre reglas en la web [http://www.snort.org/docs/writing\\_rules](http://www.snort.org/docs/writing_rules)*

La herramienta Snort utiliza un lenguaje simple para la definición de las reglas. La mayoría de las reglas están escritas en una sola línea. Ahora pasamos a describir los pasos que tenemos que

dar para crear un fichero con reglas nuevas que queramos utilizar.

El primer paso será crear el fichero “.rules” ( lo llamaremos misReglas.rules), en el que introduciremos nuestras reglas, en el directorio **rules** alojado en el directorio raíz de la herramienta(c:\Snort\rules). Una vez creado nos vamos al fichero de configuración, en este tutorial estamos utilizando **snort.conf** del directorio **etc** y después de la inclusión de los ficheros de reglas que nos proporciona snort, introducimos una sentencia del tipo:

```
include <fichero de reglas>
```

Que en nuestro caso será:

```
include $RULE_PATH/misReglas.rules
```

Con esta sentencia, le estamos diciendo a snort que, cuando arranque, introduzca las reglas del fichero misReglas.rules en su base de datos de reglas.

Lo que tenemos que hacer a continuación es acceder al fichero en el que se clasifican y se priorizan las reglas, en el cual, introduciremos el nuevo tipo del conjunto de reglas que vamos a crear, y le asignaremos prioridad. Este fichero se llama **classification.config** y se encuentra en la carpeta **etc**, la misma que el fichero de configuración de Snort.

Como se puede observar, las diferentes clasificaciones de reglas siguen el siguiente patrón:

```
config <directiva>: name, descripción, prioridad
```

En nuestro caso crearemos una clasificación (directiva **classification**) de reglas que llamaremos **user-rules**, cuya descripción será *tráfico que nos interesa*, y cuya prioridad será 5. Para crearla, introducimos la siguiente línea en el fichero:

```
config classification: user-rules, Trafico que nos interesa, 5
```

También se podrán utilizar las clasificaciones ya creadas, solamente creamos en el ejemplo una nueva clasificación para que se vea como se crearía.

Una vez que hemos realizado esto, solamente nos queda comenzar a crear las reglas que nos interesan. Es importante comentar que para la creación de reglas *medio decentes* se necesita unos pequeños conocimientos sobre protocolos, como pueden ser ICMP, TCP, ... debido a que a las reglas se le pasan parámetros en si parte de opciones.

Vamos a crear una regla muy simple, que nos va a alertar de los **ping** que se realizan desde una máquina de nuestra red hacia el exterior.

Comenzamos abriendo el fichero misReglas.rules que hemos creado anteriormente, y comenzamos a introducir las reglas. Las reglas en Snort se componen de dos grande partes, la cabecera y las opciones, teniendo las siguiente forma:

```
<cabecera> (<opciones>)
```

### **Cabeceras:**

Las cabeceras siguen en siguiente formato:

```
<acción> <protocolo> <redFuente con máscara> <puerto> --> <redDestino con máscara> <puerto>
```

- La acción que vamos a crear es una acción del tipo **alert**.
- El protocolo será **icmp**.
- La red fuente será la variable **\$HOME\_NET** que modificamos en el fichero de configuración anteriormente, que tiene incluida la máscara de red.
- El puerto será **any**, refiriéndonos a todos los puertos.

- Como red destino ponemos **\$EXTERNAL\_NET**.
- Y como puerto **any**

No obstante, se podría poner una red concreta, y un puerto concreto o rango de puertos. Para indicar un rango de puertos, utilizaremos el operador ':', Ej: 0:1023, puertos del 0 al 1023; :300 , puerto menor o igual a 300; 500:, puerto mayor o igual que 500.

Una vez hecho esto, tenemos que pasar a rellenar las opciones.

### Opciones:

Existe un gran número de opciones, solamente comentaremos las más importantes para nuestro cometido:

- **msg:** Escribe un mensaje de alerta.
- **itype:** tipo icmp.
- **icode:** código icmp.
- **flags:** flags tcp (A, S, F, U, R y P)
- **sid:** ID de la regla.
- **classtype:** tipo de la regla, ( el tipo creado en el fichero classification.config)
- **content:** buscan un patrón en el contenido de los datos del paquete.
- **rev:** numero de revisión de la regla.

Una vez comentado esto, sabemos que opciones utilizaremos, serán msg, itype, icode, sid, rev y classtype.

El código de un PING en el protocolo icmp es 0, y el tipo es 8, con lo cual el campo icode tendrá el valor 0 y el itype el 8.

Para finalizar introducimos la regla en el fichero **misReglas.rules**. La regla será la siguiente:

```
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"PING que se lanza desde mi máquina"; icode:0; itype:8; sid:10000;classtype:user-rules; rev:0;)
```

Ahora solamente tenemos que ejecutar Snort:

```
C:\Snort\bin> snort -dev -c c:\Snort\etc\snort.conf -l C:\Snort\log -i2
```

Realizamos un ping desde la línea de comandos a cualquier máquina, por ejemplo:

```
ping 155.54.1.1
```

Y en el fichero **alert.ids** podemos ver que se ha incluido el siguiente contenido:

```
[**] [1:10000:0] PING que se lanza desde mi máquina [**]
[Classification: Trafico que nos interesa] [Priority: 5]
01/05-18:46:31.958312 0:5:9A:3C:78:0 -> 0:6:52:51:B0:1B type:0x800 len:0x4A
155.54.197.42 -> 155.54.1.1 ICMP TTL:128 TOS:0x0 ID:6465 IpLen:20 DgmLen:60
Type:8 Code:0 ID:1024 Seq:768 ECHO
```

[\*\*] [1:408:5] ICMP Echo Reply [\*\*]  
[Classification: Misc activity] [Priority: 3]  
01/05-18:46:31.961734 0:6:52:51:B0:1B -> 0:5:9A:3C:78:0 type:0x800 len:0x4A  
155.54.1.1 -> 155.54.197.42 ICMP TTL:253 TOS:0x0 ID:53966 IpLen:20 DgmLen:60 DF  
Type:0 Code:0 ID:1024 Seq:768 ECHO REPLY

[\*\*] [1:10000:0] PING que se lanza desde mi máquina [\*\*]  
[Classification: Trafico que nos interesa] [Priority: 5]  
01/05-18:46:32.958729 0:5:9A:3C:78:0 -> 0:6:52:51:B0:1B type:0x800 len:0x4A  
155.54.197.42 -> 155.54.1.1 ICMP TTL:128 TOS:0x0 ID:6549 IpLen:20 DgmLen:60  
Type:8 Code:0 ID:1024 Seq:1024 ECHO

[\*\*] [1:408:5] ICMP Echo Reply [\*\*]  
[Classification: Misc activity] [Priority: 3]  
01/05-18:46:32.961565 0:6:52:51:B0:1B -> 0:5:9A:3C:78:0 type:0x800 len:0x4A  
155.54.1.1 -> 155.54.197.42 ICMP TTL:253 TOS:0x0 ID:53967 IpLen:20 DgmLen:60 DF  
Type:0 Code:0 ID:1024 Seq:1024 ECHO REPLY

[\*\*] [1:10000:0] PING que se lanza desde mi máquina [\*\*]  
[Classification: Trafico que nos interesa] [Priority: 5]  
01/05-18:46:33.959755 0:5:9A:3C:78:0 -> 0:6:52:51:B0:1B type:0x800 len:0x4A  
155.54.197.42 -> 155.54.1.1 ICMP TTL:128 TOS:0x0 ID:6614 IpLen:20 DgmLen:60  
Type:8 Code:0 ID:1024 Seq:1280 ECHO

[\*\*] [1:408:5] ICMP Echo Reply [\*\*]  
[Classification: Misc activity] [Priority: 3]  
01/05-18:46:33.962559 0:6:52:51:B0:1B -> 0:5:9A:3C:78:0 type:0x800 len:0x4A  
155.54.1.1 -> 155.54.197.42 ICMP TTL:253 TOS:0x0 ID:53968 IpLen:20 DgmLen:60 DF  
Type:0 Code:0 ID:1024 Seq:1280 ECHO REPLY

[\*\*] [1:10000:0] PING que se lanza desde mi máquina [\*\*]  
[Classification: Trafico que nos interesa] [Priority: 5]  
01/05-18:46:34.960773 0:5:9A:3C:78:0 -> 0:6:52:51:B0:1B type:0x800 len:0x4A  
155.54.197.42 -> 155.54.1.1 ICMP TTL:128 TOS:0x0 ID:6679 IpLen:20 DgmLen:60  
Type:8 Code:0 ID:1024 Seq:1536 ECHO

[\*\*] [1:408:5] ICMP Echo Reply [\*\*]  
[Classification: Misc activity] [Priority: 3]  
01/05-18:46:34.965188 0:6:52:51:B0:1B -> 0:5:9A:3C:78:0 type:0x800 len:0x4A  
155.54.1.1 -> 155.54.197.42 ICMP TTL:253 TOS:0x0 ID:53969 IpLen:20 DgmLen:60 DF

Type:0 Code:0 ID:1024 Seq:1536 ECHO REPLY

Como se puede observar, se han incluido las alertas de los ECHOs que nosotros hemos lanzado, con el código que le hemos asignado, el número de revisión, el texto que hemos puesto para que se imprimiera, el grupo de clasificación al que pertenece y su prioridad, seguido del contenido de la cabecera del paquete. Además, se crea la alerta también de la respuesta al ECHO.

Supongamos que deseamos que no se crease la alerta de los ECHO REPLY, lo único que tendríamos que hacer sería comentar la entrada en el fichero **icmp-info.rules** que se hace cargo de crear la alerta de los ECHO REPLY:

**alert icmp \$EXTERNAL\_NET any -> \$HOME\_NET any (msg:"ICMP Echo Reply"; icode:0; itype:0; classtype:misc-activity; sid:408; rev:5;)**

```
[**] [1:10000:0] PING que se lanza desde mi máquina [**]  
[Classification: Trafico que nos interesa] [Priority: 5]  
01/08-09:12:34.318655 0:5:9A:3C:78:0 -> 0:6:52:51:B0:1B type:0x800 len:0x4A  
155.54.197.245 -> 155.54.1.1 ICMP TTL:128 TOS:0x0 ID:6786 IpLen:20 DgmLen:60  
Type:8 Code:0 ID:1024 Seq:3072 ECHO
```

```
[**] [1:10000:0] PING que se lanza desde mi máquina [**]  
[Classification: Trafico que nos interesa] [Priority: 5]  
01/08-09:12:35.319667 0:5:9A:3C:78:0 -> 0:6:52:51:B0:1B type:0x800 len:0x4A  
155.54.197.245 -> 155.54.1.1 ICMP TTL:128 TOS:0x0 ID:6795 IpLen:20 DgmLen:60  
Type:8 Code:0 ID:1024 Seq:3328 ECHO
```

```
[**] [1:10000:0] PING que se lanza desde mi máquina [**]  
[Classification: Trafico que nos interesa] [Priority: 5]  
01/08-09:12:36.320685 0:5:9A:3C:78:0 -> 0:6:52:51:B0:1B type:0x800 len:0x4A  
155.54.197.245 -> 155.54.1.1 ICMP TTL:128 TOS:0x0 ID:6804 IpLen:20 DgmLen:60  
Type:8 Code:0 ID:1024 Seq:3584 ECHO
```

```
[**] [1:10000:0] PING que se lanza desde mi máquina [**]  
[Classification: Trafico que nos interesa] [Priority: 5]  
01/08-09:12:37.320725 0:5:9A:3C:78:0 -> 0:6:52:51:B0:1B type:0x800 len:0x4A  
155.54.197.245 -> 155.54.1.1 ICMP TTL:128 TOS:0x0 ID:6813 IpLen:20 DgmLen:60  
Type:8 Code:0 ID:1024 Seq:3840 ECHO
```

Como se puede observar, a comentar la línea que provoca las alarmas de los ECHO REPLY, snort no almacena las alertas de se han producido de este tráfico.

Por otro lado, también se pueden crear reglas que tras su invocación hagan que otras reglas

sean activadas durante un periodo de tiempo. Esto se consigue mediante las siguientes cláusulas:

- **activate**: Alerta y activa una regla declarada como dinámica (dynamic).
- **dynamic**: cobra sentido cuando una regla declarada como activa (activate), la activa de modo que pasa a formar parte de las reglas del registro.

El par activate/dynamic rules, le proporciona a Snort una capacidad con mucha potencia, debido a que le capacita para que una regla esté activa a partir de un determinado momento, y que vuelva a ser pasiva cuando haya recibido un número de paquetes que se le indica. Las reglas **activate** se declaran exactamente igual que las **alert**, con la salvedad de que necesitan de una opción adicional, la opción **activates**. Las reglas **dynamic** son en formato igual que las reglas de registro, con la salvedad de que necesitan de dos opciones más, **activated\_by** y **count**.

Por ejemplo, las siguientes sentencias:

```
activate tcp !$HOME_NET any -> $HOME_NET 143 (flags: PA; \
content: "|E8C0FFFFFF|/bin"; activates: 1; \
msg: "IMAP buffer overflow!\");
dynamic tcp !$HOME_NET any -> $HOME_NET 143 (activated_by: 1; count: 50;)
```

Estas reglas hará que cuando se detecte un overflow del búfer IMAP, se coleccionen los 50 siguientes paquetes que vienen desde una red exterior a la red que estamos vigilando por el puerto 143.

Existe la posibilidad de que una regla implemente respuestas ante la activación de una alerta, la ejecución de estas respuestas hace que las conexiones que se interpreten ofensivas sean cerradas. Las respuestas que nos podemos encontrar son las siguientes:

- **rst\_snd** : envía un paquete TCP-RST por el socket emisor.
- **rst\_rcv** : envía un paquete TCP-RST por el socket receptor.
- **rst\_all** : envía un paquete TCP-RST en ambas direcciones.
- **icmp\_net** : envía un ICMP\_NET\_UNREACH al emisor.
- **icmp\_host**: envía un ICMP\_HOST\_UNREACH al emisor.
- **icmp\_port**: envía ICMP\_PORT\_UNREACH al receptor .
- **icmp\_all**: send all above ICMP packets to the sender

Todas estas opciones pueden ser combinadas siguiendo el siguiente formato:

**resp:<modificación\_respuesta[, modificación\_respuesta];>**

```
alert udp any any -> 192.168.1.0/24 31 (resp: icmp_port,icmp_host; \
msg: "Hacker's Paradise access attempt");
```

### 3. Instalación y configuración de MySQL

Lo primero que tenemos que hacer es descargar el driver ODBC debido a los problemas de compatibilidad que se pueden encontrar, el nombre del archivo es mysql-connector-odbc-3.51.12-win32, y se puede descargar desde <http://dev.mysql.com/downloads/connector/odbc/3.51.html>. Se instala y proseguimos con la instalación de MySQL.

Descargamos la base de datos MySQL de la página oficial de MySQL, <http://www.mysql.org>, se instala y pasamos a configurar tanto Snort como MySQL para que se puedan utilizar y para que ambos interactúen para poder trabajar juntos.

Una vez instalado MySQL, tenemos que crear la base de datos sobre la que trabajará. Para ello accedemos como root a la base de datos. Esto se hace desde la carpeta **bin** del directorio raíz de MySQL mediante el comando:

```
mysql -u root -p
```

Una vez hecho esto, nos pedirá la clave del root.

Acto seguido, una vez dentro de mysql, tenemos que crear la base de datos:

```
create database snort;
```

Ahora tenemos que crear las tablas con las que trabajará snort, para eso, se necesita un fichero llamado **create\_mysql**, para la realización de este tutorial, fue descargado de un comentario del foro del sitio oficial de Snort, el comentario en cuestión está en <http://www.snort.org/archive-11-3647.html>. Creamos el archivo create\_mysql en la carpeta schemas contenida en el directorio raíz de Snort.

*Nosotros modificamos una línea del fichero de creación de la base de datos, debido a que si no, sería imposible almacenar las alarmas que crea un escaneo de puertos, ya que no pertenece a ninguna clase de reglas. La línea, es una de las pertenecientes a la creación de la tabla **signature**(línea 38), modificada es la siguiente:*

```
sig_class_id INT UNSIGNED NOT NULL,
```

*por*

```
sig_class_id INT UNSIGNED,
```

Una vez hecho esto, desde la línea de comandos ejecutamos:

```
mysql -u root -p -D snort < c:\Snort\schemas\create_mysql
```

Nos pedirá la clave del root, y ya están creadas las tablas necesarias. Ahora tenemos que darle permisos al usuario con el que se modificará la base de datos snort de la siguiente manera (dentro de mysql, habiendo accedido con el root):

```
grant insert,select,update,create,delete on snort.* to User@localhost identified by 'clave';
```

Salimos de mysql, y ahora accedemos con el usuario creado directamente a la base de datos snort:

```
mysql -u User -D snort -p
```

Nos pedirá la clave, y accederemos. Una vez dentro, comprobamos que se han creado todas las tablas mediante el comando

```
show tables;
```

Ahora tenemos que modificar el fichero **snort.conf** para que snort interactúe con mysql. Lo único que tenemos que modificar es descomentar la línea

```
# output database: log, mysql, user=root password=test dbname=db host=localhost
```

Cambiándola por la siguiente:

```
output database: log, mysql, user=User password=PASSWD dbname=snort host=localhost  
port=3306 sensor_name=snort_sensor
```

Una vez realizados estos cambios, lanzamos snort desde el directorio bin del fichero raíz de snort, mediante el comando:

```
snort -dev -c c:\Snort\etc\snort.conf -l C:\Snort\log -i2
```

Realizamos alguna acción para que se creen alertas, y accedemos a la base de datos snort y ejecutamos :

```
select * from event;
```

Y veremos todos los eventos que se han producido.

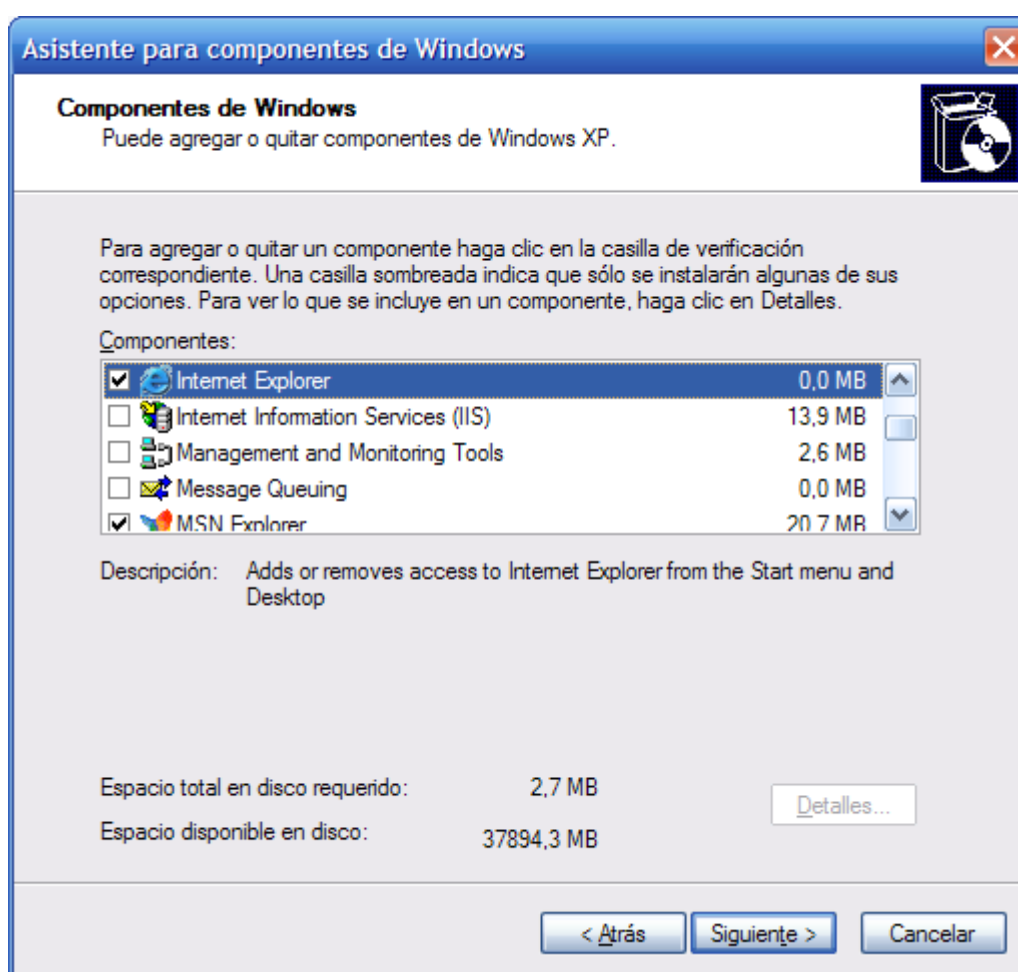
Una vez hecho esto, vemos como el acceso a los datos que nos genera Snort sigue siendo un poco emborroso , ya que tenemos que estar accediendo desde la línea de comandos que en muchas ocasiones se convierte en algo pesa de trabajar. Para paliar esto, contamos con una herramienta llamada ACID que consta de unos ficheros php, y que solamente necesita para ejecutarse un servidor que soporte php, tener php instalado, y una serie de herramientas que se comentaran.

## 4. Instalación y configuración de ACID

ACID es un sistema basado en web, creado con el lenguaje de programación PHP, con lo que necesita de un servidor capaz de interpretar PP.

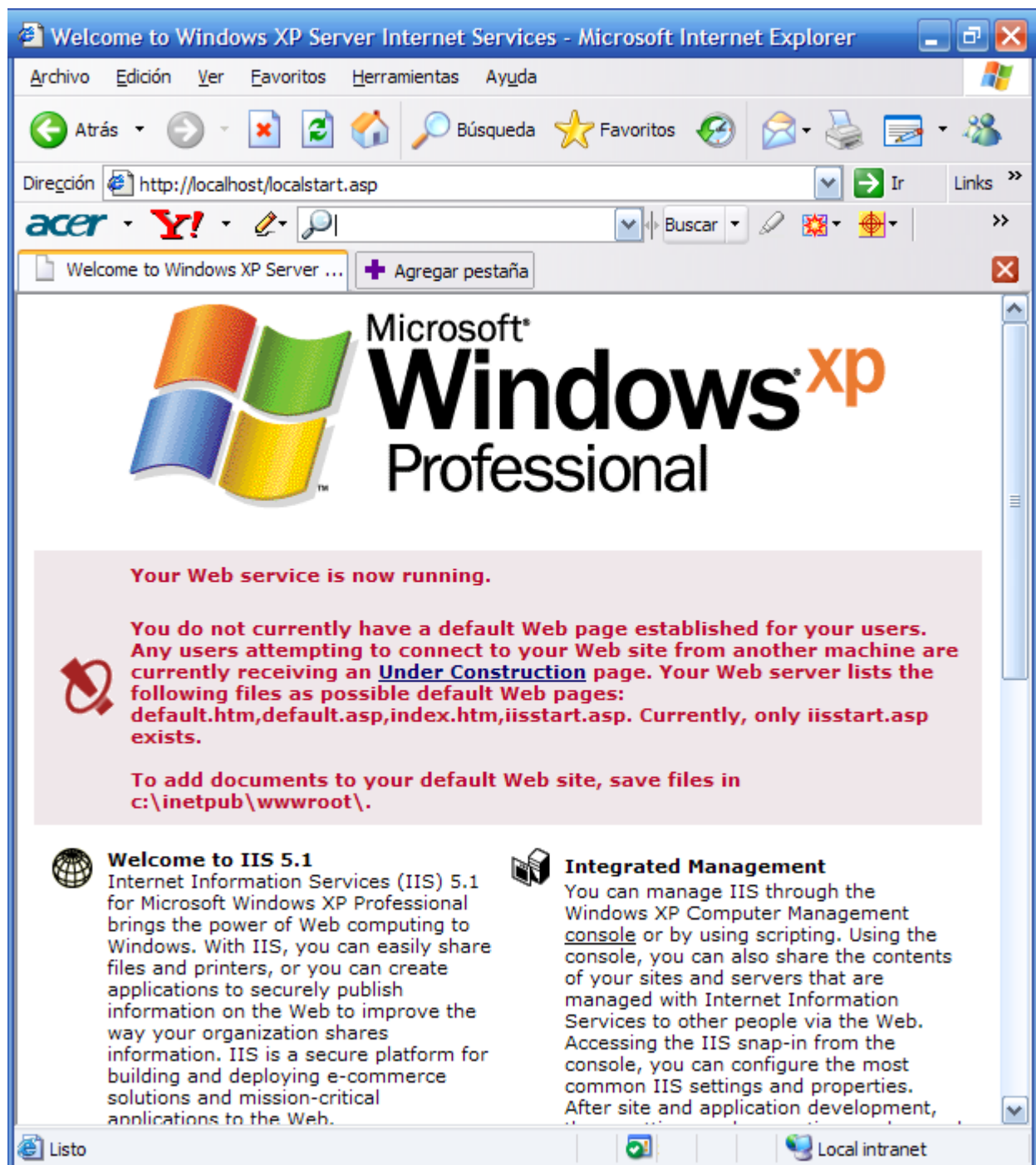
Como estamos instalando Snort en Windows, vamos a utilizar el servidor IIS que nos viene en la distribución de Windows que tenemos instalada ( el tutorial se está realizando sobre Windows Xp). No hay ningún problema en utilizar u Apache, o cualquier servidor que sepa interpretar código PHP, solamente utilizamos este porque creemos que es interesante saber instalarlo, ya que durante la carrera solamente hemos trabajado con servidores Apache y Apache Tomcat y no con IIS.

Para instalar IIS tenemos que acceder al Panel de control --> Agregar o quitar Programas --> Agregar o Quitar Componentes de Windows. Nos aparecerá una ventana como esta:



Solamente tenemos que seleccionar **Internet Information Services(IIS)**, y hacer click en siguiente.

Una vez instalado, probamos que está funcionando escribiendo en el Iexplorer( OJO!!!!!!!!!! solamente se puede acceder desde el Iexplorer, no lo intentéis desde otro navegador, Firefox, Opera por que no os dejará acceder) la dirección *localhost* , y nos tiene que salir la siguiente ventana:

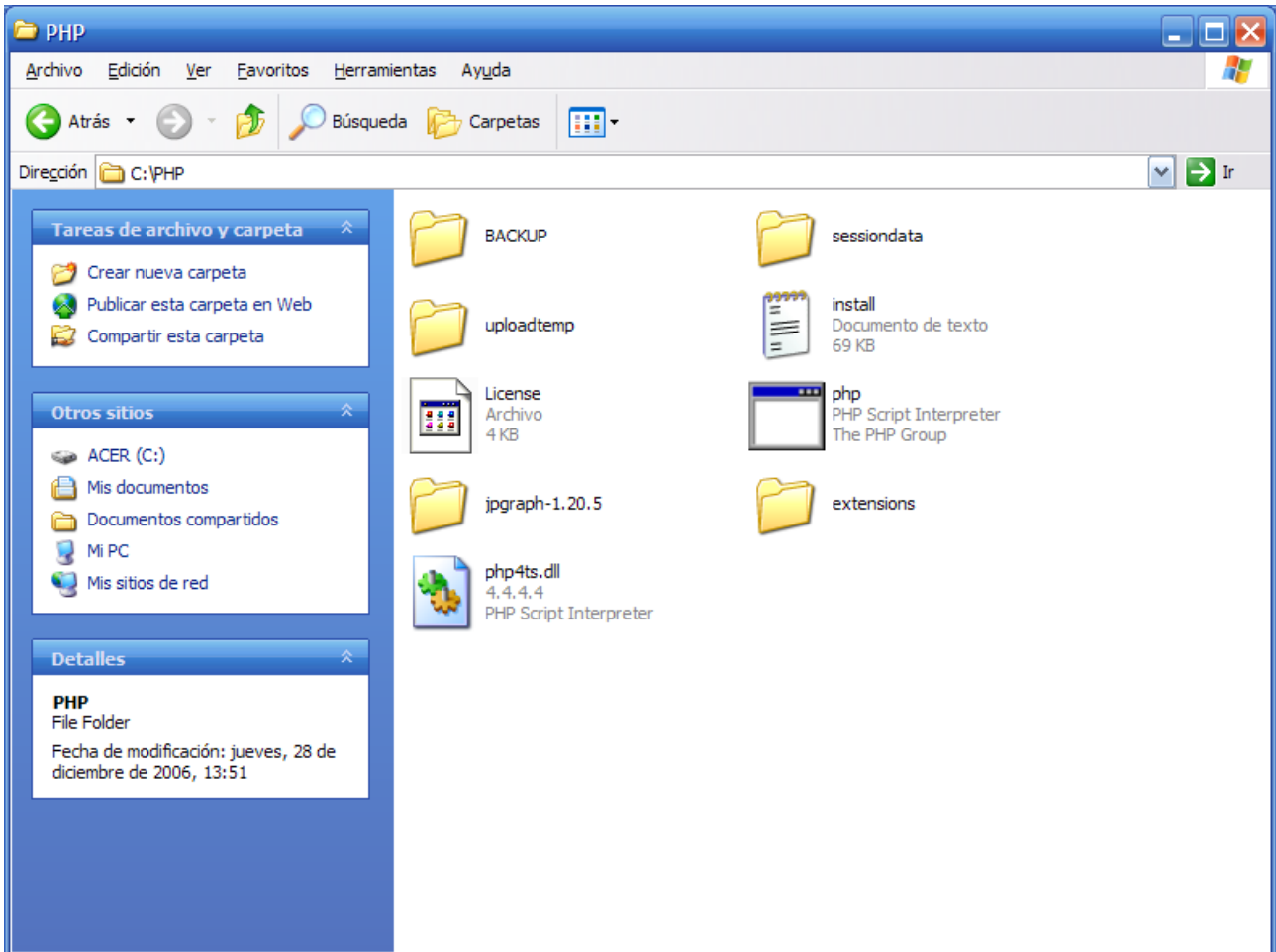


Una vez instalado el IIS, ya tenemos un servidor que nos interprete PHP.

Ahora debemos de instalar PHP. Para ello nos bajamos de [www.php.net](http://www.php.net) tanto el instalador de PHP como el archivo **.zip**. Los dos archivos que tenemos que descargarnos deben ser de la misma versión de PHP, ya que si no tendremos problemas y no podremos ejecutar ACID.

- El zip lo podemos descargar de <http://es.php.net/get/php-4.4.4-Win32.zip/from/a/mirror>
- y el ejecutable de <http://es2.php.net/get/php-4.4.4-installer.exe/from/a/mirror>

Instalamos PHP desde el instalador. En algún momento nos preguntará el instalador que a que servidor queremos darle servicio, elegimos el IIS de la versión que hayamos instalado, en nuestro caso fue *Microsoft IIS 4 or higher*. Una vez acabado esto, accedemos al directorio raíz de PHP, supuestamente estará en [c:\PHP](#). Mientras tanto abrimos el fichero **php-4.4.4-Win32.zip** y extraemos el el directorio *extensions* en el directorio raíz de PHP de modo que quedará así:



Ahora pasamos a configurar el fichero *php.ini* que se encuentra en el directorio [C:\WINDOWS](#). Una vez en el lo editamos y modificamos una serie de variables, dejándolas como quedan:

```
max_execution_time = 60 ; Maximum execution time of each script, in seconds
error_reporting = E_ALL & ~E_NOTICE
extension_dir = c:\php\extensions
extension=php_gd2.dll
session.save_path= C:\WINDOWS\Temp; argument passed to save_handler
```

Echo esto solamente nos queda configurar ACID. Y para esto, antes de nada nos tenemos que bajar dos ficheros:

- adodb452 --> <http://prdownloads.sourceforge.net/adodb/adodb452.zip?download>
- PHPPlot --> [http://sourceforge.net/project/showfiles.php?group\\_id=14653](http://sourceforge.net/project/showfiles.php?group_id=14653)

Con estos archivos lo único que tenemos que hacer es descomprimirlos en el directorio raíz de Snort.

Ahora si que llega el momento de instalar ACID. Lo único que tenemos que hacer es descargar acid-0.9.6b23.tar de <http://acidlab.sourceforge.net/>, lo descomprimos en el directorio raíz del servidor web, que en nuestro caso, por haber instalado IIS, es *C:\Inetpub\wwwroot*, y comenzamos a modificar unas líneas del fichero *acid\_conf.php*:

```
$DBlib_path = "C:\Snort\adodb";

/* Alert DB connection parameters
 * - $alert_dbname : MySQL database name of Snort alert DB
 * - $alert_host : host on which the DB is stored
 * - $alert_port : port on which to access the DB
 * - $alert_user : login to the database with this user, usuario que tiene permisos para conectarse a la base de datos de snort
 * - $alert_password : password of the DB user, clave del usuario
 *
 * This information can be gleaned from the Snort database
 * output plugin configuration.
 */
$alert_dbname = "snort";
$alert_host = "localhost";
$alert_port = "3306";
$alert_user = "USER";
$alert_password = "PASSWD";
```

ACID crea tablas adicionales para que el usuario pueda archivar alertas importantes. Se puede indicar otro usuario para acceder a ellas.

```
/* Archive DB connection parameters */

$archive_dbname = "snort_archive";
$archive_host = "localhost";
$archive_port = "";
$archive_user = "user_archive";
$archive_password = "123456";
```

Con esto ya tenemos instalado ACID, ahora para probarlo ponemos en el Iexplorer <http://localhost/acid/index.html>, y nos aparecerá lo siguiente:

**Warning:** mysql\_pconnect() [[function.mysql-pconnect](#)]: Client does not support authentication protocol requested by server; consider upgrading MySQL client in *C:\Snort\adodb\drivers\adodb-mysql.inc.php* on line 354

## Error (p)connecting to DB : snort@localhost:3306

Check the DB connection variables in *acid\_conf.php*

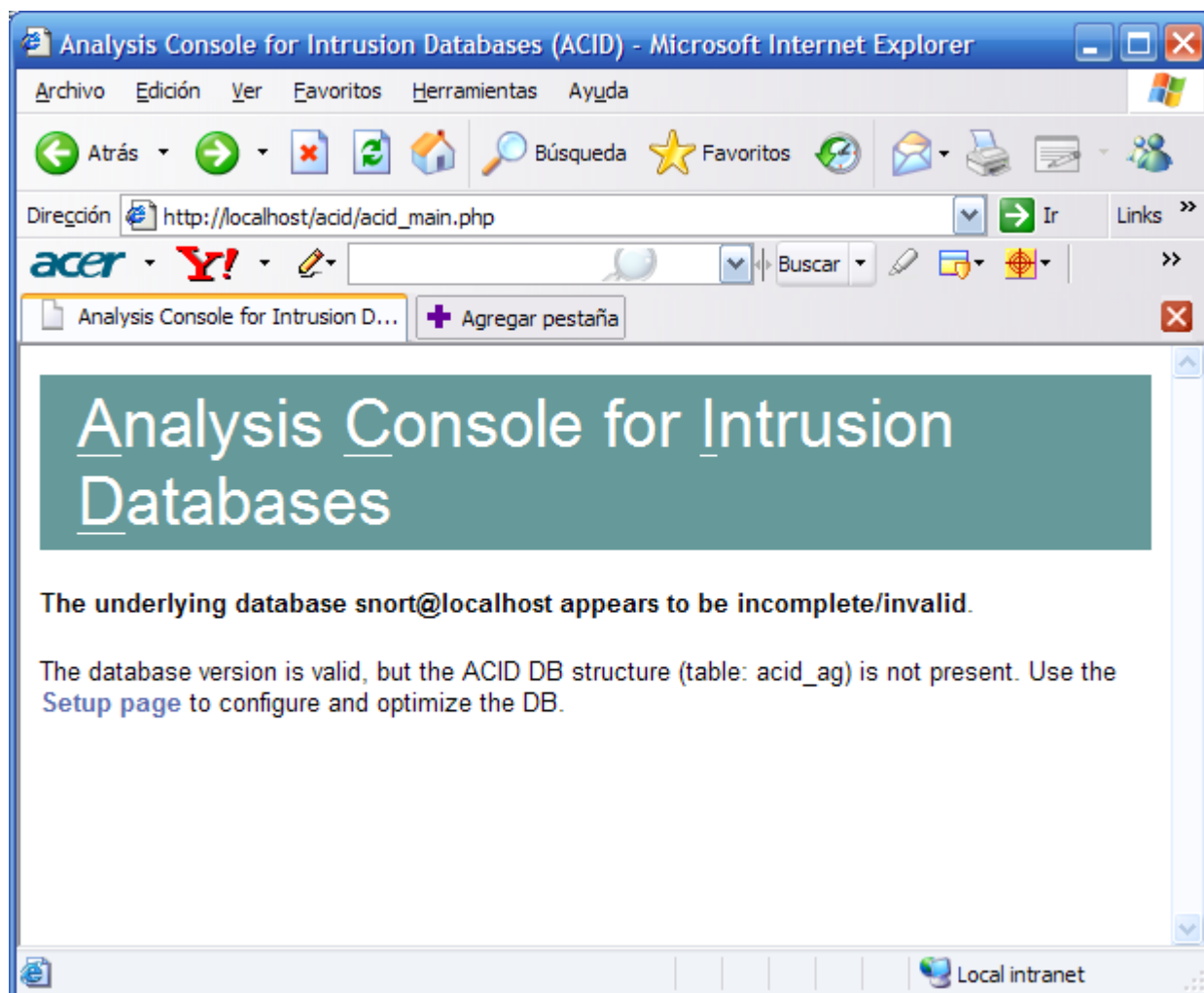
```
stored
    = $alert_dbname      : MySQL database name where the alerts are
    = $alert_host        : host where the database is stored
    = $alert_port        : port where the database is stored
    = $alert_user        : username into the database
    = $alert_password    : password for the username
```

**Database ERROR:Client does not support authentication protocol requested by server; consider upgrading MySQL client**

Este error se solventa accediendo a mysql como root y ejecutando los siguientes comandos:

```
mysql> UPDATE mysql.user SET Password = OLD_PASSWORD('newpwd')
    -> WHERE Host = 'some_host' AND User = 'some_user';
mysql> FLUSH PRIVILEGES;
```

Ahora si hacemos un refresco en el browser, aparecerá:



Hacemos click en **Setup page**:

ACID: DB Setup - Microsoft Internet Explorer

Archivo Edición Ver Favoritos Herramientas Ayuda

Atrás Búsqueda Favoritos Ir Links

Dirección http://localhost/acid/acid\_db\_setup.php

acer Y! Buscar

ACID: DB Setup + Agregar pestaña

ACID **DB Setup** Home Search | AG Maintenance [ Back ]

Operation	Description	Status
ACID tables	Adds tables to extend the Snort DB to support the ACID functionality	<input type="button" value="Create ACID AG"/>
Search Indexes	(Optional) Adds indexes to the Snort DB to optimize the speed of the queries	DONE

**[Loaded in 0 seconds]**

ACID v0.9.6b23 ( by Roman Danyliw as part of the AirCERT project )

Listo Local intranet

Click en Create ACID AG:

ACID DB Setup

Home  
Search | AG Maintenance

[ Back ]

Successfully created 'acid\_ag'

Successfully created 'acid\_ag\_alert'

Successfully created 'acid\_ip\_cache'

Successfully created 'acid\_event'

Operation	Description	Status
ACID tables	Adds tables to extend the Snort DB to support the ACID functionality	DONE
Search Indexes	(Optional) Adds indexes to the Snort DB to optimize the speed of the queries	DONE

The underlying Alert DB is configured for usage with ACID.

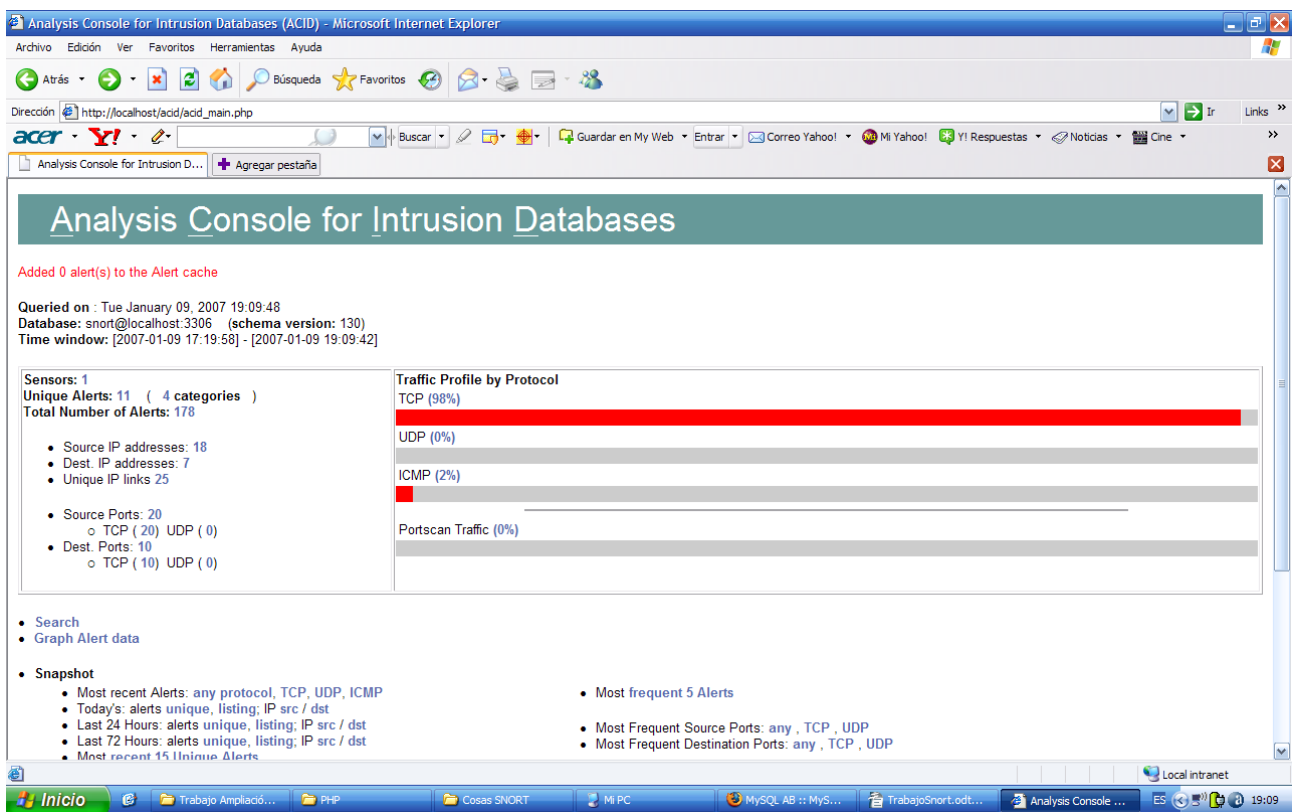
**Additional DB permissions**  
In order to support Alert purging (the selective ability to permanently delete alerts from the database) and DNS/whois lookup caching, the DB user "miky" must have the DELETE and UPDATE privilege on the database "snort@localhost"

Goto the [Main page](#) to use the application.

**[Loaded in 1 seconds]**

Local intranet

Hacemos click en Main Page y vemos como van las alertas registradas por Snort:



Ahora solamente nos queda explorar todas las ventajas que nos da ACID.

## 5. Puesta en marcha de la herramienta

Ahora lo único que tenemos que hacer es que snort sea un servicio del sistema de tal modo que no tengamos necesidad de tener que estar lanzando snort cada vez que accedamos al ordenador. Para esto, lo único que tenemos que hacer es:

1- Instalarlo como servicio:

```
snort /SERVICE /INSTALL -dev -c c:\Snort\etc\snort.conf -l c:\Snort\log -i2
```

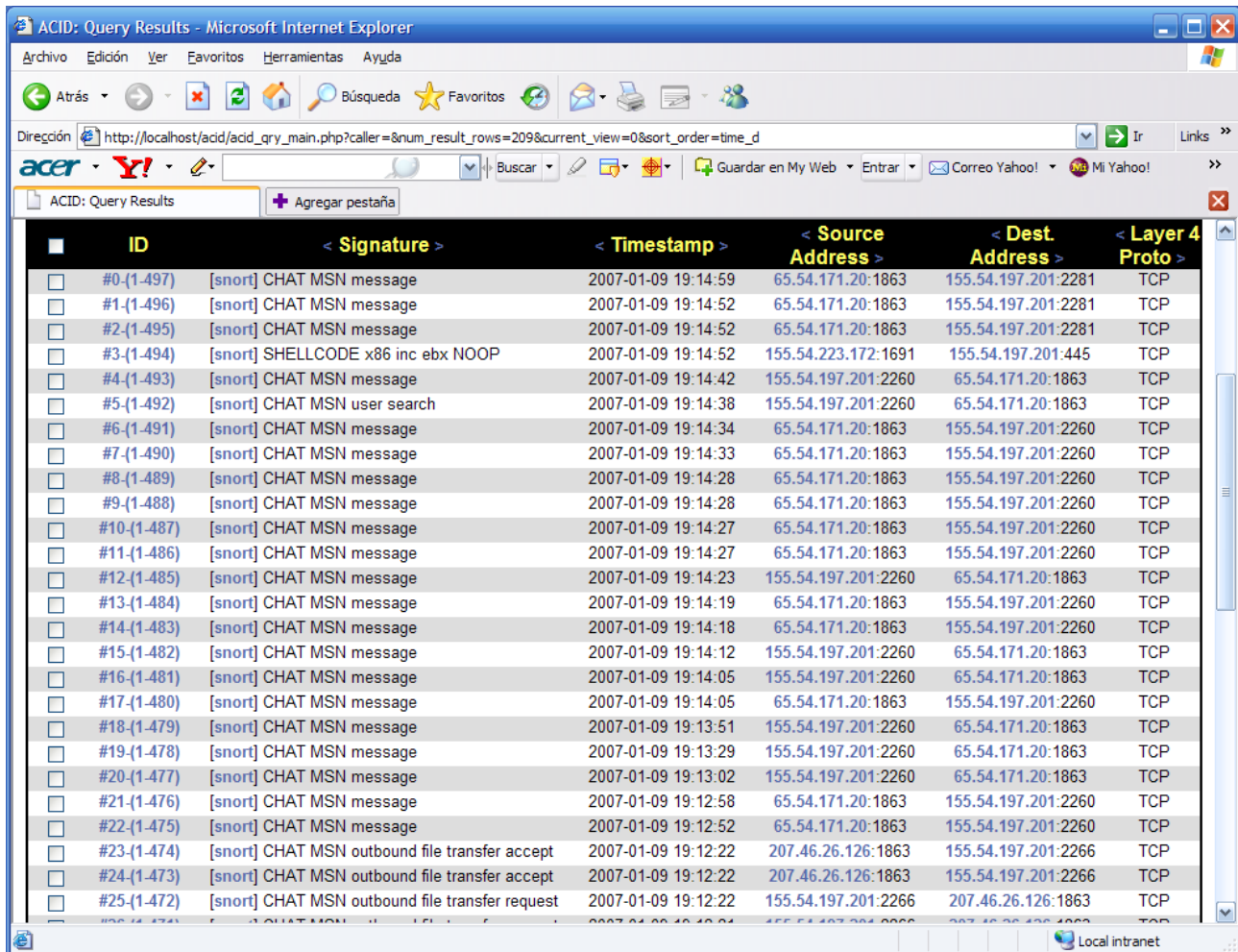
2- Lanzar el servicio:

```
net start snort
```

Con esto ya tenemos snort funcionando, interactuando con una base de datos mysql, y además tenemos la ayuda de una herramienta que nos permite visualizar los datos de una manera muy sencilla desde un navegador web.

## 6. Interpretación y manejo de los resultados, ejemplo práctico

Cuando comenzamos a trabajar con Snort podemos darnos cuenta de que muchas de las alertas que se producen son falsas alertas. Por ejemplo, mientras la realización del tutorial he estado hablando con un amigo por el aMsn y resulta que cuando he terminado de instalar ACID, me salían las siguientes alarmas:



ID	Signature	Timestamp	Source Address	Dest. Address	Layer 4 Proto
#0-(1.497)	[snort] CHAT MSN message	2007-01-09 19:14:59	65.54.171.20:1863	155.54.197.201:2281	TCP
#1-(1.496)	[snort] CHAT MSN message	2007-01-09 19:14:52	65.54.171.20:1863	155.54.197.201:2281	TCP
#2-(1.495)	[snort] CHAT MSN message	2007-01-09 19:14:52	65.54.171.20:1863	155.54.197.201:2281	TCP
#3-(1.494)	[snort] SHELLCODE x86 inc ebx NOOP	2007-01-09 19:14:52	155.54.223.172:1691	155.54.197.201:445	TCP
#4-(1.493)	[snort] CHAT MSN message	2007-01-09 19:14:42	155.54.197.201:2260	65.54.171.20:1863	TCP
#5-(1.492)	[snort] CHAT MSN user search	2007-01-09 19:14:38	155.54.197.201:2260	65.54.171.20:1863	TCP
#6-(1.491)	[snort] CHAT MSN message	2007-01-09 19:14:34	65.54.171.20:1863	155.54.197.201:2260	TCP
#7-(1.490)	[snort] CHAT MSN message	2007-01-09 19:14:33	65.54.171.20:1863	155.54.197.201:2260	TCP
#8-(1.489)	[snort] CHAT MSN message	2007-01-09 19:14:28	65.54.171.20:1863	155.54.197.201:2260	TCP
#9-(1.488)	[snort] CHAT MSN message	2007-01-09 19:14:28	65.54.171.20:1863	155.54.197.201:2260	TCP
#10-(1.487)	[snort] CHAT MSN message	2007-01-09 19:14:27	65.54.171.20:1863	155.54.197.201:2260	TCP
#11-(1.486)	[snort] CHAT MSN message	2007-01-09 19:14:27	65.54.171.20:1863	155.54.197.201:2260	TCP
#12-(1.485)	[snort] CHAT MSN message	2007-01-09 19:14:23	155.54.197.201:2260	65.54.171.20:1863	TCP
#13-(1.484)	[snort] CHAT MSN message	2007-01-09 19:14:19	65.54.171.20:1863	155.54.197.201:2260	TCP
#14-(1.483)	[snort] CHAT MSN message	2007-01-09 19:14:18	65.54.171.20:1863	155.54.197.201:2260	TCP
#15-(1.482)	[snort] CHAT MSN message	2007-01-09 19:14:12	155.54.197.201:2260	65.54.171.20:1863	TCP
#16-(1.481)	[snort] CHAT MSN message	2007-01-09 19:14:05	155.54.197.201:2260	65.54.171.20:1863	TCP
#17-(1.480)	[snort] CHAT MSN message	2007-01-09 19:14:05	65.54.171.20:1863	155.54.197.201:2260	TCP
#18-(1.479)	[snort] CHAT MSN message	2007-01-09 19:13:51	155.54.197.201:2260	65.54.171.20:1863	TCP
#19-(1.478)	[snort] CHAT MSN message	2007-01-09 19:13:29	155.54.197.201:2260	65.54.171.20:1863	TCP
#20-(1.477)	[snort] CHAT MSN message	2007-01-09 19:13:02	155.54.197.201:2260	65.54.171.20:1863	TCP
#21-(1.476)	[snort] CHAT MSN message	2007-01-09 19:12:58	65.54.171.20:1863	155.54.197.201:2260	TCP
#22-(1.475)	[snort] CHAT MSN message	2007-01-09 19:12:52	65.54.171.20:1863	155.54.197.201:2260	TCP
#23-(1.474)	[snort] CHAT MSN outbound file transfer accept	2007-01-09 19:12:22	207.46.26.126:1863	155.54.197.201:2266	TCP
#24-(1.473)	[snort] CHAT MSN outbound file transfer accept	2007-01-09 19:12:22	207.46.26.126:1863	155.54.197.201:2266	TCP
#25-(1.472)	[snort] CHAT MSN outbound file transfer request	2007-01-09 19:12:22	155.54.197.201:2266	207.46.26.126:1863	TCP

Que sucede con esto, que Snort por defecto está creando muchas falsas alarmas. Al darnos cuenta de esto, lo único que tenemos que hacer es comentar la regla que hace que salte dicha alarma. Para ello primero paramos snort:

```
net stop snort
```

Comentamos las reglas que creemos que son innecesarias, en nuestro caso comentaremos del fichero **chat.rules**:

```
#alert tcp $HOME_NET any <> $EXTERNAL_NET 1863 (msg:"CHAT MSN message";  
flow:established; content:"MSG "; depth:4; content:"Content-Type|3A|"; nocase;  
content:"text/plain"; distance:1; classtype:policy-violation; sid:540; rev:11;)
```

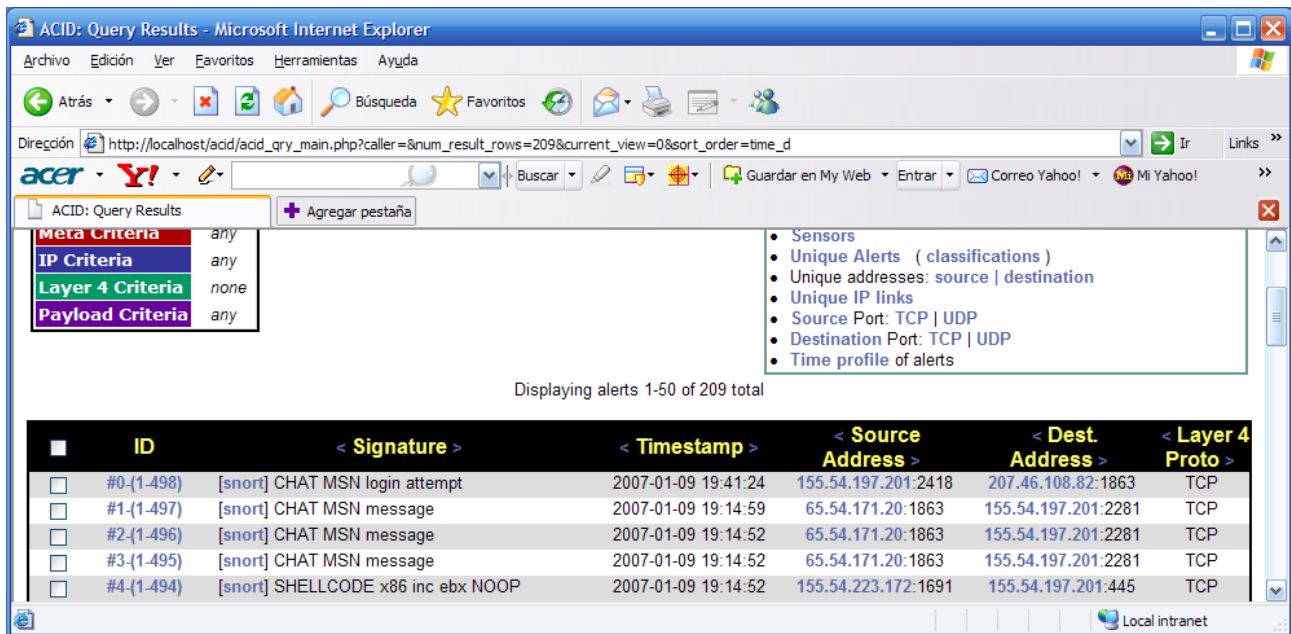
Ahora tendremos que volver a instalar el servicio para que las reglas modificadas queden en el registro de reglas como que han sido modificadas:

```
snort /SERVICE /UNINSTALL
```

```
snort /SERVICE /INSTALL -dev -c c:\Snort\etc\snort.conf -l c:\Snort\log -i2
```

```
net start snort
```

Vuelvo a iniciar sesión y mantengo otra conversación vía aMsn con un amigo, y la única alarma registrada a sido la que indica que hemos iniciado sesión : **CHAT MSN login attempt**, las demás son anteriores a esta nueva conexión:



Supongamos que como administradores queremos aprovechar la potencia de Snort, para poder apreciar si se produce algún ataque por el puerto del messenger, y que máquina(s) tenían abierta una sesión en ese momento. Para ello, tenemos que registrar el momento en el que un equipo realiza el logueo ( que ya nos lo da hecho Snort con la alerta CHAT MSN login attempt), y tenemos que registrar momento en el que se realiza el logout. Para registrar el logout tenemos que crear una regla que sea capaz de registrar el momento en el que se realiza.

Para ello tenemos que saber el contenido de un mensaje logout. Mirando en el alert.ids, encontramos las siguientes tramas:

```
=====  
01/10-12:08:01.703664 0:5:9A:3C:78:0 -> 0:6:52:51:B0:1B type:0x800 len:0x36  
155.54.197.77:3172 -> 207.46.108.82:1863 TCP TTL:128 TOS:0x0 ID:10667 IpLen:20 DgmLen:40 DF  
***A*** Seq: 0xC544D936 Ack: 0x63FF6B1A Win: 0x44E8 TcpLen: 20  
=====  
01/10-12:08:04.468543 0:5:9A:3C:78:0 -> 0:6:52:51:B0:1B type:0x800 len:0x3B  
155.54.197.77:3172 -> 207.46.108.82:1863 TCP TTL:128 TOS:0x0 ID:10695 IpLen:20 DgmLen:45 DF  
***AP*** Seq: 0xC544D936 Ack: 0x63FF6B1A Win: 0x44E8 TcpLen: 20  
4F 55 54 0D 0A OUT..
```

```

=====
01/10-12:08:04.469457 0:5:9A:3C:78:0 -> 0:6:52:51:B0:1B type:0x800 len:0x36
155.54.197.77:3172 -> 207.46.108.82:1863 TCP TTL:128 TOS:0x0 ID:10696 IpLen:20 DgmLen:40 DF
***A***F Seq: 0xC544D93B Ack: 0x63FF6B1A Win: 0x44E8 TcpLen: 20

=====
01/10-12:08:04.705276 0:6:52:51:B0:1B -> 0:5:9A:3C:78:0 type:0x800 len:0x36
207.46.108.82:1863 -> 155.54.197.77:3172 TCP TTL:110 TOS:0x0 ID:31933 IpLen:20 DgmLen:40 DF
***A***F Seq: 0x63FF6B1A Ack: 0xC544D93B Win: 0xFC EE TcpLen: 20

=====
01/10-12:08:04.705321 0:5:9A:3C:78:0 -> 0:6:52:51:B0:1B type:0x800 len:0x36
155.54.197.77:3172 -> 207.46.108.82:1863 TCP TTL:128 TOS:0x0 ID:10697 IpLen:20 DgmLen:40 DF
***A**** Seq: 0xC544D93C Ack: 0x63FF6B1B Win: 0x44E8 TcpLen: 20

=====
01/10-12:08:04.705422 0:6:52:51:B0:1B -> 0:5:9A:3C:78:0 type:0x800 len:0x36
207.46.108.82:1863 -> 155.54.197.77:3172 TCP TTL:110 TOS:0x0 ID:32010 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x63FF6B1B Ack: 0xC544D93C Win: 0xFC EE TcpLen: 20
=====

```

Como ya se comento, se necesitan los conceptos avanzados de protocolos de redes para entender como crear reglas que más o menos sean buenas, o que por lo menos registren lo que deseamos.

Como se puede observar es una desconexión TCP a tres bandas (thee hand shake) entre las direcciones 155.54.197.77 (la nuestra) y 207.46.108.82 (la del servidor de messenger).

Con lo cual, tenemos que hacer una regla que se active ante la salida de una trama TCP con contenido OUT con los bits A y P activos, y que vaya al puerto 1863, con lo que la regla sería:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 1863 (msg:"CHAT MSN logout";
flags:AP; content:"OUT"; depth:3;classtype:policy-violation; sid:10009; rev:2;)
```

Obsérvese que le asignamos la clase *policy-violation* que no hemos creado nosotros, como en el apartado **2.1. Manejo de las reglas**, la clase con la que se clasificará esta alarma. Esto lo hemos hecho para que la alerta de esta regla sea almacenada con la misma clase con la que se almacenan las demás reglas referentes a las de CHAT. *Mirar chat.rules*.

Esta regla que hemos introducido no nos identifica solamente los paquetes que vienen para cerrar una sesión de messenger, si no que también nos alertará del tráfico generado cuando cerramos una ventana. Esto nos servirá para saber cuantas conversaciones ha tenido un usuario, ya que por defecto hay una regla que registra en el momento en el que realizar una conexión con un contacto mediante la alarma **CHAT MSN user search**. Con lo tendremos:

1. Cuando inicia la sesión un usuario.

2. Número de conversaciones y tiempo de cada una
3. Final de la sesión.

Por ejemplo:

ID	Signature	Timestamp	Source Address	Dest. Address	Layer 4 Proto
#0-(1-8)	[snort] CHAT MSN logout	2007-01-10 16:51:10	155.54.197.212:1298	207.46.108.82:1863	TCP
#1-(1-7)	[snort] CHAT MSN logout	2007-01-10 16:50:57	155.54.197.212:1303	207.46.27.68:1863	TCP
#2-(1-6)	[snort] CHAT MSN user search	2007-01-10 16:50:51	155.54.197.212:1303	207.46.27.68:1863	TCP
#3-(1-5)	[snort] CHAT MSN logout	2007-01-10 16:50:47	155.54.197.212:1302	207.46.26.155:1863	TCP
#4-(1-4)	[snort] CHAT MSN outbound file transfer accept	2007-01-10 16:50:43	207.46.26.155:1863	155.54.197.212:1302	TCP
#5-(1-3)	[snort] CHAT MSN outbound file transfer request	2007-01-10 16:50:40	155.54.197.212:1302	207.46.26.155:1863	TCP
#6-(1-2)	[snort] CHAT MSN user search	2007-01-10 16:50:36	155.54.197.212:1302	207.46.26.155:1863	TCP
#7-(1-1)	[snort] CHAT MSN login attempt	2007-01-10 16:50:08	155.54.197.212:1298	207.46.108.82:1863	TCP

Se comenzó la sesión el 10-1-2007 a las 16:50:08 y se cerró el 10-1-2007 a las 16:51:10.

Los *User Search* ocurren cuando hacemos doble click sobre cualquier contacto. Los *logout* solamente se crean si se ha producido una conversación con el contacto con el que intentamos hablar, es decir, pueden haber ocasiones en las que no aparezca el *logout* debido a que el tráfico no se produce ya que no se había llegado a producir la conexión completa, el hecho de que se registren los *user search* es debido a que la regla se activa cuando buscamos al contacto, no cuando se ha producido la conexión en sí.

## 7. Ataques Remotos

Un ataque remoto es cualquier ataque que se lanza desde un ordenador a otro cualquiera, independientemente del lugar en el que se encuentren ambos. Puede ser que se encuentren en una misma oficina, o que la distancia entre ambos sea muy grande.

En este apartado vamos a describir los siguientes ataques:

- **Escaneos de puertos**
- **Spoofing**
- **Negaciones de servicio**
- **Interceptación**
- **Ataques a aplicaciones**
  - **Correo electrónico**
  - **Ataques vía web**

Para cada uno de los ataques vamos a describir sus características, modo en el que atacan a los equipos, finalidades, vulnerabilidades que aprovechan, y posibles soluciones a los diferentes ataques.

### 7.1. Escaneo de puertos

Una de las primeras actividades que un potencial (o no tan potencial) atacante realizará contra su objetivo será sin duda un escaneo de puertos, un portscan; esto le permitirá obtener en primer lugar información básica acerca de qué servicios estamos ofreciendo en nuestras máquinas y, adicionalmente, otros detalles de nuestro entorno como qué sistema operativo tenemos instalados en cada host o ciertas características de la arquitectura de nuestra red. Analizando qué puertos están abiertos en un sistema, el atacante puede buscar agujeros en cada uno de los servicios ofrecidos: cada puerto abierto en una máquina es una potencial puerta de entrada a la misma.

Comprobar el estado de un determinado puerto es a priori una tarea muy sencilla; incluso es posible llevarla a cabo desde la línea de órdenes, usando una herramienta tan genérica como **telnet**.

Por lo general, nadie en su sano juicio usaría telnet para realizar un escaneo de puertos masivo contra un sistema o contra toda una red; existen herramientas como **strobe** o **nmap** (la más conocida) que pueden realizar esta tarea de una forma más o menos cómoda y automatizable. Evidentemente, ninguno de estos programas se dedica a lanzar telnets contra los puertos de un sistema: los escaneadores de puertos actuales implementan diferentes técnicas que permiten desde detectar desde la versión del sistema operativo usado en la máquina atacada hasta pasar inadvertidos ante diferentes sistemas de detección de intrusos.

Existen diferentes aproximaciones para clasificar los escaneos de puertos, tanto en función de las técnicas seguidas en el ataque como en función de a qué sistemas o puertos concretos va dirigido. Por ejemplo, se habla de un **escaneo horizontal** cuando el atacante busca la disponibilidad de determinado servicio en diferentes máquinas de una red; por ejemplo, si el pirata dispone de un exploit que aprovecha un fallo en la implementación de sendmail, es normal que trate de averiguar qué máquinas aceptan peticiones SMTP en un determinado segmento para posteriormente atacar a dichos sistemas. Si por contra el pirata sólo escanea puertos de una máquina, se denomina al **ataque**

**escaneo vertical**, y suele denotar el interés del atacante en ese host concreto; si comprueba todos los puertos del sistema al escaneo se le denomina **vanilla**, mientras que si sólo lo hace contra determinados puertos o rangos, se le denomina **strobe** (en referencia al programa del mismo nombre). Si nos basamos en las técnicas utilizadas, podemos dividir los escaneos en tres grandes familias: **open, half-open y stealth**; vamos a hablar con más detalle de cada una de ellas y de los diferentes tipos escaneos que las forman.

Los escaneos **open** se basan en el establecimiento de una conexión TCP completa mediante el conocido como protocolo de acuerdo de tres vías o three-way handshake, por lo que son muy sencillos de detectar y detener. Utilizan la llamada connect(), el escaneador intenta establecer una conexión con un puerto concreto del host atacado, y en función de la respuesta obtenida conoce su estado: una técnica rápida, sencilla, fiable y que no necesita de ningún privilegio especial en la máquina atacante.

La segunda técnica que hemos comentado es la de los escaneos **half-open**; en este caso, el pirata finaliza la conexión antes de que se complete el protocolo de acuerdo de tres vías, lo que de entrada dificulta - aunque no mucho - la detección del ataque por parte de algunos detectores de intrusos muy simples (casi todos los actuales son capaces de detectarlos). Dentro de esta técnica se encuentra el SYN Scanning: cuando el origen - atacante - recibe del destino - máquina escaneada - los bits SYN+ACK, envía un bit RST (no es necesaria una nueva trama, ya que este bit se envía automáticamente a nivel de núcleo) en lugar del ACK correspondiente a un three-way handshake completo. Los escaneos SYN son fácilmente detectables y pueden ser bloqueados en cualquier cortafuegos; existe una variable de esta técnica denominada dumb scanning en la que entra en juego una tercera máquina denominada 'tonta' (por el poco tráfico que emite y recibe), algo que puede ayudar al pirata a camuflar su origen real. Sin embargo, el dumb scanning es más complicado que el SYN scanning, por lo que se utiliza mucho menos en la vida real.

Finalmente, existe otro modelo de escaneo denominado **stealth scanning**. Por stealth scanning se conoce a una familia de técnicas de escaneo que cumplen alguna de las siguientes condiciones:

- Eludir cortafuegos o listas de control de acceso.
- No ser registradas por sistemas de detección de intrusos, ni orientados a red ni en el propio host escaneado.
- Simular tráfico normal y real para no levantar sospechas ante un analizador de red.

Una de las técnicas que encontramos dentro de la familia de los escaneos stealth es la conocida como SYN+ACK. La idea es muy simple, y consiste en una violación del three-way handshake: el atacante, en lugar de enviar en primer lugar una trama SYN, envía SYN+ACK. Si el puerto está abierto simplemente se ignora, y si está cerrado sabe que no ha recibido previamente un paquete SYN, por lo que lo considera un error y envía una trama RST para finalizar la conexión.

Los escaneos basados en este método se usan poco en la actualidad, debido al elevado número de falsos positivos que pueden generar: sólo debemos pensar en los múltiples motivos - aparte de un puerto abierto - que pueden existir para que un sistema no responda ante una petición SYN+ACK: desde listas de control de accesos en los routers o cortafuegos hasta simples timeouts.

Otra técnica dentro de los escaneos stealth es el FIN scanning: en este caso, el atacante envía a su objetivo una trama con el bit FIN activo, ante lo que este responde con un RST si el puerto está cerrado, o simplemente no responde en caso de estar abierto; como en el caso de los escaneos SYN+ACK este método puede proporcionar muchos falsos positivos, por lo que tampoco se utiliza

mucho hoy en día.

Se propone un método de escaneo algo más complejo: el ACK. El atacante envía una trama con este bit activo, y si el puerto destino está abierto es muy posible que o bien el campo TTL del paquete de vuelta sea menor que el del resto de las tramas RST recibidas, o que el tamaño de ventana sea mayor que cero: como podemos ver, en este caso no basta con analizar el bit RST sino también la cabecera IP del paquete respuesta. Este método es difícil de registrar por parte de los detectores de intrusos, pero se basa en el código de red de BSD, por lo que es dependiente del operativo escaneado.

Para finalizar con la familia de stealth scanning vamos a hablar de dos métodos opuestos entre sí pero que se basan en una misma idea y proporcionan resultados similares: se trata de **XMAS** y **NULL**. Los primeros, también denominados escaneos 'árbol de navidad', se basan en enviar al objetivo tramas con todos los bits TCP (URG, ACK, PST, RST, SYN y FIN) activos; si el puerto está abierto el núcleo del sistema operativo eliminará la trama, ya que evidentemente la considera una violación del three-way handshake, pero si está cerrado devolverá un RST al atacante. Como antes, este método puede generar demasiados falsos positivos, y además sólo es aplicable contra máquinas Unix debido a que está basado en el código de red de BSD.

Por contra, el método opuesto al XMAS se denomina NULL scanning, y evidentemente consiste en enviar tramas con todos los bits TCP reseteados; el resultado es similar: no se devolverá ningún resultado si el puerto está abierto, y se enviará un RST si está cerrado. Aunque en principio este método sería aplicable a cualquier pila TCP/IP, la implementación - incorrecta - que de la misma hacen algunos operativos (entre ellos HP/UX o IRIX) hace que en ocasiones se envíen bits RST también desde los puertos abiertos, lo que puede proporcionar demasiados falsos positivos.

Antes de acabar el punto, vamos a hablar de otra técnica de escaneo de puertos que no se puede englobar en las tres clases de las que hemos hablado: se trata de los escaneos UDP, que al contrario de todos los comentados hasta el momento utiliza este protocolo, y no TCP, para determinar el estado de los puertos de una máquina. Al enviar un datagrama UDP a un puerto abierto este no ofrece respuesta alguna, pero si está cerrado devuelve un mensaje de error ICMP: ICMP/SMALL>\_PORT/SMALL>\_UNREACHABLE.

Evidentemente, estos ataques son muy sencillos de detectar y evitar tanto en un sistema de detección de intrusos como en los núcleos de algunos Unices, y por si esto fuera poco debemos recordar que UDP no es un protocolo orientado a conexión (como lo es TCP), por lo que la pérdida de datagramas puede dar lugar a un elevado número de falsos positivos.

## **7.2. Spoofing**

Por spoofing se conoce a la creación de tramas TCP/IP utilizando una dirección IP falseada; la idea de este ataque es muy sencilla (no así la ejecución): desde su equipo, un pirata simula la identidad de otra máquina de la red para conseguir acceso a recursos de un tercer sistema que ha establecido algún tipo de confianza basada en el nombre o la dirección IP del host suplantado. Y como los anillos de confianza basados en estas características tan fácilmente falsificables son aún demasiado abundantes (no tenemos más que pensar en los comandos r-, los accesos NFS, o la protección de servicios de red mediante TCP Wrapper), el spoofing sigue siendo en la actualidad un ataque no trivial, pero factible contra cualquier tipo de organización.

Como hemos visto, en el spoofing entran en juego tres máquinas: un atacante, un atacado, y

un sistema suplantado que tiene cierta relación con el atacado; para que el pirata pueda conseguir su objetivo necesita por un lado establecer una comunicación falseada con su objetivo, y por otro evitar que el equipo suplantado interfiera en el ataque. Probablemente esto último no le sea muy difícil de conseguir: a pesar de que existen múltiples formas de dejar fuera de juego al sistema suplantado - al menos a los ojos del atacado - que no son triviales (modificar rutas de red, ubicar un filtrado de paquetes entre ambos sistemas...), lo más fácil en la mayoría de ocasiones es simplemente lanzar una negación de servicio contra el sistema en cuestión. Aunque en el punto siguiente hablaremos con más detalle de estos ataques, no suele ser difícil 'tumbar', o al menos bloquear parcialmente, un sistema medio; si a pesar de todo el atacante no lo consigue, simplemente puede esperar a que desconecten de la red a la máquina a la que desea suplantar (por ejemplo, por cuestiones de puro mantenimiento).

El otro punto importante del ataque, la comunicación falseada entre dos equipos, no es tan inmediato como el anterior y es donde reside la principal dificultad del spoofing. En un escenario típico del ataque, un pirata envía una trama SYN a su objetivo indicando como dirección origen la de esa tercera máquina que está fuera de servicio y que mantiene algún tipo de relación de confianza con la atacada. El host objetivo responde con un SYN+ACK a la tercera máquina, que simplemente lo ignorará por estar fuera de servicio (si no lo hiciera, la conexión se resetearía y el ataque no sería posible), y el atacante enviará ahora una trama ACK a su objetivo, también con la dirección origen de la tercera máquina. Para que la conexión llegue a establecerse, esta última trama deberá enviarse con el número de secuencia adecuado; el pirata ha de predecir correctamente este número: si no lo hace, la trama será descartada), y si lo consigue la conexión se establecerá y podrá comenzar a enviar datos a su objetivo, generalmente para tratar de insertar una puerta trasera que permita una conexión normal entre las dos máquinas.

Podemos comprobar que el spoofing no es inmediato; de entrada, el atacante ha de hacerse una idea de cómo son generados e incrementados los números de secuencia TCP, y una vez que lo sepa ha de conseguir 'engañar' a su objetivo utilizando estos números para establecer la comunicación; cuanto más robusta sea esta generación por parte del objetivo, más difícil lo tendrá el pirata para realizar el ataque con éxito. Además, es necesario recordar que el spoofing es un ataque ciego: el atacante no ve en ningún momento las respuestas que emite su objetivo, ya que estas van dirigidas a la máquina que previamente ha sido deshabilitada, por lo que debe presuponer qué está sucediendo en cada momento y responder de forma adecuada en base a esas suposiciones.

Para evitar ataques de spoofing exitosos contra nuestros sistemas podemos tomar diferentes medidas preventivas; en primer lugar, parece evidente que una gran ayuda es reforzar la secuencia de predicción de números de secuencia TCP. Otra medida sencilla es eliminar las relaciones de confianza basadas en la dirección IP o el nombre de las máquinas, sustituyéndolas por relaciones basadas en claves criptográficas; el cifrado y el filtrado de las conexiones que pueden aceptar nuestras máquinas también son unas medidas de seguridad importantes de cara a evitar el spoofing.

Hasta ahora hemos hablado del ataque genérico contra un host denominado spoofing o, para ser más exactos, IP Spoofing; existen otros ataques de falseamiento relacionados en mayor o menor medida con este, entre los que destacan el DNS Spoofing, el ARP Spoofing y el Web Spoofing.

Para finalizar este punto, vamos a comentarlos brevemente e indicar algunas lecturas donde se puede ampliar información sobre los mismos:

- **DNS Spoofing**

Este ataque hace referencia al falseamiento de una dirección IP ante una consulta de resolución de nombre (esto es, resolver con una dirección falsa un cierto nombre DNS), o viceversa (resolver con un nombre falso una cierta dirección IP). Esto se puede conseguir de

diferentes formas, desde modificando las entradas del servidor encargado de resolver una cierta petición para falsear las relaciones dirección-nombre, hasta comprometiendo un servidor que infecte la caché de otro (lo que se conoce como DNS Poisoning); incluso sin acceso a un servidor DNS real, un atacante puede enviar datos falseados como respuesta a una petición de su víctima sin más que averiguar los números de secuencia correctos.

- **ARP Spoofing**

El ataque denominado ARP Spoofing hace referencia a la construcción de tramas de solicitud y respuesta ARP falseadas, de forma que en una red local se puede forzar a una determinada máquina a que envíe los paquetes a un host atacante en lugar de hacerlo a su destino legítimo. La idea es sencilla, y los efectos del ataque pueden ser muy negativos: desde negaciones de servicio hasta interceptación de datos, incluyendo algunos Man in the Middle contra ciertos protocolos cifrados.

- **Web Spoofing**

Este ataque permite a un pirata visualizar y modificar cualquier página web que su víctima solicite a través de un navegador, incluyendo las conexiones seguras vía SSL. Para ello, mediante código malicioso un atacante crea una ventana del navegador correspondiente, de apariencia inofensiva, en la máquina de su víctima; a partir de ahí, enruta todas las páginas dirigidas al equipo atacado - incluyendo las cargadas en nuevas ventanas del navegador - a través de su propia máquina, donde son modificadas para que cualquier evento generado por el cliente sea registrado (esto implica registrar cualquier dato introducido en un formulario, cualquier click en un enlace, etc.).

### **7.3. Negociación de servicios**

Las negaciones de servicio (conocidas como DoS, Denial of Service) son ataques dirigidos contra un recurso informático (generalmente una máquina o una red, pero también podría tratarse de una simple impresora o una terminal) con el objetivo de degradar total o parcialmente los servicios prestados por ese recurso a sus usuarios legítimos; constituyen en muchos casos uno de los ataques más sencillos y contundentes contra todo tipo de servicios, y en entornos donde la disponibilidad es valorada por encima de otros parámetros de la seguridad global puede convertirse en un serio problema, ya que un pirata puede interrumpir constantemente un servicio sin necesidad de grandes conocimientos o recursos, utilizando simplemente sencillos programas y un módem y un PC caseros.

Las negaciones de servicio más habituales suelen consistir en la inhabilitación total de un determinado servicio o de un sistema completo, bien porque ha sido realmente bloqueado por el atacante o bien porque está tan degradado que es incapaz de ofrecer un servicio a sus usuarios. En la mayor parte de sistemas, un usuario con acceso shell no tendría muchas dificultades en causar una negación de servicio que tirara abajo la máquina o la ralentizara enormemente; esto no tiene porqué ser - y de hecho en muchos casos no lo es - un ataque intencionado, sino que puede deberse a un simple error de programación. El problema real no es que usuarios legítimos de un entorno causen, intencionada o inintencionadamente, negaciones de servicio: el mayor problema es que esas negaciones sean causadas de forma remota por piratas ajenos por completo a nuestra organización, capaces de tumbar un servidor de miles de euros con sencillos programas, sin dejar ningún rastro y lo peor, sin ser muchas veces conscientes del daño que están haciendo.

Estos ataques remotos de negación de servicio son considerados negativos incluso por muchos de los propios piratas - especialmente los más experimentados -, ya que realmente no suelen demostrar nada positivo de quien los lanza (si es que algún ataque demuestra algo positivo

de alguien, lo cual sería muy discutible...): generalmente se trata de un script-kiddie ejecutando un programa que ni ha hecho, ni entiende, ni será capaz de entender. En la mayor parte de los casos la negación de servicio tiene éxito porque el objetivo utiliza versiones no actualizadas de demonios (si se para un servicio concreto) o del propio núcleo del sistema operativo, si se detiene o degrada por completo la máquina. Para evitar esto, la norma a seguir es evidente: mantener siempre actualizados nuestros sistemas, tanto en lo referente al nivel de parcheado o versiones del núcleo, como en lo referente a programas críticos encargados de ofrecer un determinado servicio (demonios, por norma general: sendmail, httpd, pop3d...).

De un tiempo a esta parte - en concreto, desde 1999 - se ha popularizado mucho el término “negación de servicio distribuida” (Distributed Denial of Service, DDoS): en este ataque un pirata compromete en primer lugar un determinado número de máquinas y, en un determinado momento, hace que todas ellas ataquen masiva y simultáneamente al objetivo u objetivos reales enviándoles diferentes tipos de paquetes; por muy grandes que sean los recursos de la víctima, el gran número de tramas que reciben hará que tarde o temprano dichos recursos sean incapaces de ofrecer un servicio, con lo que el ataque habrá sido exitoso. Si en lugar de cientos o miles de equipos atacando a la vez lo hiciera uno sólo las posibilidades de éxito serían casi inexistentes, pero es justamente el elevado número de “pequeños” atacantes lo que hace muy difícil evitar este tipo de negaciones de servicio.

Los ataques de negación de servicio distribuidos más habituales consisten en el envío de un gran número de paquetes a un determinado objetivo por parte de múltiples hosts, lo que se conoce como packet flooding (en función del tipo de paquetes utilizados se habla de ping flood, de SYN flood...). Defenderse de este tipo de ataques es difícil: en primer lugar, uno piensa en bloquear de alguna forma (probablemente en un cortafuegos o en un router) todo el tráfico proveniente de los atacantes; pero ¿qué sucede cuando tenemos miles de ordenadores atacando desde un gran número de redes diferentes? ¿Los bloqueamos uno a uno? Esto supondría un gran esfuerzo que difícilmente ayudaría, ya que lo más probable es que en el tiempo que nos cueste bloquear el tráfico de una determinada máquina, dos o tres nuevas nos comiencen a atacar. Entonces, ¿bloqueamos todo el tráfico dirigido hacia el objetivo? Si hacemos esto, estamos justamente ayudando al atacante, ya que somos nosotros mismos los que causamos una negación en el servicio a los usuarios legítimos de nuestro sistema...

Como vemos, la defensa ante una negación de servicio distribuida no es inmediata; en cualquier caso, podemos tomar ciertas medidas preventivas que nos ayudarán a limitar el alcance de uno de estos ataques (y en general de las negaciones de servicio remotas, distribuidas o no). De entrada, un correcto filtrado del tráfico dirigido a nuestras máquinas es vital para garantizar nuestra seguridad: no hay que responder a pings externos a nuestra red, es necesario activar el antispoofing en nuestros cortafuegos, etc. Establecer correctamente límites a la utilización de nuestros recursos, como ya hemos visto, es también una importante medida preventiva; es posible limitar el ancho de banda dedicado a una determinada aplicación o a un protocolo, de forma que las utilidades por encima del margen son negadas. También podemos limitar los recursos del sistema (CPU, memoria, disco...) que puede consumir en global una determinada aplicación servidora (por ejemplo, un demonio sirviendo páginas web), además de restringir sus recursos por cliente simultáneo (en base, por ejemplo, a la dirección origen de ese cliente).

A pesar de las dificultades con las que nos podemos encontrar a la hora de prevenir ataques de negación de servicio, una serie de medidas sencillas pueden ayudarnos de forma relativa en esa tarea; las negaciones de servicio son por desgracia cada día más frecuentes, y ninguna organización está a salvo de las mismas. Especialmente en los ataques distribuidos, la seguridad de cualquier

usuario conectado a Internet (aunque sea con un sencillo PC y un módem) es un eslabón importante en la seguridad global de la red, ya que esos usuarios se convierten muchas veces sin saberlo en satélites que colaboran en un ataque masivo contra organizaciones de cualquier tipo.

## 7.4. Interceptación

En este apartado nos vamos a centrar en la interceptación lógica. Y sin duda, la interceptación lógica de datos más conocida y extendida es el sniffing.

En las redes de difusión, cuando una máquina envía una trama a otra indica en un campo reservado la dirección del host destino; todas las máquinas del dominio de colisión ven esa trama, pero sólo su receptora legítima la captura y elimina de la red. Este es el funcionamiento normal de TCP/IP; sin embargo, es necesario insistir en un aspecto: todas las máquinas ven la trama, y si no leen todos sus campos es porque no “quieren”. Existe un modo de funcionamiento de las interfaces de red denominado modo promiscuo, en el cual la tarjeta lee todas las tramas que circulan por la red, tanto dirigidas a ella como a otras máquinas; el leerlas no implica el eliminarlas de la red, por lo que el host destino legítimo la recibirá y eliminará sin notar nada extraño.

Para hacer funcionar un interfaz de red en modo promiscuo es necesario tener un control total del sistema o, dicho de otra forma, ser root en la máquina; esto ayuda un poco en la defensa, pero ni de lejos soluciona el problema que estamos planteando: no podemos permitir que cualquiera que sea superusuario de un sistema pueda capturar todo el tráfico que pasa por el mismo (incluyendo claves, correo electrónico, y cientos de datos privados). Por si esto fuera poco, en los sistemas donde todos los usuarios tienen un control total de la máquina (por ejemplo, en toda la familia Windows 9x) ni siquiera hace falta ese privilegio: cualquiera que se sienta en un PC puede ejecutar un sniffer y capturar todo el tráfico de la red.

Programas para ‘esnifar’ tráfico hay para todos los gustos y colores: desde dsniff y su familia, capaces hasta de capturar los correos electrónicos directamente en formato SMTP o cargar de forma automática en un navegador las mismas páginas que visita la víctima del ataque, hasta el arcaico snoop de Solaris, que vuelca paquetes en un formato por defecto casi ilegible, pasando por los clásicos tcpdump o sniffit (que en algunas de sus versiones incluía el Touch of Dead, capaz de cortar conexiones establecidas entre dos máquinas sin más que pulsar F5). Para evitar que programas de este tipo capturen nuestra información existen diferentes aproximaciones más o menos efectivas, como sustituir los HUBs de nuestra red por switches que aíslan dominios de colisión (¡jojo, esto dificulta el ataque pero no lo imposibilita!) o implantar redes privadas virtuales.

Pero sin ninguna duda la más barata y sencilla es el uso de protocolos cifrados siempre que nos sea posible (que lo suele ser casi siempre); sustituir telnet y rlogin por SSH y FTP por scp o sftp es muy sencillo, y nos proporciona un incremento de seguridad abismal en nuestro entorno.

Implantar SSL o túneles seguros quizás es algo más costoso - en tiempo solamente -, pero también en la mayoría de ocasiones es algo que vale la pena hacer: en todo momento hemos de tener presente que el sniffing es un peligro real, que no necesita de grandes medios y, lo que es peor, indetectable en la mayor parte de casos; a pesar de que existen métodos para tratar de detectar sistemas con un interfaz en modo promiscuo, no suelen ser todo lo efectivos que uno podría esperar, ya que detectar una máquina en este estado no es ni de lejos inmediato.

Para finalizar este punto podemos reflexionar brevemente sobre la peligrosidad de los ataques de interceptación; muchos de ellos necesitan privilegios de superusuario en al menos una

máquina, pero por lo demás son realmente sencillos. Sencillos y peligrosos: aunque se trate de ataques pasivos, y aunque alguien pueda pensar que si el pirata ya es root no se puede atacar más al sistema, permiten capturar datos relativos no sólo al sistema comprometido, sino a otras máquinas que quizás aún no han sido atacadas y que posiblemente representan el objetivo real del pirata.

Evitar estos ataques pasa en primera instancia por no permitir que un pirata consiga privilegios en un sistema - mejor si no consigue nada, pero esto no siempre es posible -, y en segunda por lo que ya sabemos: cifrar cuanto más tráfico mejor.

## **7.5. Ataques de aplicaciones**

### **7.5.1. Correo Electrónico**

Desde hace muchos años los sistemas de correo electrónico de una organización han sido para los piratas una fuente inagotable de puntos de entrada a la misma; lo más probable es que si le preguntamos a cualquier administrador con algo de experiencia cuál ha sido el software que más problemas de seguridad le ha causado nos responda sin dudar: sendmail, por supuesto. Y ya no sólo sendmail y el protocolo SMTP, sino que también, con la popularización de POP3, los servidores de este protocolo son un peligro potencial a tener en cuenta en cualquier entorno informático donde se utilice el correo electrónico: es decir, en todos.

De entrada, un programa como sendmail - lo ponemos como ejemplo por ser el más popular, pero podríamos hablar en los mismos términos de casi cualquier servidor SMTP - proporciona demasiada información a un atacante que simplemente conecte a él:

Y no sólo se proporcionan datos útiles para un pirata como la versión del programa utilizada o la fecha del sistema, sino que se llega incluso más lejos: tal y como se instalan por defecto, muchos servidores SMTP (aparte de sendmail, algunos tan populares como Netscape Messaging Server) informan incluso de la existencia o inexistencia de nombres de usuario y de datos sobre los mismos.

Independientemente del programa que utilicemos como servidor de correo y su versión concreta, con vulnerabilidades conocidas o no, otro gran problema de los sistemas de correo SMTP es el relay: la posibilidad de que un atacante interno utilice nuestros servidores para enviar correo electrónico a terceros, no relacionados con nuestra organización. Aunque en principio esto a muchos les pueda parecer un mal menor, no lo es; de entrada, si nuestros servidores permiten el relay estamos favoreciendo el SPAM en la red, el envío de e-mail no deseado con fines casi siempre publicitarios, algo que evidentemente a nadie le hace gracia recibir. Además, el relay causa una negación de servicio contra nuestros usuarios legítimos, tanto desde un punto de vista estrictamente teórico - alguien consume nuestros recursos de forma no autorizada, degradando así el servicio ofrecido a nuestros usuarios legítimos - como en la práctica: cuando un robot encuentra un servidor SMTP en el que se permite el relay lo utiliza masivamente mientras puede, cargando enormemente la máquina y entorpeciendo el funcionamiento normal de nuestros sistemas. Por si esto fuera poco, si se incluye a nuestra organización en alguna 'lista negra' de servidores que facilitan el SPAM se causa un importante daño a nuestra imagen, e incluso ciertos dominios pueden llegar a negar todo el correo proveniente de nuestros servidores.

Sólo existe una manera de evitar el relay: configurando correctamente todos nuestros servidores de correo. Por supuesto, esto es completamente dependiente de los programas (sendmail, iPlanet...) utilizados en nuestro entorno, por lo que es necesario consultar en la documentación correspondiente la forma de habilitar filtros que eviten el relay; por Internet existen incluso filtros

genéricos para los servidores más extendidos, por lo que nuestro trabajo no será excesivo ni complicado.

Es muy importante para nosotros cuidar cualquier aspecto de la seguridad relativo a nuestros sistemas de correo, ya que en la actualidad el correo electrónico constituye uno de los servicios básicos de cualquier empresa; simplemente hemos de contar el número de e-mails que solemos recibir al día para hacernos una idea del desastre que supondría un fallo en los sistemas encargados de procesarlo.

### **7.5.2. Ataques vía web**

Durante los últimos años los servidores web se han convertido en una excelente fuente de diversión para piratas: cualquier empresa que se precie, desde las más pequeñas a las grandes multinacionales, tiene una página web en las que al menos trata de vender su imagen corporativa. Si hace unos años un pirata que quisiera atacar a una empresa (y no a todas, ya que muy pocas tenían representación en la red) tenía que agenciárselas para obtener primero información de la misma y después buscar errores de configuración más o menos comunes de sus sistemas (o esperar al próximo bug de sendmail), hoy en día le basta con teclear el nombre de su objetivo en un navegador y añadir la coletilla '.com' detrás del mismo para contactar con al menos una de sus máquinas: su servidor web.

La mayor parte de estos ataques tiene éxito gracias a una configuración incorrecta del servidor o a errores de diseño del mismo: si se trata de grandes empresas, los servidores web suelen ser bastante complejos (alta disponibilidad, balanceo de carga, sistemas propietarios de actualización de contenidos...) y difíciles de administrar correctamente, mientras que si la empresa es pequeña es muy posible que haya elegido un servidor web simple en su instalación y administración pero en el cual es casi imposible garantizar una mínima seguridad: sí, hablamos de Microsoft Internet Information Server, un sistema que reconocidos expertos en seguridad han recomendado públicamente no utilizar en entornos serios. Sea por el motivo que sea, la cuestión es que cada día es más sencillo para un pirata ejecutar órdenes de forma remota en una máquina, o al menos modificar contenidos de forma no autorizada, gracias a los servidores web que un sistema pueda albergar.

Cualquier analizador de vulnerabilidades que podamos ejecutar contra nuestros sistemas (NESSUS, ISS Security Scanner, NAI CyberCop Scanner...) es capaz de revelar información que nos va a resultar útil a la hora de reforzar la seguridad de nuestros servidores web; incluso existen analizadores que están diseñados para auditar únicamente este servicio, como whisker.

¿Cómo evitar estos problemas de seguridad de los que estamos hablando? Una medida elemental es eliminar del servidor cualquier directorio o CGI de ejemplo que se instale por defecto; aunque generalmente los directorios (documentación, ejemplos...) no son especialmente críticos, el caso de los CGIs es bastante alarmante: muchos servidores incorporan programas que no son ni siquiera necesarios para el correcto funcionamiento del software, y que en ciertos casos - demasiados - abren enormes agujeros de seguridad, como el acceso al código fuente de algunos archivos, la lectura de ficheros fuera del DocumentRoot, o incluso la ejecución remota de comandos bajo la identidad del usuario con que se ejecuta el demonio servidor.

Otra medida de seguridad básica es deshabilitar el Directory Indexing que por defecto muchos servidores incorporan: la capacidad de obtener el listado de un directorio cuando no existe un fichero index.html o similar en el mismo; se trata de una medida extremadamente útil y sobre todo sencilla de implantar, ya que en muchos casos un simple 'chmod -r' sobre el directorio en

cuestión es suficiente para evitar este problema. A primera vista esta medida de protección nos puede resultar curiosa: a fin de cuentas, a priori todo lo que haya bajo el Document Root del servidor ha de ser público, ya que para eso se ubica ahí. Evidentemente la teoría es una cosa y la práctica otra muy diferente: entre los ficheros de cualquier servidor no es extraño encontrar desde archivos de log - por ejemplo, del cliente FTP que los usuarios suelen usar para actualizar remotamente sus páginas, como WS/SMALL>\_FTP.LOG - hasta paquetes TAR con el contenido de subdirectorios completos. Por supuesto, la mejor defensa contra estos ataques es evitar de alguna forma la presencia de estos archivos bajo el Document Root, pero en cualquier caso esto no es siempre posible, y si un atacante sabe de su existencia puede descargarlos, obteniendo en muchos casos información realmente útil para atacar al servidor (como el código de ficheros JSP, PHP, ASP...o simplemente rutas absolutas en la máquina), y una excelente forma de saber que uno de estos ficheros está ahí es justamente el Directory Indexing; por si esto no queda del todo claro, no tenemos más que ir a un buscador cualquiera y buscar la cadena `Index of /admin`, por poner un ejemplo sencillo, para hacernos una idea de la peligrosidad de este error de configuración.

Además, en cualquier servidor web es muy importante el usuario bajo cuya identidad se ejecuta el demonio httpd: ese usuario no debe ser nunca el root del sistema (evidente), pero tampoco un usuario genérico como nobody; se ha de tratar siempre de un usuario dedicado y sin acceso real al sistema. Por supuesto, las páginas HTML (los ficheros planos, para entendernos) nunca deben ser de su propiedad, y mucho menos ese usuario ha de tener permiso de escritura sobre los mismos: con un acceso de lectura (y ejecución, en caso de CGIs) es más que suficiente en la mayoría de los casos. Hemos de tener en cuenta que si el usuario que ejecuta el servidor puede escribir en las páginas web, y un pirata consigue - a través de cualquier tipo de error (configuración, diseño del demonio...) - ejecutar órdenes bajo la identidad de dicho usuario, podrá modificar las páginas web sin ningún problema (que no olvidemos, es lo que perseguirá la mayoría de atacantes de nuestro servidor web).

Igual de importante que evitar estos problemas es detectar cuando alguien trata de aprovecharlos intentando romper la seguridad de nuestros servidores; para conseguirlo no tenemos más que aplicar las técnicas de detección de intrusos que veremos en el capítulo siguiente. Una característica importante de los patrones de detección de ataques vía web es que no suelen generar muchos falsos positivos, por lo que la configuración de la base de datos inicial es rápida y sencilla, al menos en comparación con la detección de escaneos de puertos o la de tramas con alguna característica especial en su cabecera.

## **7.6 Ejemplo de ataque remotos (Nessus)**

Una vez explicado como manejar las reglas de Snort, como interpretarlas, y la necesidad de que el administrador trabaje para que sus alertas sean las que necesite, vamos a pasar a realizar un ataque remoto real mediante la herramienta Nessus.

El sitio oficial de Nessus es [www.nessus.org](http://www.nessus.org). **Nessus** es un potente escáner de redes de Software Libre. Consta de dos partes (cliente/servidor) que pueden estar instaladas en la misma máquina por simplicidad. Se debe comentar que si el ataque se hace hacia localhost lo que se consigue es auditar nuestra propia máquina. Cuando Nessus finaliza el escaneo genera unos informes muy útiles si se sabe aprovechar e interpretar la información obtenida.

Consiste en **nessusd**, el daemon Nessus, que realiza el escaneo en el sistema objetivo, y

nessus, el cliente (basado en consola o gráfico) que muestra el avance y reporte de los escaneos. Desde consola nessus puede ser programado para hacer escaneos programados con cron.

En operación normal, nessus **comienza escaneando los puertos con nmap** o con su propio escaneador de puertos para buscar puertos abiertos y **después** intentar varios **exploits**, es el nombre con el que se identifica un programa informático malicioso, o parte del programa, que trata de forzar alguna deficiencia o vulnerabilidad de otro programa (llamadas **bugs**). El fin puede ser la destrucción o inhabilitación del sistema atacado, aunque normalmente se trata de violar las medidas de seguridad para poder acceder al mismo de forma no autorizada y emplearlo en beneficio propio o como origen de otros ataques a terceros, para atacarlo. Las pruebas de vulnerabilidad, disponibles como una larga lista de plugins, son escritos en NASL (Nessus Attack Scripting Language, Lenguaje de Scripting de Ataque Nessus por sus siglas en inglés), un lenguaje scripting optimizado para interacciones personalizadas en redes.

Opcionalmente, los resultados del escaneo pueden ser exportados en reportes en varios formatos, como texto plano, XML, HTML, y LaTeX. Los resultados también pueden ser guardados en una base de conocimiento para referencia en futuros escaneos de vulnerabilidades.

Algunas de las pruebas de vulnerabilidades de Nessus pueden causar que los servicios o sistemas operativos se corrompan y caigan. El usuario puede evitar esto desactivando "unsafe test" (pruebas no seguras) antes de escanear

Descargamos Nessus y lo instalamos en una máquina diferente a la que tenemos Snort. Nosotros instalamos Nessus en 155.54.197.186 y atacamos al equipo que tiene Snort, cuya IP es 155.54.197.77. Snort creará las alertas pertinentes, las capturas que realiza son las siguientes:

ID	Signature	Timestamp	Source Address	Dest. Address	Layer 4 Proto
#0-(1-2542)	[snort] (portscan) Open Port	2007-01-10 10:29:24	155.54.197.186	155.54.197.77	Raw IP
#1-(1-2543)	[snort] (portscan) Open Port	2007-01-10 10:29:24	155.54.197.186	155.54.197.77	Raw IP
#2-(1-2544)	[snort] (portscan) Open Port	2007-01-10 10:29:24	155.54.197.186	155.54.197.77	Raw IP
#3-(1-2545)	[snort] (portscan) Open Port	2007-01-10 10:29:24	155.54.197.186	155.54.197.77	Raw IP
#4-(1-2546)	[snort] (portscan) Open Port	2007-01-10 10:29:24	155.54.197.186	155.54.197.77	Raw IP
#5-(1-2547)	[snort] (portscan) Open Port	2007-01-10 10:29:24	155.54.197.186	155.54.197.77	Raw IP
#6-(1-2533)	[snort] WEB-MISC /etc/passwd	2007-01-10 10:29:23	155.54.197.186:1979	155.54.197.77:80	TCP
#7-(1-2534)	[snort] (http_inspect) WEBROOT DIRECTORY TRAVERSAL	2007-01-10 10:29:23	155.54.197.186:1979	155.54.197.77:80	TCP
#8-(1-2535)	nessus[snort] WEB-MISC nessus 2.x 404 probe	2007-01-10 10:29:23	155.54.197.186:1981	155.54.197.77:80	TCP
#9-(1-2536)	[snort] WEB-MISC /etc/passwd	2007-01-10 10:29:23	155.54.197.186:1983	155.54.197.77:80	TCP
#10-(1-2537)	[arachNIDS][snort] WEB-MISC http directory traversal	2007-01-10 10:29:23	155.54.197.186:1983	155.54.197.77:80	TCP
#11-(1-2538)	[snort] (http_inspect) WEBROOT DIRECTORY TRAVERSAL	2007-01-10 10:29:23	155.54.197.186:1983	155.54.197.77:80	TCP
#12-(1-2539)	nessus[cve][icat][bugtraq][snort] WEB-IIS .httr access	2007-01-10 10:29:23	155.54.197.186:1984	155.54.197.77:80	TCP
#13-(1-2540)	[cve][icat][bugtraq][snort] WEB-MISC counter.exe access	2007-01-10 10:29:23	155.54.197.186:1990	155.54.197.77:80	TCP
#14-(1-2541)	[cve][icat][bugtraq][snort] WEB-MISC counter.exe access	2007-01-10 10:29:23	155.54.197.186:1993	155.54.197.77:80	TCP
#15-(1-2514)	[cve][icat][bugtraq][snort] WEB-MISC ion-p access	2007-01-10 10:29:22	155.54.197.186:1958	155.54.197.77:80	TCP

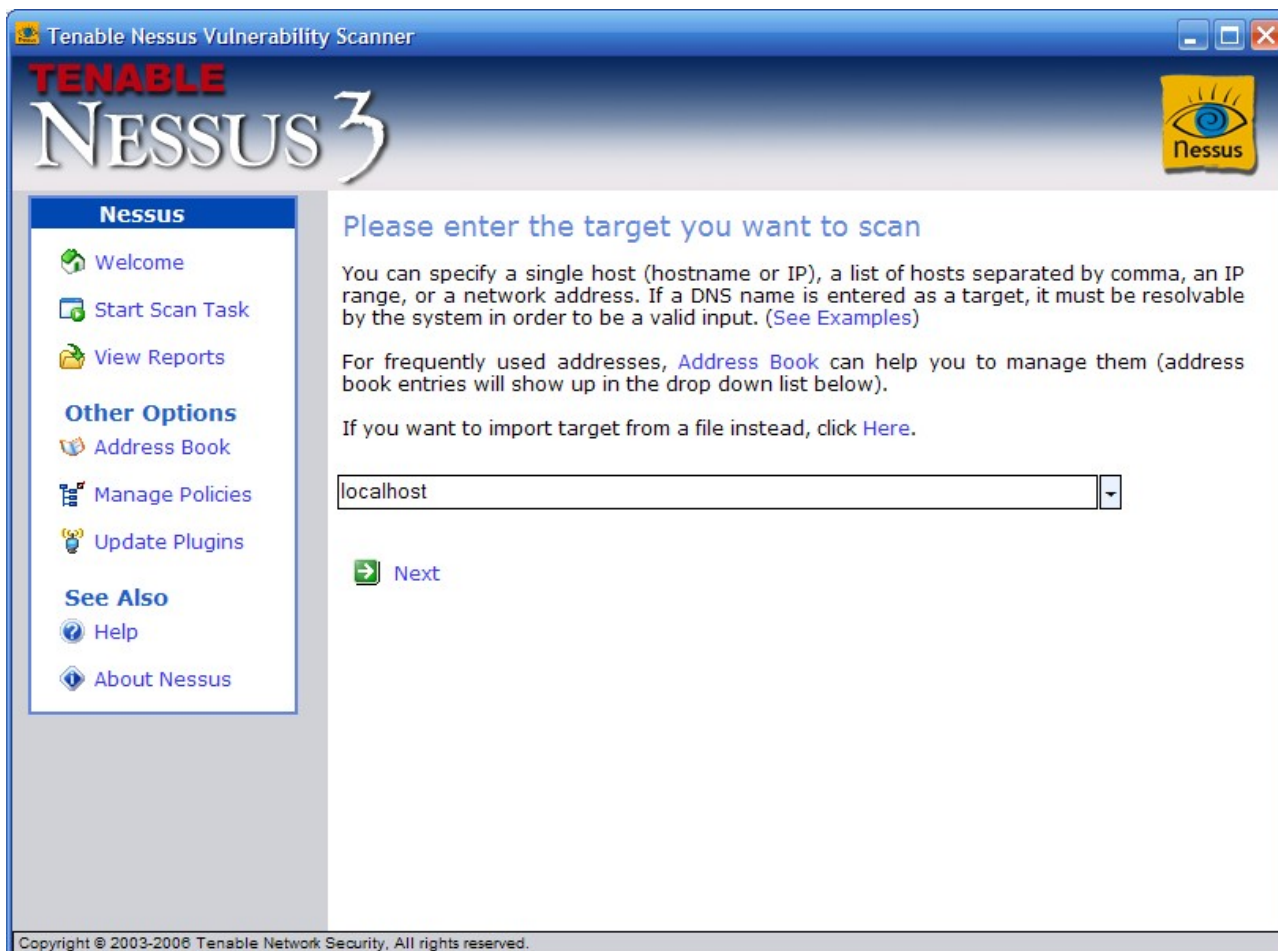
Se crean 1999 alertas creadas los el tráfico que Snort detecta como malicioso, lo que deja totalmente plasmado todos los accesos que realiza el equipo atacante hacia el atacado.

Este ejemplo nos muestra como Snort nos permitirá realizar posteriormente un análisis forense de los datos que hemos registrado sobre el escaneo de puertos, y los exploit realizados posteriormente.

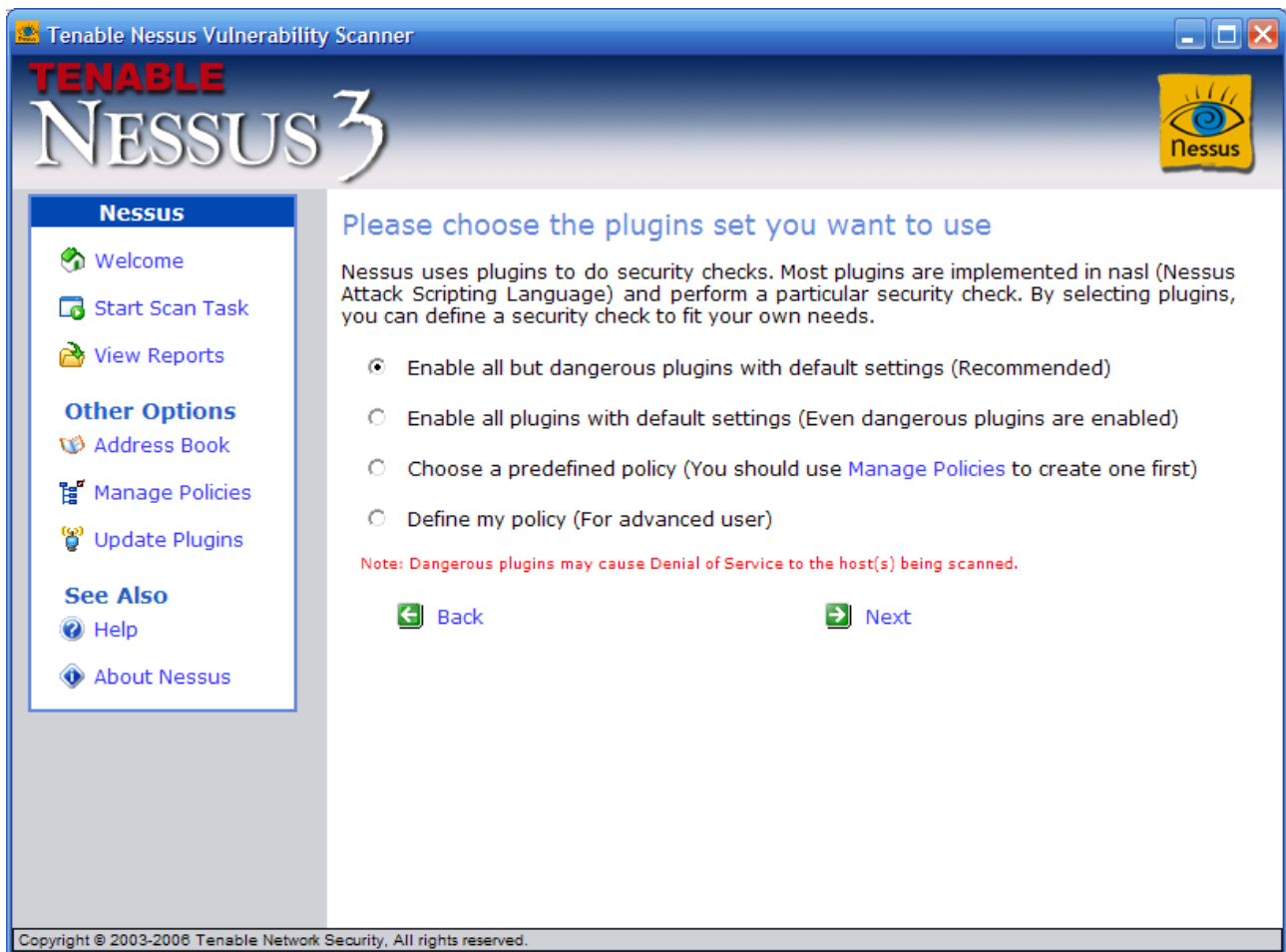
No obstante, una de las grandes capacidades que nos produce el uso de Nessus será el echo de que si lo realizamos contra nuestra red o equipo, nos dará un informe detallado de todo lo que ha realizado.

Pongamos un ejemplo:

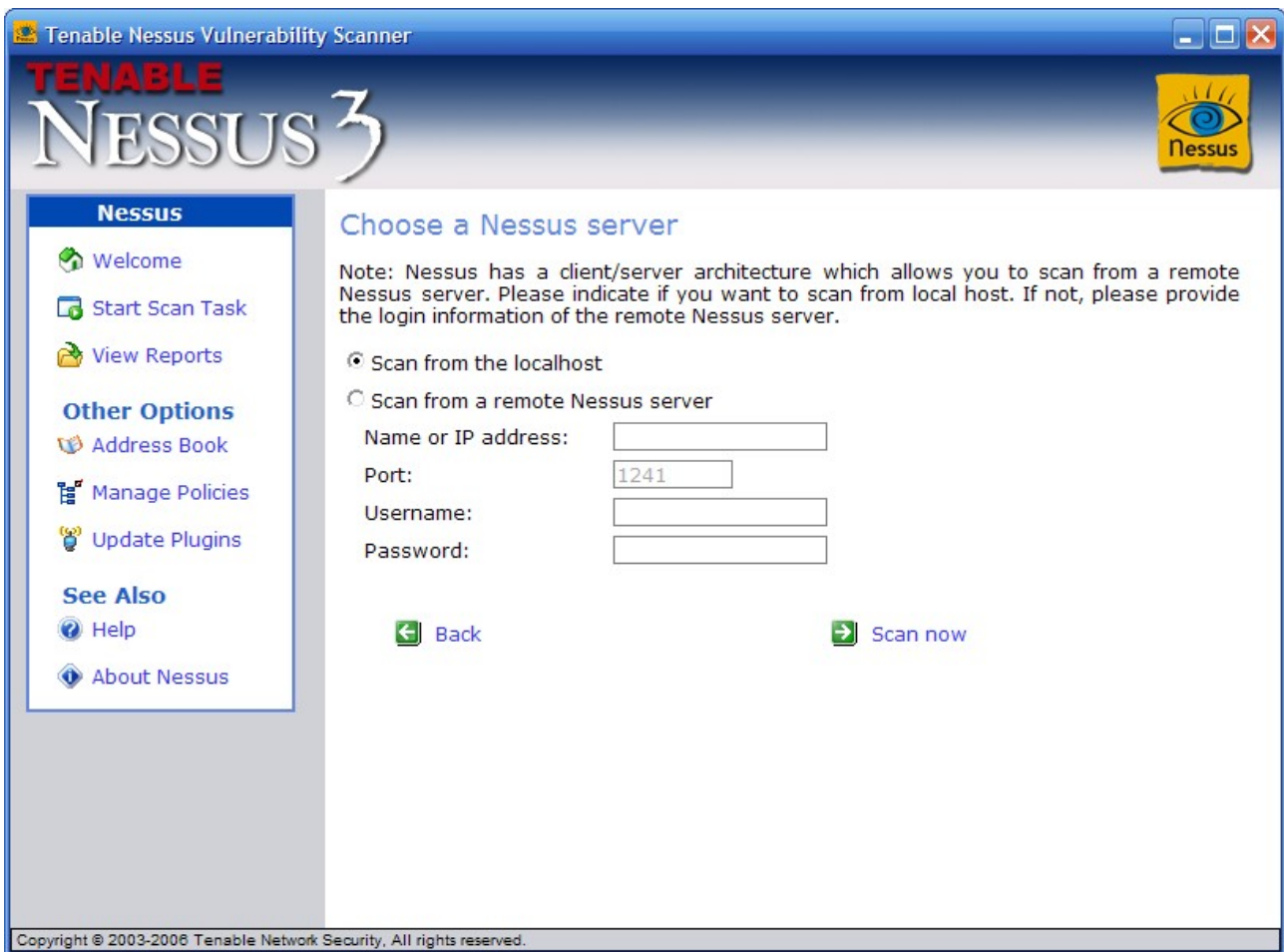
1- Decimos que queremos realizar el scan sobre nuestro equipo.



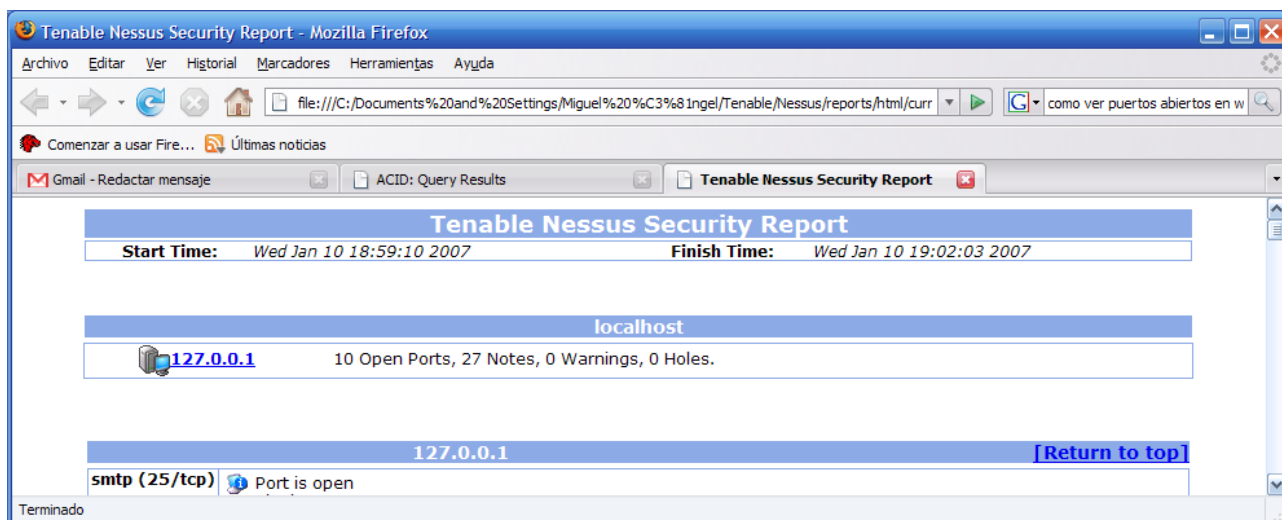
2- Elejimos la configuración por defecto:



3- Le decimos que escanee desde nuestro propio equipo:

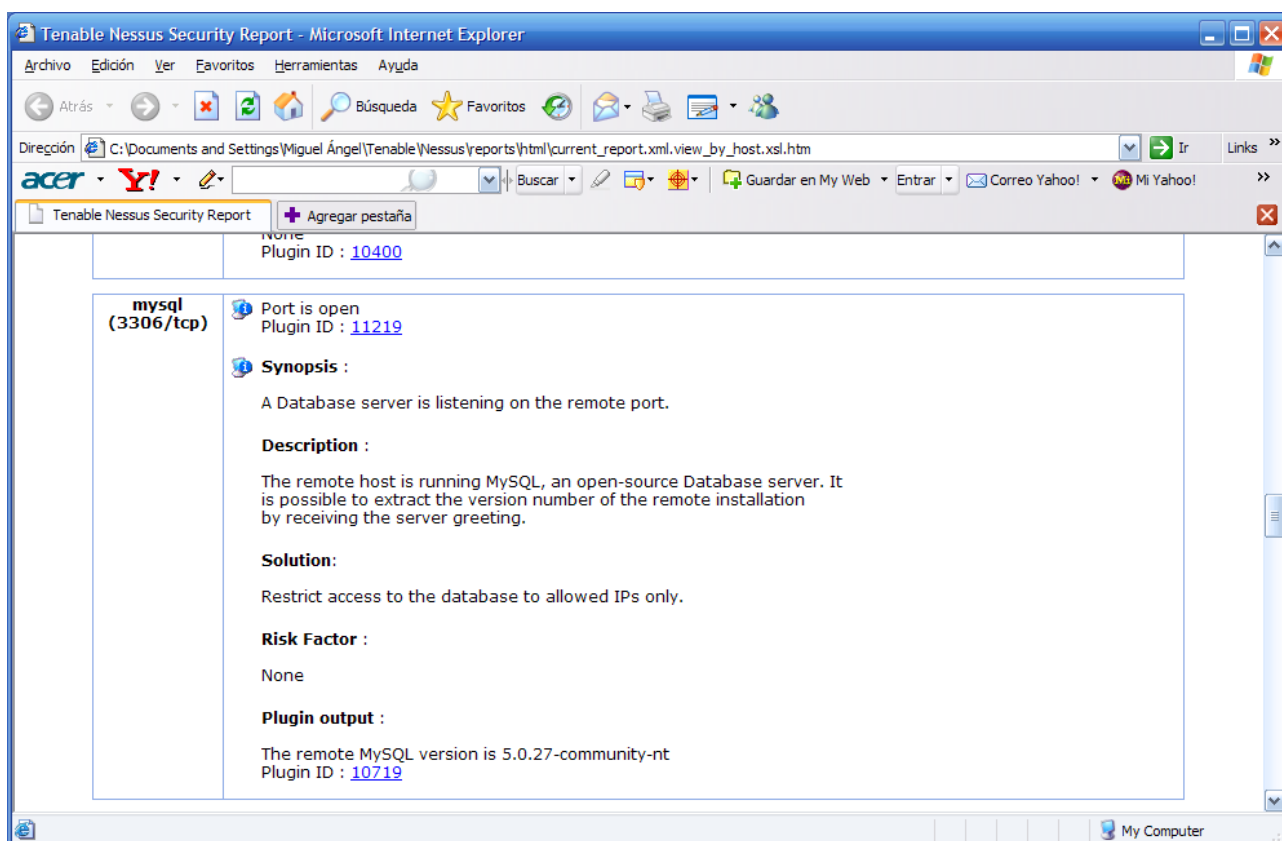


#### 4- Estos son los datos recogidos:



Los datos se muestran en una página HTML, y nos dice la máquina escaneada, los puertos abiertos detectados, ... Posteriormente comienza a explicar toda la información que ha podido conseguir sobre la máquina escaneada.

Información que se consigue del puerto de MySQL:



## 8. Conclusión

En este documento hemos querido dejar claro la utilidad de Snort, así como el trabajo que un administrador deberá realizar para que solamente se muestren las alertas que él, como administrador, considera que merecen la pena.

Por otro lado, también hemos querido mostrar, como la utilización de otras herramientas que complementen Snort, como MySQL y ACID harán que la tarea de análisis forense, para la toma de decisiones sea menos costosa, así como facilitar un tutorial para la instalación de Snort+MySQL+ACID.

También hemos mostrado los típicos, que no todos, ataques remotos, de manera que puede ser de utilidad el conocer los por menores de cada uno de ellos.

Y por último, hemos tratado de mostrar la utilidad de otra herramienta, Nessus, para la labor del administrador, así como mostrar los resultados que se darían en el caso de que un ataque remoto, en concreto un escaneo de puertos con su posterior exploit.